Refining Semantics for Multi-stage Programming

by

Rui Ge

B.Sc., Simon Fraser University, 2013
B.Eng., Zhejiang University, 2013

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate and Postdoctoral Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

October 2016

Abstract

Multi-stage programming is a programming paradigm that supports runtime code generation and execution. Though researchers have extended several mainstream programming languages to support it, multi-stage programming has not been widely recognised or used. The popularisation of multi-stage programming has been impeded by the lack of development aids such as code refactoring and optimisation, for which the culprit is the lack of static analysis support.

Van Horn and Might proposed a general-purpose approach to systematically developing static analyses for a programming language by applying transformations to its formal semantics, an approach we believe is applicable to multi-stage programming. The approach requires that the initial semantics be specified as an environmental abstract machine that records the change of control strings, environments and continuations as a program evaluates. Developing an environmental abstract machine for a multi-stage language is not straightforward and has not been done so far in the literature.

In the thesis, we study multi-stage programming through a functional language, MetaML. The main research problem of the thesis is:

> Can we refine the pre-existing substitutional natural semantics of MetaML to a corresponding environmental abstract machine and demonstrate their equivalence?

We first develop a substitutional structural operational semantics for MetaML. Then we simplify the research problem along two dimensions—each dimension leads to a less complicated semantics refinement problem. The first dimension is to refine semantics for a single-stage language rather than a multi-stage language: we stepwise develop an environmental abstract machine, the CEK machine, for a single-stage language, ISWIM, based on its substitutional structural operational semantics. The second dimension is to derive a substitutional abstract machine rather than an environmental abstract machine: we stepwise develop a substitutional abstract machine, the MK machine, for the multi-stage language MetaML, based on its substitutional structural operational semantics. Finally, utilising the experience of refining semantics along two dimensions, we stepwise develop an environmental abstract machine, the MEK machine, for MetaML, based on its substitutional structural operational semantics. Furthermore, we introduce three proof techniques which are used throughout the thesis to prove the equivalence of semantics.

Preface

This thesis is original, unpublished, independent work by the author, Rui Ge, under the supervision of Dr. Ronald Garcia (Assistant Professor at UBC).

Table of Contents

Appendices

List of Tables

List of Figures

Acknowledgements

I am greatly indebted to my supervisor, Dr. Ronald Garcia (Assistant Professor), for his continued support, guidance and encouragement over the past two and a half years. In the spring of 2014, I was fortunate in talking his postgraduate-level programming language course in which I was led into the world of programming language theory (PLT) for the first time. Since then, he has been supervising me on the project of refining semantics for multi-stage programming, which is presented by this thesis. I thank him for encouraging me to attend academic conferences and to interact with the academia, especially for financially supporting my participation in POPL '15 in Mumbai, India. I thank him for answering every single question of me thoroughly, for being forbearing when I contradicted him, for arousing my enthusiasm when I felt lazy, for comforting me when I had negative results, and for backing me up when I had hard times. I thank him for agreeing to review my thesis during his busiest time of the year, for patiently reviewing each section of my thesis in three to five iterations, and for providing numerous feedback on both the technical details and my writing style. I thank him for sharing the following quotation with me, which inspires me each and every day.

> *Nobody tells this to people who are beginners. I wish someone had told me. All of us who do creative work, we get into it because we have good taste. But there is this gap. For the first couple years you make stuff, it's just not that good. It's trying to be good, it has potential, but it's not. But your taste, the thing that got you into the game, is still killer. And your taste is why your work disappoints you. A lot of people never get past this phase, they quit. Most people I know who do interesting, creative work went through years of this. We know our work doesn't have this special thing that we want it to have. We all go through this. And if you are just starting out or you are still in this phase, you gotta know its normal and the most important thing you can do is do a lot of work. Put yourself on a deadline so that every week you finish one piece. It's only by going through a volume of work that you will close that gap, and your work will be as good as your ambitions. And I took longer to figure out how to do this than anyone I've ever met. It's gonna take awhile. It's normal to take awhile. You just gotta fight your way through.*
>
> — *Ira Glass*

I am grateful to the second reader, Dr. Ivan Beschastnikh (Assistant Professor), for reviewing my thesis and providing helpful feedback when he had a busy schedule out of town.

I thank Dr. Mark Greenstreet (Professor) and his students Ms. Jijie Wei and Ms. Yan Peng for helping me settle into the department. I also thank my tentative advisor, Dr. David Poole (Professor), for assisting me in academic decision making in the first eight months of the master's programme. Thanks to Dr. Ronald

Garcia (Assistant Professor), Dr. Alan Hu (Professor) and Dr. Steve Wolfman (Professor of Teaching) for recommending me for admission to the PhD programme. Thanks to Ms. Joyce Poon and Ms. Hermie Lam for their help in administrative affairs.

I thank my colleagues in the programming language group, especially Dr. Ronald Garcia (Assistant Professor), Dr. Joshua Dunfield (Research Associate), Mr. Felipe Banados Schwerter, Mr. Evgeny Roubinchtein, Mr. Jonatan Milewski and Mr. Khurram Ali Jafery for the insightful discussions in our weekly reading group meetings. I also thank my colleagues in the Software Practices Laboratory for motivating me to work hard. Thanks to Mr. Felipe Banados Schwerter for being in charge of the coffee machine and Mr. C. Albert Thompson for organising activities in the laboratory.

Employed as a teaching assistant for six terms, I thank the instructors and students for providing me the opportunities to strengthening my teaching skills.

Thanks to Dr. Evgenia Ternovska (Associate Professor), Dr. Wo-Shun Luk (Professor), Dr. Greg Baker (Senior Lecturer), Dr. Zonghua Gu (Associate Professor), Ms. Lieping Peng, Dr. Gencai Chen (Professor), Dr. Shuhua Liu (Associate Professor) and Dr. Yangzheng Wang (Professor) for their various kinds of support and encouragement when I studied at Simon Fraser University and Zhejiang University. Thanks to my friend, Mr. Yuke Zhu, for setting a good example to me in academia, and for motivating and supporting me to pursue an academic career.

Thanks to my friends and fellow students of the 2013 cohort at UBC, especially Mr. Zongxu Mu, Mr. Michael Ming-An Wu, Mr. Jonatan Milewski, Mr. Yifan Peng, Mr. Ben Zhu, Mr. Shuochen Su, Mr. Yuzan Yang, Ms. Yidan Liu, Ms. Kailun Zhang and Ms. Jianing Yu for the colourful times we have had.

Thanks to Mr. Wei Wang, Mr. Shuhan Wang, Ms. Mengran Guo and Mr. Siyu Wang for our unfailing friendship. Thanks to my dormitory roommates, including Mr. Yizhuo Hu, Mr. Zhenyu Xia, Mr. Sifang Liu, Mr. Boyu Li and Mr. Zhu Li, for the unforgettable experience of collective life.

Last but not least, I thank my family for their faith and love.

謹將此論文獻給父親母親

This thesis is dedicated to my parents.

Chapter 1

Introduction

To motivate the idea of multi-stage programming, we first demonstrate why general-purpose solutions to programming problems can be easier to implement and more reusable, but run more slowly than special-purpose solutions. We then discuss how a special-purpose program can be derived from a general-purpose program. Multi-stage programming exploits this possibility, allowing programmers to systematically transform a general-purpose program into a special-purpose program generator.

1.1 General-purpose and Special-purpose Programming

Consider the problem of computing a positive integer raised to a non-negative integer. Here are two possible solutions.[1]

Example 1. `power n x` returns the n^{th} power of `x`.

```
power n x
| n == 0 = 1
| otherwise = x * (power (n - 1) x)
```

Example 2. `power0 x` returns the 0^{th} power of `x`, `power1 x` returns the 1^{st} power of `x`, `power2 x` returns 2^{nd} power of `x`, and so on.

```
power0 x = 1
power1 x = x
power2 x = x * x
power3 x = x * x * x
......
power2016 x = x * x *...* x
......          (2016 x's)
```

As illustrated above, one may abstract the problem into a general problem that computes a positive integer raised to *any* non-negative integer. We call the `power` function a *general-purpose* program (or solution).

In contrast, one may split the problem into multiple special problems each of which computes a positive integer raised to a *fixed* non-negative integer. We call the functions `power0`, `power1`, `power2, ...`, `power2016, ...` *special-purpose* programs (or solutions).

[1] The code is presented in a Haskell-like syntax style.

For example, to compute the 2016^{th} power of natural numbers `[1..100]`, we can utilise either the general-purpose solution, the `power` function:

```
map (power 2016) [1..100]
```

or the special-purpose solution, the `power2016` function:

```
map power2016 [1..100]
```

As expected, they both produce the same correct result.

Now compare the performance of the general-purpose solution and the special purpose solution.

- Provided that most programmers have knowledge of branching and recursion, the general-purpose program `power` is easier to be implemented than the special-purpose program `power2016` because the former requires writing less code than the latter.
- The general-purpose program `power` is more reusable than the special-purpose program `power2016` because the `power` function is much more adaptable to similar computational tasks than the `power2016` function.
- The general-purpose program `power` runs more slowly than the special-purpose program `power2016`. For example, evaluating the `power` function with arguments `2016` and `100` involves `2016` recursive calls of the `power` function before getting the multiplication $\underbrace{100*100*...*100}_{\text{2016 100's}}*1$. In contrast, executing the special-purpose program `power2016` with argument `100` immediately computes the multiplication $\underbrace{100*100*...*100}_{\text{2016 100's}}$ without having any recursive calls or branching on an argument.

  A special-purpose program does not always run faster than a corresponding general-purpose program. In an extreme case, if the special-purpose program has an extremely large amount of code, it consumes a significant amount of time and space to load the code, in which case code explosion may cause the special-purpose program to have a higher execution cost.

The above presents a representative result of comparing a general-purpose program and a special-purpose program that solve the same problem. In most cases, a general-purpose program is easier to implement and more reusable, but less efficient than its special-purpose counterpart.

1.2 Specialising a General-purpose Program

Given a general-purpose program, we may predict the definite results of some of the ensuing computations. We can replace these computations with their anticipated values, which yields a special-purpose program. We call this process the *specialisation* of a general-purpose program. The program generated through specialisation inherits the advantage of running fast from ordinary special-purpose programs and avoids code explosion whenever possible.

Specialisation is a well-known technique in high-performance programming. For example, popular domain-specific languages for graphics programming such as OpenGL allows writing a general-purpose program does not reply on a particular underlying GPU architecture. A compiler completes the job of specialising a general-purpose program to a special program that is tailored to a specific GPU.

As an illustration, consider the problem of specialising the general-purpose `power` function, if there is a priori information that we will frequently compute the 2016^{th} power of any given positive integer. We go through two steps:

1. We define:

```
power2016' = \x -> (power 2016 x)
```

2. We unfold the body of the `power2016'` function so that the function `\x -> (power 2016 x)` becomes

$$\texttt{\textbackslash x ->}\ \underbrace{\texttt{x}*\texttt{x}*\ldots*\texttt{x}}_{\text{2016 x's}}*1$$

   which eliminates the relatively high runtime overhead associated with the `power` function that is mainly caused by repeated recursive calls.

We make two assumptions in order to be able to unfold the body of the `power2016'` function. The first assumption is that we can evaluate the body of a function at our convenience. This allows us to go inside a function and evaluate its body at the time that the function is defined. Under this assumption, when we define the function `power2016'`, we can evaluate the body `power 2016 x` of the function `\x -> (power 2016 x)`. The second assumption is that we can evaluate a function with partially known parameters, which is known as *partial evaluation* [JGS93]. Although the second parameter `x` of the `power` function is unknown, we can evaluate the partial application `power 2016 x` based on its first argument `2016`. Then `power 2016 x` is unfolded to the expression $\underbrace{\texttt{x}*\texttt{x}*\ldots*\texttt{x}}_{\text{2016 x's}}*1$. As a result under these two assumptions, the function `\x -> (power 2016 x)` evaluates to $\texttt{\textbackslash x ->}\ \underbrace{\texttt{x}*\texttt{x}*\ldots*\texttt{x}}_{\text{2016 x's}}*1$.

As a matter of fact, the essence of these two assumptions is to let programmers customise the order in which the terms of a program are evaluated. This capability constitutes the foundation for the programming paradigm of our interest, *multi-stage programming* (MSP) [TS97, Tah99a, She01, Tah04].

1.3 Multi-stage Programming

Multi-stage programming is a programming paradigm that supports runtime code generation and execution [Tah04]. Multi-stage programming supports meta-programming in the sense that a multi-stage program can be executed within a multi-stage program. A multi-stage language serves as both a meta-language and an object language.

Multi-stage programming also supports *program specialisation* [JGS93]. One can start programming with a general-purpose program and then specialise it based on the a priori information about partial inputs of the program. A well-written multi-stage solution to a programming problem is easier to implement and maintain, more reusable and reliable than a special-purpose solution, and runs faster than a general-purpose solution.

Multi-stage programming lets programmers control the order in which the terms of a program are evaluated in order to optimise the time and space resources consumed by evaluating the program [She98]. One can start programming with a conventional single-stage program and then specify the evaluation order of its terms with staging annotations to make the program multi-stage.

There have been developed several multi-stage programming languages and language extensions such as MetaML [TS97], MetaOCaml [Tah04], MetaHaskell [Mai12] and Mint [WRI+09]. We intensively study MetaML, a functional multi-stage language that extends ML, in the thesis.

To demonstrate what how a multi-stage program differs from its single-stage counterpart, we annotate the single-stage `power` and `powerN` functions with three staging annotations supported by MetaML: code, splice and run. Code, denoted by $\langle$ and $\rangle$, are for delaying a computation. Splice, denoted by $\sim$, is for combining delayed computations. Run, denoted by !, is for running a delayed computation.

Example 3. The multi-stage `power'` and `powerN'` functions are the single-stage `power` and `powerN` functions with staging annotations.

```
power' n x
| n == 0 = ⟨1⟩
| otherwise = ⟨∼x * ∼(power' (n - 1) x)⟩
powerN' n = ⟨\x -> ∼(power' n ⟨x⟩)⟩
```

For example, to compute the 2016^{th} power of 1, 2, ..., 100, we evaluate following two-stage program:

```
map !(powerN' 2016) [1..100]
```

At the first stage, `!(powerN' 2016)` evaluates to the function `\x ->` $\underbrace{\texttt{x}*\texttt{x}*\ldots*\texttt{x}}_{\text{2016 x's}}*1$. At the second stage, the program proceeds as:

$$\texttt{map (\x ->}\ \underbrace{\texttt{x}*\texttt{x}*\ldots*\texttt{x}}_{\text{2016 x's}}*1\texttt{) [1..100]}$$

The recursive calls to the `power'` function all happen intensively at the first stage when the run operation `!(powerN' 2016)` evaluates. They never happen again at the second stage.

We analyse this example in detail in the next chapter (Chapter 2).

1.4 Static Analysis of Multi-stage Programs

Although researchers have extended many mainstream programming languages to support multi-stage programming, multi-stage programming as a programming paradigm has not been widely recognised or used.

The popularisation of multi-stage programming has been impeded by the lack of development aids such as code refactoring and optimisation, for which the culprit is the lack of static analysis support. Our ultimate goal is to design a sound and decidable static analysis for a multi-stage programming language.

Recently, [VHM12] proposed a general-purpose approach to systematically developing static analyses for a programming language by applying transformations to its formal semantics. The approach requires that the initial semantics be specified as an environmental abstract machine that records the change of control strings, environments and continuations as a program evaluates. The transformations involve using a traditional store to collect execution information and associate it with the source program. After a number of transformations, the machine can be abstracted through its address allocation strategy. With this primary point of abstraction, we get a sound and decidable control flow analysis, where the preciseness of the analysis is determined by the chosen structure of the addresses used by the address allocator.

To apply the framework to multi-stage programming, we shall develop an environmental abstract machine for a multi-stage programming language. The abstract machine must be environmental because it uses environments to archive variable bindings, allowing us to trace the function calls in its history.

1.5 Refining Semantics

Developing an environmental abstract machine for a multi-stage programming language based on a pre-existing semantics is essentially a semantics refinement problem. A stepwise refinement process may produce several intermediate semantics before deriving the destination semantics. We briefly introduce four operational semantics that are usually used in solving a refinement problem.

1.5.1 Operational Semantics

To model the runtime behaviour of programs in a language, we can build an *operational semantics* for the language. Operational semantics include *big-step semantics* and *small-step semantics*.

- Big-step semantics is also known as *natural semantics* [Kah87]. It describes how the overall computation of a program takes place.

  For example, in a big-step semantics, the program `power 2016 100` big-steps to 100^{2016}.

- Small-step semantics include *structural operational semantics* [Plo81], *reduction semantics* [FH92] and *abstract machine semantics* [Lan64], ordered from the least practical to the most practical implementation strategies, all of which define how a single step of computation of a program takes place.

  - Structural operational semantics defines how a program computes with respect to the terms of the program in a syntax-oriented and inductive way.
  - Reduction semantics defines where a reduction may happen and what reduction may happen but does not provide a deterministic strategy for finding a place to perform reduction.
  - Abstract machine semantics refines reduction semantics in the sense that it provides a deterministic strategy for finding a place to perform reduction.

For example, in a small-step semantics, the program `power 2016 100` small-steps to `100 * (power 2015 100)`.

We differentiate a *substitutional* operational semantics from an *environmental* operational semantics.

- An operational semantics is substitutional if a function call is performed by substituting the function's parameters by arguments where substitutions may be modelled implicitly in the meta-language or explicitly [Ros96] as several steps of computation.

  For example, calling the function `\x -> x + 1` with the argument `x = 1` results in `1 + 1`. Then just looking at `1 + 1` itself does not tell whether a function call has been performed.

- An operational semantics is environmental if it uses an environment to keep track of variable bindings so that a function call is performed by updating the function's parameters in the environment to the function's arguments.

  For example, calling the function `\x -> x + 1` with the argument `x = 1` results in `x + 1` together with a variable binding that `x = 1`. Then just looking at `x + 1` together with the variable binding that `x = 1` implies that a function call with the argument `x = 1` has been performed.

1.5.2 Refining Semantics

Having what an operational semantics is in mind, we now consider how to develop an environmental abstract machine for a multi-stage programming language.

Main Semantics Refinement Problem. The multi-stage programming language that we study in the thesis is MetaML. We take the pre-existing substitutional natural semantics defined in [Tah99a] as our reference semantics for MetaML. The main research problem of the thesis which we call the *main semantics refinement problem* is:

> Can we refine the pre-existing substitutional natural semantics of MetaML to a corresponding environmental abstract machine and demonstrate their equivalence?

As an environmental abstract machine is a small-step operational semantics, its development is more natural and convenient to start from a structural operational semantics than a natural semantics. Thus the very first step to take is to derive a substitutional structural operational semantics for MetaML and show its equivalence with respect to the substitutional natural semantics. We illustrate this step in Table 1.1 and discuss it in detail in Chapter 2.

Developing an environmental abstract machine for MetaML is not straightforward and has not been done so far in the literature. We first attempt the main semantics refinement problem along two dimensions—each dimension leads to a less complicated semantics refinement problem.

Stepwise Developing the CEK Machine for ISWIM. Following the first dimension of simplifying the main semantics refinement problem, we study how to stepwise develop an environmental abstract machine for a single-stage language ISWIM [Lan66] rather than the multi-stage language MetaML. The problem is described as follows.

> Can we refine the substitutional structural operational semantics of ISWIM to a corresponding environmental abstract machine, which is known as the *CEK machine* [FF86], and demonstrate their equivalence?

We solve the above problem step by step as follows (see Table 1.1 for a summary).

1. ISWIM: Substitutions are defined in the meta-language level. Any instance of substitution in program immediately replaces every relevant variable as part of a single step of computation. We call this "inexplicit" in contrast to explicit substitutions, which we discuss next.

   We take the (inexplicitly) substitutional structural operational semantics of ISWIM as the starting point of solving the refinement problem for ISWIM.

2. Explicit ISWIM: Substitutions are defined as explicit object-language constructs. An instance of substitution replaces every relevant variable step by step by percolating explicit substitutions as several steps of computation. This provides a manageable step on the way to developing an environmental semantics.

3. Suspended ISWIM: Substitutions are modelled explicitly as in Explicit ISWIM and are suspended outside of a lambda abstraction until an application is performed. This step turns the semantics more environmental and makes the proofs of semantics equivalence tractable.

4. Environmental ISWIM: Environmental ISWIM turns explicit substitutions to environments.

   Environmental ISWIM is first derived as a structural operational semantics. We then systematically transform the semantics from structural operational semantics to reduction semantics and finally to an abstract machine. The abstract machine is also known as the CEK machine.

The stepwise development of CEK machine for ISWIM is discussed in detail in Chapter 3.

Stepwise Developing the MK Machine for MetaML. Following the second dimension of simplifying the main semantics refinement problem, we study how to stepwise develop a substitutional abstract machine rather than an environmental abstract machine for the multi-stage language MetaML. The problem is described as follows.

> Can we refine the substitutional structural operational semantics of MetaML to a corresponding substitutional abstract machine, which we call the *MK machine*, and demonstrate their equivalence?

As summarised in Table 1.2, starting from the substitutional structural operational semantics of MetaML, we first derive a substitutional reduction semantics and then a substitutional abstract machine, which we call the MK machine. The stepwise development of MK machine for MetaML is discussed in detail in Chapter 4.

Stepwise Developing the MEK Machine for MetaML. Utilising the experience of refining semantics along two dimensions, we eventually study how to stepwise develop an environmental abstract machine for the multi-stage language MetaML. The problem is described as follows.

> Can we refine the substitutional structural operational semantics of MetaML to a corresponding environmental abstract machine, which we call the *MEK machine*, and demonstrate their equivalence?

We attempt the above problem step by step as follows (see Table 1.3 for a summary).

1. MetaML: Substitutions are modelled inexplicitly, analogously to ISWIM.

   We take the (inexplicitly) substitutional structural operational semantics of MetaML as the starting point of refinement.

2. Explicit MetaML: Substitutions are modelled explicitly, analogously to Explicit ISWIM. This provides a manageable step on the way to developing an environmental semantics.

3. Suspended MetaML: Substitutions are modelled explicitly, analogously to Suspended ISWIM. This step turns the semantics more environmental and makes the proofs of semantics equivalence tractable.

4. Environmental MetaML: Environmental MetaML turns explicit substitutions to environments.

   Environmental MetaML is first derived as a structural operational semantics. We then systematically transform the semantics from structural operational semantics to reduction semantics and finally to an abstract machine. We call the abstract machine the MEK machine.

The stepwise development of MEK machine for MetaML is discussed in detail in Chapter 5.

Proving Equivalence of Semantics. We introduce and use throughout the thesis three proof techniques to prove the equivalence of two structural operational semantics, the equivalence of a structural operational semantics and a reduction semantics, and the equivalence of a reduction semantics and an abstract machine. We summarise the proof methodology in detail in Chapter 6.

No.	Language	Semantics	Substitutional or Environmental	Section
1	ISWIM	Structural Operational Semantics	(Inexplicitly) Substitutional	3.1
2	Explicit ISWIM	Structural Operational Semantics	Explicitly Substitutional	3.2
3	Suspended ISWIM	Structural Operational Semantics	Explicitly Substitutional	3.3
4	Environmental ISWIM	Structural Operational Semantics	Environmental	3.4
5	Environmental ISWIM	Reduction Semantics	Environmental	3.5
6	Environmental ISWIM	CEK Abstract Machine	Environmental	3.6

Table 1.1: Summary of semantics used to refine ISWIM's substitutional structural operational semantics to the CEK abstract machine.

No.	Language	Semantics	Substitutional or Environmental	Section
1	MetaML	Structural Operational Semantics	(Inexplicitly) Substitutional	2.2
2	MetaML	Reduction Semantics	(Inexplicitly) Substitutional	4.1
3	MetaML	MK Abstract Machine	(Inexplicitly) Substitutional	4.2

Table 1.2: Summary of semantics used to refine MetaML's substitutional structural operational semantics to the MK abstract machine.

No.	Language	Semantics	Substitutional or Environmental	Section
1	MetaML	Natural Semantics	(Inexplicitly) Substitutional	2.2
2	MetaML	Structural Operational Semantics	(Inexplicitly) Substitutional	2.2, 5.1
3	Explicit MetaML	Structural Operational Semantics	Explicitly Substitutional	5.2
4	Suspended MetaML	Structural Operational Semantics	Explicitly Substitutional	5.3
5	Environmental MetaML	Structural Operational Semantics	Environmental	5.4
6	Environmental MetaML	Reduction Semantics	Environmental	5.5
7	Environmental MetaML	MEK Abstract Machine	Environmental	5.6

Table 1.3: Summary of semantics used to solve the main semantics refinement problem.

Chapter 2

Formal Semantics of MetaML

This chapter introduces multi-stage programming and the formal semantics of MetaML. We first introduce the three staging annotations of MetaML and informally discuss how a multi-stage program evaluates using examples. Then we study the pre-existing substitutional natural semantics of MetaML [Tah99a] and present our newly developed substitutional structural operational semantics for MetaML. We finally demonstrate that the substitutional natural semantics and the substitutional structural operational semantics are equivalent.

The definitions in Section 2.1 are based on [She98, TS97]. The substitutional natural semantics of MetaML in Section 2.2 is based on [Tah99a].

2.1 Staging Annotations

MetaML uses staging annotations to explicitly control the evaluation order of terms of a program. Staging annotations include the code operator, the run operator and the splice operator.

Code Operation. A *code* operation, consisting of (1) the code operator "$\langle$" "$\rangle$" and (2) an operand, indicates delaying computing the operand. If a code operation evaluates to itself, it is a code value.

Example 4. `3 + 7` evaluates to `10`. The code operation $\langle$`3 + 7`$\rangle$ evaluates to itself because of delaying computing its operand `3 + 7`. Hence the code operation $\langle$`3 + 7`$\rangle$ is is also a code value.

Run Operation. A *Run* operation, consisting of (1) the run operator `!` and (2) an operand, indicates executing the delayed computation of the operand. A run operation expects its operand to reduce to a code operation. A run operation eliminates the code brackets from the result of evaluating its operand.

Example 5. `!`$\langle$`3 + 7`$\rangle$ evaluates to `10`. Its step-by-step reduction is as follows.

1 `!`$\langle$`3 + 7`$\rangle$

The run operator executes the delayed computation of the operand, `3 + 7`.

2 `3 + 7`

The addition operation `3 + 7` evaluates to `10`.

3 `10`

The natural number `10` is irreducible.

Splice Operation. An *splice* operation, consisting of (1) the splice operator "∼" and (2) an operand, indicates splicing the delayed computation produced by evaluating the operand into the current context. A splice operation may only appear in a delayed computation, i.e., under code brackets. Only a splice operation can be reduced under code brackets. A splice operation expects its operand to reduce to a code value. A splice operation eliminates the code brackets from the result of evaluating its operand.

Example 6. ⟨ ∼⟨3 + 7⟩ * ∼⟨3 + 7⟩ ⟩ evaluates to ⟨ (3 + 7) * (3 + 7) ⟩. Its step-by-step reduction is as follows.

1 ⟨ ∼⟨3 + 7⟩ * ∼⟨3 + 7⟩ ⟩

The first escape operator splices the delayed computation of the operand, 3 + 7, into the context surrounded by code brackets.

2 ⟨ (3 + 7) * ∼⟨3 + 7⟩ ⟩

The second escape operator splices the delayed computation of the operand, 3 + 7, into the context surrounded by code brackets.

3 ⟨ (3 + 7) * (3 + 7) ⟩

The code operation ⟨ (3 + 7) * (3 + 7) ⟩ is irreducible.

Level of a Term. To explicitly regulate under what circumstances the run operation eliminates code and the splice operation combines code, we introduce the concept of levels. The *level* of a term is the difference of the number of surrounding brackets and the number of surrounding escapes.

Example 7. (1) 3 in ⟨3 + 7⟩ is at level 1.

(2) The first 3 in ⟨ ∼⟨3 + 7⟩ * ∼⟨3 + 7⟩ ⟩ is at level $2-1=1$.

(3) The first 3 in ⟨ (3 + 7) * (3 + 7) ⟩ is at level 1.

(4) 3 in !⟨3 + 7⟩ is at level 1.

(5) The function \x -> x in !⟨ ∼⟨3 + 7⟩ * ∼((\x -> x) ⟨3 + 7⟩) ⟩ is at level $1-1=0$.

Roughly speaking, a splice operation ...∼⟨*t*⟩... reduces to ...*t*... only if ∼⟨*t*⟩ is at level 1, and a run operation ...!⟨*t*⟩... reduces to ...*t*... only if !⟨*t*⟩ is at level 0, where *t* is an arbitrarily legal term. Level 0 corresponds to single-stage programming. An function call or an addition only reduces at level 0. For example, 3 + 7 reduces to 10 at level 0 and is irreducible at other levels.

Example 8. !⟨ ∼⟨3 + 7⟩ * ∼((\x -> x) ⟨3 + 7⟩) ⟩ evaluates to 100. Its step-by-step reduction is as follows. The reducible term of each step is underlined.

1 !⟨ ∼⟨3 + 7⟩ * ∼((\x -> x) ⟨3 + 7⟩) ⟩

In the splice operation ∼⟨3 + 7⟩, the delayed computation of the operand 3 + 7 is spliced into the context.

2 !⟨ (3 + 7) * ∼((\x -> x) ⟨3 + 7⟩) ⟩

The application (\x -> x) ⟨3 + 7⟩ reduces to ⟨3 + 7⟩.

3 !⟨ (3 + 7) * ∼⟨3 + 7⟩ ⟩

In the splice operation ∼⟨3 + 7⟩, the delayed computation of the operand 3 + 7 is spliced into the context.

4 `!⟨ (3 + 7) * (3 + 7) ⟩`

The run operator executes the delayed computation of the operand `(3 + 7) * (3 + 7)`.

5 `(3 + 7) * (3 + 7)`

The first operand of the multiplication operation `3 + 7` reduces to `10`.

6 `10 * (3 + 7)`

The second operand of the multiplication operation `3 + 7` reduces to `10`.

7 `10 * 10`

The multiplication operation `10 * 10` reduces to `100`.

8 `100`

The natural number `100` irreducible.

The code operation introduces a code value, the run operation eliminates a code value, and the splice operation combines code values. Taking the above evaluation as an example, in the first step, the outermost code operation generates a code value in which two inner code operations introduce two smaller code values. In the third step, the splice operation combines code values. In the fourth step, the run operation executes a code value.

Evaluating a Multi-stage Program. A multi-stage program can be constructed from a conventional single-stage program by manually adding staging annotations [TS97]. We demonstrate how a multi-stage program evaluates through Example 3 from Chapter 1, which is re-described below as Example 9.

Example 9. The multi-stage `power'` and `powerN'` functions are the single-stage `power` and `powerN` functions with staging annotations.

```
power' n x
| n == 0 = ⟨1⟩
| otherwise = ⟨∼x * ∼(power' (n - 1) x)⟩
powerN' n = ⟨\x -> ∼(power' n ⟨x⟩)⟩
```

The `power'` function is a special-purpose program generator. When the parameter `n` of the `powerN'` function is known, `!(powerN n)` generates a special-purpose program that computes its parameter raised to `n`. For example, to compute the 4^{th} power of an integer, `!(powerN' 4)` eventually reduces to `\x -> (x * x * x * x * 1)`. Its step-by-step reduction is as follows. The reducible term of each step is underlined.

```
1  !(powerN' 4)
2  !⟨ \x -> ∼(power' 4 ⟨x⟩) ⟩
3  !⟨ \x -> ∼(⟨ ∼⟨x⟩ * ∼(power' 3 ⟨x⟩) ⟩) ⟩
4  !⟨ \x -> ∼(⟨ x * ∼(power' 3 ⟨x⟩) ⟩) ⟩
5  !⟨ \x -> ∼(⟨ x * ∼(⟨ ∼⟨x⟩ * ∼(power' 2 ⟨x⟩) ⟩) ⟩) ⟩
6  !⟨ \x -> ∼(⟨ x * ∼(⟨ x * ∼(power' 2 ⟨x⟩) ⟩) ⟩) ⟩
7  !⟨ \x -> ∼(⟨ x * ∼(⟨ x * ∼(⟨ ∼⟨x⟩ * ∼(power' 1 ⟨x⟩) ⟩) ⟩) ⟩) ⟩
8  !⟨ \x -> ∼(⟨ x * ∼(⟨ x * ∼(⟨ x * ∼(power' 1 ⟨x⟩) ⟩) ⟩) ⟩) ⟩
```

```
9   !⟨ \x -> ∼(⟨ x * ∼(⟨ x * ∼(⟨ x * ∼(⟨ ∼⟨x⟩ * ∼(power' 0 ⟨x⟩) ⟩) ⟩) ⟩) ⟩) ⟩
10  !⟨ \x -> ∼(⟨ x * ∼(⟨ x * ∼(⟨ x * ∼(⟨ x * ∼(power' 0 ⟨x⟩) ⟩) ⟩) ⟩) ⟩) ⟩
11  !⟨ \x -> ∼(⟨ x * ∼(⟨ x * ∼(⟨ x * ∼(⟨ x * ∼(⟨1⟩) ⟩) ⟩) ⟩) ⟩) ⟩
12  !⟨ \x -> ∼(⟨ x * ∼(⟨ x * ∼(⟨ x * ∼(⟨ x * 1⟩) ⟩) ⟩) ⟩) ⟩
13  !⟨ \x -> ∼(⟨ x * ∼(⟨ x * ∼(⟨ x * x * 1 ⟩) ⟩) ⟩) ⟩
14  !⟨ \x -> ∼(⟨ x * ∼(⟨ x * x * x * 1 ⟩) ⟩) ⟩
15  !⟨ \x -> ∼(⟨ x * x * x * x * 1 ⟩) ⟩
16  !⟨ \x -> (x * x * x * x * 1) ⟩
17  \x -> (x * x * x * x * 1)
```

We make a few observations from the example above. (1) A run operation expects its operand to reduce to a code operation, as illustrated by Steps 1 to 16. (2) A splice operation expects its operand to reduce to a code operation, as illustrated by Steps 2 to 15, Steps 4 to 14, Steps 6 to 13, Steps 8 to 12 and Steps 10 to 11. (3) A run operation eliminates the code brackets of its operand only if the operand is irreducible, as illustrated by Steps 1 to 16. (4) A splice operation eliminates the code brackets of its operand only if the operand is irreducible, as illustrated by Steps 2 to 15, Steps 4 to 14, Steps 6 to 13, Steps 8 to 12 and Steps 10 to 11.

Informal Reasoning is Insufficient. Our explanation about how a multi-stage program evaluates is informal. One may ask: what computation can be performed in the operand of a code operation, a splice operation or a run operation? When can the body of a function be evaluated? Is our evaluation strategy deterministic? How do we implement our evaluation strategy?

Furthermore, it is tedious and error-prone to evaluate unintuitive programs such as $!\langle \lambda y. \sim ((\lambda x.\langle x \rangle)\,(\lambda x.\langle y \rangle))\,0 \rangle\,5$, which was introduced in [Tah99a], through informal reasoning as Example 9. To rigorously explain how a multi-stage program evaluates, we need to define a formal semantics.

2.2 Formal Semantics of MetaML

Previously, we have informally discussed how MetaML works as a multi-stage programming language. In this section, we explain the formal semantics of MetaML: its syntax, substitutional natural semantics and substitutional structural operational semantics. For the sake of simplicity and convenience, we make two minor modifications to the formal specifications of MetaML presented in [Tah99a]: we add natural numbers and addition to the language, and we discard the fixed point operator from the language as a fixed point operator can be implemented using the Y combinator.

2.2.1 Syntax

We first define the basic syntax of the language: terms, values and denotable terms. Then we present some properties implied by the definitions. Finally we define the free variable function and the substitution function.

2.2.1.1 Terms

We start with two sets: the set of variables, VAR, and the set of natural numbers, $\mathbb{N}$.

Definition 10 (Terms)**.** Let TERM be the set of terms.

$$x \in \text{VAR},\ n \in \mathbb{N},\ i \in \mathbb{N},\ t \in \text{TERM}$$

$$t \quad ::= \quad x \mid t\ t \mid \lambda x.t \mid \langle t \rangle \mid {\sim}t \mid\ !t \mid n \mid t + t$$

The definition tells that a term can be (1) a variable, (2) an application in which both the operator and the operand are terms, (3) a lambda abstraction whose body is a term, (4) a code operation whose operand is a term, (5) a splice operation whose operand is a term, (6) a run operation whose operand is a term, (7) a natural number, or (8) an addition operation whose two operands are terms.

Recall that the level of a term is the difference of the number of surrounding brackets and the number of surrounding escapes. A term cannot occur at the position of an arbitrary level. For example, a splice operation cannot occur at a level-0 position because it must be in some code operation. We use levels as indexes to finely distinguish subclasses of terms.

Definition 11 (Level-indexed Terms)**.** Let TERM^i be the set of terms at level i.

$$\frac{}{x \in \text{TERM}^i} \qquad \frac{t_1 \in \text{TERM}^i \quad t_2 \in \text{TERM}^i}{t_1\ t_2 \in \text{TERM}^i} \qquad \frac{t \in \text{TERM}^i}{\lambda x.t \in \text{TERM}^i} \qquad \frac{t \in \text{TERM}^{i+1}}{\langle t \rangle \in \text{TERM}^i} \qquad \frac{t \in \text{TERM}^i}{{\sim}t \in \text{TERM}^{i+1}}$$

$$\frac{t \in \text{TERM}^i}{!t \in \text{TERM}^i} \qquad \frac{}{n \in \text{TERM}^i} \qquad \frac{t_1 \in \text{TERM}^i \quad t_2 \in \text{TERM}^i}{t_1 + t_2 \in \text{TERM}^i}$$

The definition tells that (1) a variable can be at an arbitrary level, (2) an application can be at the any level that its operator and operand share, (3) a lambda abstraction can be at any level that its body can, (4) a code operation can be at any level that is one level lower than its operand, (5) a splice operation can be at any level that is one level higher than its operand, (6) a run operation is can be at any level that its operand can, (7) a natural number can be at an arbitrary level, and (8) an addition can be at any level that its two operands share.

The definition ensures that a splice operation cannot be a term at level 0. A splice combines a delayed computation into the context surrounded by code brackets. Hence the context of a splice has to be at some level higher than 0.

We use t^i with or without any subscript or any other superscript as a metavariable to range over TERM^i. We continue to use t with or without any subscript or superscript as a metavariable to range over TERM.

We make two observations about terms and level-indexed terms. (1) The sets TERM^i contain strictly more terms as the level i increases. For example, the term ${\sim}x$ can be at any level higher than 0 and the term ${\sim}{\sim}x$ can be at any level higher than 1. Our syntax lets us write code generators that generates code generators. (2) Every term has at least one level. To make a conventional single-stage program multi-stage, we can experiment annotating the program with various combinations of staging annotations as long as the program is at level 0. We formalise our observations as the following propositions.

Proposition 12. *For any* $i \in \mathbb{N}$, $\text{TERM}^i \subset \text{TERM}^{i+1}$.

Proposition 13. $\text{TERM} = \bigcup_{i \in \mathbb{N}} \text{TERM}^i$.

2.2.1.2 Values

A value at level i is a term at level i that represents a result of computation at its indicated level. Thus the definition of values closely relates to the operational semantics of the language.

Definition 14 (Level-indexed Values)**.** Let VALUE^i be the set of values at level i.

$$\frac{}{x \in \text{VALUE}^{i+1}} \qquad \frac{t_1 \in \text{VALUE}^{i+1} \quad t_2 \in \text{VALUE}^{i+1}}{t_1\, t_2 \in \text{VALUE}^{i+1}} \qquad \frac{}{\lambda x.t \in \text{VALUE}^0} \text{ where } t \in \text{TERM}^0$$

$$\frac{t \in \text{VALUE}^{i+1}}{\lambda x.t \in \text{VALUE}^{i+1}} \qquad \frac{t \in \text{VALUE}^{i+1}}{\langle t \rangle \in \text{VALUE}^{i}} \qquad \frac{t \in \text{VALUE}^{i+1}}{\sim t \in \text{VALUE}^{i+2}} \qquad \frac{v \in \text{VALUE}^{i+1}}{!v \in \text{VALUE}^{i+1}} \qquad \frac{}{n \in \text{VALUE}^{i}}$$

$$\frac{t_1 \in \text{VALUE}^{i+1} \quad t_2 \in \text{VALUE}^{i+1}}{t_1 + t_2 \in \text{VALUE}^{i+1}}$$

Recall that in the multi-stage language MetaML, level 0 corresponds to single-stage programming, and levels higher than zero delay some computation. We briefly explain the intuition behind the above definition. (1) A variable is not a value at level 0, analogous to the fact that a variable is not a result of computation in single-stage programming. At higher levels, we delay dereferencing a variable, making the variable a value. (2) As a lambda abstraction is a value in single-stage programming, it is a value at level 0. At higher levels, it is a value if its body is a value. (3) For an application or an addition, an interesting reduction may only happen at level 0. At higher levels, it is a value if its immediate substructures are values. (4) For a code operation $\langle t \rangle$, it is a code value if its operand t is irreducible one level up. (5) For a splice operation and a run operation, an interesting reduction may only happen at level 1 and level 0 respectively. For any higher level, a splice/run operation is a value if its operand is a value. (6) A natural number is always a value.

We use v^i with or without any subscript or any other superscript as a metavariable to range over VALUE^i.

Definition 15 (Values)**.** Let VALUE be the set of values.

$$\text{VALUE} = \textstyle\bigcup_i \text{VALUE}^i \text{ where } i \in \mathbb{N}$$

We use v with or without any subscript or superscript as a metavariable to range over VALUE.

To demonstrate the relationship between terms and values, we make three observations. (1) There is a one-to-one correspondence between the subclasses of terms and the subclasses of values. One the one hand, a value at some level is a term at the next lower level. This justifies the semantics of run: if we have a level-i value $\langle t \rangle$, then t is a level-$(i+1)$ value but not necessarily a level-i value. Removing the brackets makes t a level-i term and allows us to reduce t at level i. Although running happens only at level 0, this uniformity makes it clear that we can reason seamlessly about multi-stage programs at levels higher than 0 and 1 in the same way that you reason about two-stage programs. One the other hand, a term at some level is a value at the next higher level. We can always delay a computation by putting a pair of brackets around it. (2) A value at some level represents a result of computation at that level. Hence a value at some level must be a term at that level. This corresponds to our intuition and later formalisation of evaluating at a level. (3) Terms and values are represented by the exact same set of syntactic symbols. More importantly, every term is a value

at some level. It is always possible to turn a term to a value by putting a finite number of brackets around it. We list our observations as the following propositions.

Proposition 16. *For any* $i \in \mathbb{N}$, $\text{VALUE}^{i+1} = \text{TERM}^i$.

Proposition 17. *For any* $i \in \mathbb{N}$, $\text{VALUE}^i \subset \text{TERM}^i$.

Proposition 18. $\text{TERM} = \text{VALUE}$.

2.2.1.3 Denotable Terms

The denotable terms are terms that are substituted for variables in the semantics.

Definition 19 (Denotable Terms)**.** Let DENOTABLE be the set of denotable terms.

$$\text{DENOTABLE} = \text{VAR} \cup \text{VALUE}^0$$

There are two circumstances that a variable is substituted. If a variable gets renamed, the variable is substituted by a variable. If a lambda abstraction is applied to a value at level 0, then the lambda bound variable is substituted by the value. Hence a variable may refer to a variable or a value at level 0. We use w with or without any subscript or superscript as a metavariable to range over DENOTABLE.

2.2.1.4 Free Variable Function

We define $FV(t)$ to be the set of all variables that occur free in the term t.

Definition 20 (Free Variable Function)**.** Let the free variable function FV be a total function from the set of terms to the power set of variables.

$$FV : \text{TERM} \to \mathscr{P}(\text{VAR})$$

$$\begin{aligned}
FV(x) &= x && (1)\\
FV(t_1\ t_2) &= FV(t_1) \cup FV(t_2) && (2)\\
FV(\lambda x.t) &= FV(t) \backslash \{x\} && (3)\\
FV(\langle t \rangle) &= FV(t) && (4)\\
FV(!t) &= FV(t) && (5)\\
FV(\sim t) &= FV(t) && (6)\\
FV(n) &= \emptyset && (7)\\
FV(t_1 + t_2) &= FV(t_1) \cup FV(t_2) && (8)
\end{aligned}$$

Example 21. $FV(\lambda x.(x+y)) = FV(x+y)\backslash\{x\} = (FV(x) \cup FV(y))\backslash\{x\} = (\{x\} \cup \{y\})\backslash\{x\} = \{x,y\}\backslash\{x\} = \{y\}$.

Example 22. $FV((\lambda x.x)\ x) = FV(\lambda x.x) \cup FV(x) = (FV(x)\backslash\{x\}) \cup \{x\} = (\{x\}\backslash\{x\}) \cup \{x\} = \emptyset \cup \{x\} = \{x\}$.

Definition 23 (Closed Terms)**.** A term t is *closed* if and only if $FV(t) = \emptyset$.

2.2.1.5 Substitution Function

We define $t[w/x]$ to be the result of substituting the denotable term w for each free occurrence of the variable x in the term t.

Definition 24 (Substitution Function)**.** Let the substitution function $\cdot[\cdot/\cdot]$ be a total function from the 3-tuple of the set of terms, the set of denotable terms and the set of variables, to the set of terms.

$$\cdot[\cdot/\cdot] : (\textsc{Term} \times \textsc{Denotable} \times \textsc{Var}) \to \textsc{Term}$$

$$\begin{array}{rcll}
x[w/x] & = & w & (1)\\
x_1[w/x_2] & = & x_1 \text{ where } x_1 \not\equiv x_2 & (2)\\
(t_1\ t_2)[w/x] & = & (t_1[w/x])\ (t_2[w/x]) & (3)\\
(\lambda x_1.t_0)[w/x_2] & = & \lambda x_3.t_0[x_3/x_1][w/x_2] & \\
 & & \text{where } x_3 \notin FV(\lambda x_1.t_0) \cup FV(w) \cup \{x_2\} & (4)\\
\langle t_0 \rangle[w/x] & = & \langle t_0[w/x] \rangle & (5)\\
(!t_0)[w/x] & = & !t_0[w/x] & (6)\\
(\sim t_0)[w/x] & = & \sim t_0[w/x] & (7)\\
n[w/x] & = & n & (8)\\
(t_1 + t_2)[w/x] & = & (t_1[w/x]) + (t_2[w/x]) & (9)
\end{array}$$

Equation (4) could be replaced by the following two rules:

$$\begin{array}{rcll}
(\lambda x_1.t_0)[w/x_2] & = & \lambda x_1.t_0 \text{ where } x_1 \equiv x_2 & (4\text{-}1)\\
(\lambda x_1.t_0)[w/x_2] & = & \lambda x_3.t_0[x_3/x_1][w/x_2] \text{ where } x_1 \not\equiv x_2 \text{ and } x_3 \notin FV(\lambda x_1.t_0) \cup FV(w) \cup \{x_2\} & (4\text{-}2)
\end{array}$$

The above two rules do not rename a lambda bound variable when it does not have to. For example, by (4) we have:

$$(\lambda x_1.x_1)[x_1/x_1] = \lambda x_2.x_1[x_2/x_1][x_1/x_1] = \lambda x_2.x_2 \text{ where } x_1 \not\equiv x_2$$

In contrast, by (4-1), we have:

$$(\lambda x_1.x_1)[x_1/x_1] = \lambda x_1.x_1$$

For the sake of proof simplicity, we do not replace (4) by (4-1) and (4-2). It can be proved that these two forms always produce alpha equivalent results. Definition 130 defines the alpha equivalence relation.

Strictly speaking, the definition itself does not define a function. In (4), x_3 can be an arbitrary variable as long as $x_3 \notin FV(\lambda x_1.t_0) \cup FV(w) \cup \{x_2\}$. The definition indeed defines a relation rather than a function. To make what we have defined become a real function, we can impose a total order on variables, i.e., VAR is a total order. In (4), let x_3 be the least element from VAR such that $x_3 \notin FV(\lambda x_1.t_0) \cup FV(w) \cup \{x_2\}$. As a result, $\cdot[\cdot/\cdot]$ truly becomes a function. However, this approach makes the later proofs overcomplicated.

Alternatively, we can equate all terms that only differ in the names of their lambda bound variables, i.e., we equal all α-equivalent terms. Then, a term is a representative of its α-equivalent class. Recall that the destination semantics of the main semantics refinement problem is an abstract machine, which is close

to a practical implementation. We do not use this approach because meta-linguistic equivalence classes of variables do not map well to a concrete implementation.

At this moment, we are not concerned with how a fresh variable is chosen so we call $\cdot[\cdot/\cdot]$ the substitution function.

Example 25. Let $x \not\equiv y$. Then $(\lambda x.x+y)[x/y] = \lambda z.(x+y)[z/x][x/y] = \lambda z.(x[z/x]+y[z/x])[x/y] = \lambda z.(z+y)[x/y] = \lambda z.(z[x/y]+y[x/y]) = \lambda z.z+x$ where $z \not\equiv y$.

2.2.2 Substitutional Natural Semantics

We lay out the substitutional natural semantics through a family of level-indexed big-step relations.

Definition 26 (Level-indexed Big-step Relations)**.** For any $i \in \mathbb{N}$, define the level-indexed big-step relation $\Downarrow^i$ be a binary relation between the set of terms at level i and the set of values at level i.

$$\Downarrow^i \subseteq \textsc{Term}^i \times \textsc{Value}^i$$

$$\frac{}{\lambda x.t^0 \Downarrow^0 \lambda x.t^0}\ \text{(lambda-0)} \qquad \frac{t_1^{i+1} \Downarrow^{i+1} v_2^{i+1}}{\lambda x.t_1^{i+1} \Downarrow^{i+1} \lambda x.v_2^{i+1}}\ \text{(lambda-(i+1))}$$

$$\frac{t_1^0 \Downarrow^0 \lambda x.t_{11}^0 \quad t_2^0 \Downarrow^0 v_2^0 \quad t_{11}^0[v_2^0/x] \Downarrow^0 v^0}{t_1^0\ t_2^0 \Downarrow^0 v^0}\ \text{(app-0)} \qquad \frac{t_1^{i+1} \Downarrow^{i+1} v_1^{i+1} \quad t_2^{i+1} \Downarrow^{i+1} v_2^{i+1}}{t_1^{i+1}\ t_2^{i+1} \Downarrow^{i+1} v_1^{i+1}\ v_2^{i+1}}\ \text{(app-(i+1))}$$

$$\frac{t_1^0 \Downarrow^0 \langle v_1^1 \rangle \quad v_1^1 \Downarrow^0 v_2^0}{!t_1^0 \Downarrow^0 v_2^0}\ \text{(run-0)} \qquad \frac{t^{i+1} \Downarrow^{i+1} v^{i+1}}{!t^{i+1} \Downarrow^{i+1} !v^{i+1}}\ \text{(run-(i+1))}$$

$$\frac{t^{i+1} \Downarrow^{i+1} v^{i+1}}{\langle t^{i+1} \rangle \Downarrow^i \langle v^{i+1} \rangle}\ \text{(code-i)}$$

No (splice-0) $\qquad \frac{t^0 \Downarrow^0 \langle v^1 \rangle}{\sim t^0 \Downarrow^1 v^1}\ \text{(splice-1)} \qquad \frac{t^{i+1} \Downarrow^{i+1} v^{i+1}}{\sim t^{i+1} \Downarrow^{i+2} \sim v^{i+1}}\ \text{(splice-(i+2))}$

No (ref-0) $\qquad \frac{}{x \Downarrow^{i+1} x}\ \text{(ref-(i+1))}$

$$\frac{}{n \Downarrow^i n}\ \text{(num-i)}$$

$$\frac{t_1^0 \Downarrow^0 n_1 \quad t_2^0 \Downarrow^0 n_2}{t_1^0 + t_2^0 \Downarrow^0 n}\ \text{where } n = n_1 + n_2\ \text{(plus-0)} \qquad \frac{t_1^{i+1} \Downarrow^{i+1} v_1^{i+1} \quad t_2^{i+1} \Downarrow^{i+1} v_2^{i+1}}{t_1^{i+1} + t_2^{i+1} \Downarrow^{i+1} v_1^{i+1} + v_2^{i+1}}\ \text{(plus-(i+1))}$$

The big-step relation $t_1^i \Downarrow^i v_2^i$ reads as "t_1 big-steps to v_2 at level i", meaning that the value v_2 is the result of computing the term t_1 at level i.

Recall that a term can be a variable x, an application $t_1\ t_2$, a lambda abstraction $\lambda x.t$, a code operation $\langle t \rangle$, a splice operation $\sim t$, a run operation $!t$, a natural number n and an addition operation $t_1 + t_2$. We explain how an arbitrary term gets evaluated by the above relation.

Given a variable x, we may use the (ref-(i+1)) depending on its level. (1) If the variable x is at level 0, we get stuck because there is no (ref-0) rule. (2) If the variable is at level $i+1$, the variable evaluates to itself as it is a value at level $i+1$.

Given an application $t_1\ t_2$, we may evaluate it using the (app-0) rule or the (app-(i+1)) rule depending on the current level. (1) If the application is at level 0, we first evaluate its operator t_1 at level 0 to a lambda abstraction $\lambda x.t_{11}$ and evaluate its operand t_2 at level 0 to a value v_2. Then we substitute every free occurrence of the variable x in the term t_{11} by the value v_2, denoted by $t_{11}[v_2/x]$. Finally we evaluate $t_{11}[v_2/x]$ at level 0 to a value v, which is the final result of evaluating the application $t_1\ t_2$ at level 0. Evaluating an application at level 0 is the same as evaluating an application in a single-stage language. (2) If the application is at level $i+1$, we evaluate its operator t_1 at level $i+1$ to a value v_1 and evaluate its operand t_2 at level $i+1$ to a value v_2. The application $v_1\ v_2$ is a value at level $i+1$, which is the result of evaluating the application $t_1\ t_2$ at level $i+1$. We do not perform any application at levels higher than 0.

Given a lambda abstraction $\lambda x.t$, we may evaluate it using the (lambda-0) rule or the (lambda-(i+1)) rule depending on its level. (1) If the lambda abstraction is at level 0, then it evaluates to itself as it is a value at level 0. (2) If the lambda abstraction is at level $i+1$, we evaluate its body t to at level $i+1$ a value v, corresponding to the intuition that we may evaluate the body of a function at levels higher than 0. Then the lambda abstraction $\lambda x.v$ is the result of evaluating the lambda abstraction $\lambda x.t$ at level $i+1$.

Given a code operation $\langle t \rangle$ at level i, we may evaluate it using the (code-i) rule. We evaluate the body of the code operation at level $i+1$ to a value v. Then bracketing the value gives the result of evaluating the code operation at level i.

Given a splice operation $\sim t$, we may evaluate it using the (splice-1) rule or the (splice-(i+2)) rule depending on the current level. (1) If the splice operation is at level 0, we get stuck because there is no (splice-0) rule. (2) If the splice operation is at level 1, we evaluate its operand at level 0 to a bracketed value, corresponding to the intuition that a splice operation expects its operand to be a code operation. The bracketed value is spliced into the context of the splice operation and its the result of evaluating the splice operation $\sim t$. (3) If the splice operation is at level $i+2$, we evaluate its operand t at level $i+1$ to a value v. Then the splice operation $\sim v$ is the result of evaluating the splice operation $\sim t$ at level $i+2$.

Given a run operation $!t$, we may evaluate it using the (run-0) rule or the (run-(i+1)) rule depending on the current level. (1) If the run operation is at level 0, we first evaluate its operand at level 0 to a bracketed value, corresponding to the intuition that a run operation expects its operand to be a code operation. Then we evaluate the bracketed value at level 0 (rather than level 1), corresponding to the intuition that a run operation executes a code operation. What this step returns is the result of evaluating the run operation $!t$ at level 0. (2) If the run operation is at level $i+1$, we evaluate its operand t at level $i+1$ to a value v. Then the run operation $!v$ is the result of evaluating the run operation $!t$ at level $i+1$.

Given a natural number n, we may evaluate it using the (num-i). The natural number itself is a value and evaluates to itself regardless of its level.

Given an addition operation $t_1 + t_2$, we may evaluate it using the (plus-0) rule or the (plus-(i+1)) rule depending on the current level. (1) If the addition is at level 0, we first evaluate its first operand t_1 at level 0 to a natural number n_1 and evaluate its second operand t_2 at level 0 to a natural number n_2. Then we compute

n_1 plus n_2 and get its result n. The natural number n is the result of evaluating the addition operation $t_1 + t_2$ at level 0. Evaluating an addition at level 0 is the same as evaluating an addition in a single-stage language. (2) If the addition is at level $i+1$, we evaluate its first operand t_1 at level $i+1$ to a value v_1 and evaluate its second operand t_2 at level $i+1$ to a value v_2. The addition operation $v_1 + v_2$ is a value at level $i+1$, which is the result of evaluating the addition $t_1 + t_2$ at level $i+1$. We do not perform any addition at levels higher than 0.

Observations. The rules of the big-step relations can be classified into three categories. (1) The rules (lambda-0), (ref-(i+1)) and (num-i) are *value* rules where the term being evaluated is a value at its indicated level. (2) All the rules except the three axiomatic rules are *structural* rules which define how to evaluate a term with respect to its sub-terms. (3) The rules (app-0), (run-0), (splice-1) and (plus-0) are *reduction* rules which perform a real step of computation.

The big-step relations tell that the language is call-by-value. In (app-0) the variable x is substituted by the value v_2^0 that t_1^0 big-steps to.

The rules of the big-step relations cooperate with each other in various ways. As an illustration, let i in (code-i) be 0. Given a code operation $\langle t^1 \rangle$ at level 0, we first evaluate its operand t^1 at level 1. When a splice operation is encountered, we invoke (splice-1). The operand of the splice operation evaluates to a code operation whose operand is a value. The value is then spliced to the context of t^1 which is surrounded by code brackets.

Not every term has a corresponding rule. (1) There is no (splice-0) because a splice operation is always at some level higher than 0. (2) There is no (ref-0). Level 0 of multi-stage programming corresponds to single-stage programming in which solely evaluating a free variable is disallowed.

Nontermination terms do not big-step. For example, to evaluate $(\lambda x.x\ x)\ (\lambda x.x\ x)$ at level 0, we need to know what $(\lambda x.x\ x)\ (\lambda x.x\ x)$ big-steps to at level 0. Since this circular reasoning never terminates, $(\lambda x.x\ x)\ (\lambda x.x\ x)$ does not big-step at level 0.

In (plus-0), the addition operator in the side condition is different from the addition operator in the bottom of the rule. The addition operation in the side condition happens in the metalanguage, i.e., the language that defines MetaML. For the sake of simplicity, we do not make clear distinction between natural numbers in the language and in the metalanguage.

Examples. To get familiar with the big-step relations, consider the following examples. We use the black triangle ▲ to indicate where an evaluation gets stuck.

Example 27. Evaluate $\langle \sim\langle 1 \rangle \rangle$ at level zero using the big-step relations.

We have:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{1 \Downarrow^1 1}\ \text{(num-i)}}{\langle 1 \rangle \Downarrow^0 \langle 1 \rangle}\ \text{(code-i)}}{\sim\langle 1 \rangle \Downarrow^1 1}\ \text{(splice-1)}}{\langle \sim\langle 1 \rangle \rangle \Downarrow^0 \langle 1 \rangle}\ \text{(code-i)}$$

A code operation big-steps to a code value. A splice operator splices code into its current context.

Example 28. Evaluate $\langle \lambda x.x \rangle$ at level zero using the big-step relations.

We have:

$$\dfrac{\dfrac{\dfrac{}{x \Downarrow^1 x}\ \text{(ref-(i+1))}}{\lambda x.x \Downarrow^1 \lambda x.x}\ \text{(lambda-(i+1))}}{\langle \lambda x.x \rangle \Downarrow^0 \langle \lambda x.x \rangle}\ \text{(code-i)}$$

A code value big-steps to itself.

Example 29. Evaluate $\langle \lambda x.{\sim}\langle x \rangle \rangle$ at level zero using the big-step relations.

We have:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x \Downarrow^1 x}\ \text{(ref-(i+1))}}{\langle x \rangle \Downarrow^0 \langle x \rangle}\ \text{(code-0)}}{{\sim}\langle x \rangle \Downarrow^1 x}\ \text{(splice-1)}}{\lambda x.{\sim}\langle x \rangle \Downarrow^1 \lambda x.x}\ \text{(lambda-(i+1))}}{\langle \lambda x.{\sim}\langle x \rangle \rangle \Downarrow^0 \langle \lambda x.x \rangle}\ \text{(code-i)}$$

At levels higher than 0, we can go inside a lambda abstraction to evaluate its body.

Example 30. Evaluate $\langle \lambda x.{\sim}x \rangle$ at level zero using the big-step relations.

Observe $\langle \lambda x.{\sim}x \rangle \not\Downarrow^0$ because:

$$\dfrac{\dfrac{\dfrac{x \Downarrow^0 \blacktriangle}{{\sim}x \Downarrow^1}\ \text{(splice-1)}}{\lambda x.{\sim}x \Downarrow^1}\ \text{(lambda-(i+1))}}{\langle \lambda x.{\sim}x \rangle \Downarrow^0}\ \text{(code-i)}$$

$\langle \lambda x.{\sim}x \rangle$ is a bad program. It is indeed a term, but is not meant to be written. The evaluation gets stuck at $\blacktriangle$ because to evaluate x at level 0, no rule can apply. The splice operator $\sim$ expects the operand to evaluate to code, but x is stuck.

Example 31. Evaluate $\langle \lambda x.{\sim}(1+1) \rangle$ at level zero using the big-step relations.

Observe $\langle \lambda x.{\sim}(1+1) \rangle \not\Downarrow^0$ because

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{1 \Downarrow^0 1}\ \text{(num-i)}\quad \dfrac{}{1 \Downarrow^0 1}\ \text{(num-i)}}{1+1 \Downarrow^0 2}\ \text{(plus-0)}}{{\sim}(1+1) \Downarrow^1 \blacktriangle}\ \text{(splice-1)}}{\lambda x.{\sim}(1+1) \Downarrow^1}\ \text{(lambda-(i+1))}}{\langle \lambda x.{\sim}(1+1) \rangle \Downarrow^0}\ \text{(code-i)}$$

$\langle \lambda x.{\sim}(1+1) \rangle$ is a bad program. The evaluation gets stuck at $\blacktriangle$ because the rule (splice-1) expects $1+1$ to evaluate to code but 2 is not code. The splice operator $\sim$ expects the operand to evaluate to code. Unlike the previous example, the operand $1+1$ is not stuck but does not evaluate to code.

Programs and Answers. We can define an evaluator that takes a program as its input and provides an answer as its output. We first define the set of programs and the set of answers.

Closed level-0 terms are programs in MetaML

Definition 32 (Programs)**.** Let the set of programs $\textsc{Prgm}_{\text{MetaML}}$ be the set of closed terms at level 0.

$$\textsc{Prgm}_{\text{MetaML}} = \{t \in \textsc{Term}^0 \mid FV(t) = \emptyset\}.$$

Answers are the observational results of evaluating programs. An answer can be the text `function`, the text `code` or a natural number.

Definition 33 (Answers)**.** Let the set of answers $\textsc{Ans}_{\text{MetaML}}$ be the union of the set $\{\texttt{function}, \texttt{code}\}$ and the set of natural numbers.

$$\textsc{Ans}_{\text{MetaML}} = \{\texttt{function}, \texttt{code}\} \cup \mathbb{N}.$$

The set of programs $\textsc{Prgm}_{\text{MetaML}}$ and answers $\textsc{Ans}_{\text{MetaML}}$ are defined for MetaML, not for any particular semantics of MetaML.

Evaluator. We now define an evaluator in terms of the substitutional natural semantics of MetaML. Given a program t, the evaluator applies the big-step relations on t at level 0. If the program big-steps to a natural number, then the evaluator outputs the number. Otherwise, the evaluator indicates the class of value that the program big-steps to, i.e., either `function` or `code`. The evaluator is undefined for programs that get stuck and programs that do not terminate.

Definition 34 (Evaluator based on Substitutional Natural Semantics)**.** Let the evaluator $\mathit{eval}_{\text{MetaML:SubNat}}$ be a partial function from the set of programs $\textsc{Prgm}_{\text{MetaML}}$ to the set of answers $\textsc{Ans}_{\text{MetaML}}$.

$$\mathit{eval}_{\text{MetaML:SubNat}} : \textsc{Prgm}_{\text{MetaML}} \rightharpoonup \textsc{Ans}_{\text{MetaML}}$$

$$\mathit{eval}_{\text{MetaML:SubNat}}(t) = \begin{cases} \texttt{function} & \text{if } t \Downarrow^0 \lambda x.t'^0 \\ \texttt{code} & \text{if } t \Downarrow^0 \langle v^1 \rangle \\ n & \text{if } t \Downarrow^0 n \end{cases}$$

This above evaluator is defined in terms of the substitutional natural semantics. The subscript "$_{\text{MetaML:SubNat}}$" in $\mathit{eval}_{\text{MetaML:SubNat}}$ denotes the substitutional natural semantics of MetaML.

We demonstrate how the evaluator works by evaluating the puzzle program below that was originally presented in [Tah99a].

$$!\langle \lambda a. \sim ((\lambda x.\langle x \rangle)\,(\lambda x.\langle a \rangle))\,0 \rangle\, 5$$

Example 35. We have

$$\mathit{eval}_{\text{MetaML:SubNat}}(!\langle \lambda a. \sim ((\lambda x.\langle x \rangle)\,(\lambda x.\langle a \rangle))\,0 \rangle\, 5) = \texttt{code}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\lambda x.\langle x\rangle \Downarrow^0 \lambda x.\langle x\rangle}\quad \dfrac{}{\lambda x.\langle a\rangle \Downarrow^0 \lambda x.\langle a\rangle}\quad \dfrac{\dfrac{\dfrac{\dfrac{}{a \Downarrow^2 a}}{\langle a\rangle \Downarrow^1 \langle a\rangle}}{\lambda x.\langle a\rangle \Downarrow^1 \lambda x.\langle a\rangle}}{\langle \lambda x.\langle a\rangle\rangle \Downarrow^0 \langle \lambda x.\langle a\rangle\rangle}}{(\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle) \Downarrow^0 \langle \lambda x.\langle a\rangle\rangle}}{\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle)) \Downarrow^1 \lambda x.\langle a\rangle}\quad \dfrac{}{0 \Downarrow^1 0}}{\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0 \Downarrow^1 (\lambda x.\langle a\rangle)\,0}}{\lambda a.\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0 \Downarrow^1 \lambda a.((\lambda x.\langle a\rangle)\,0)}}{\langle \lambda a.\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0\rangle \Downarrow^0 \langle \lambda a.((\lambda x.\langle a\rangle)\,0)\rangle}\quad \dfrac{}{\lambda a.((\lambda x.\langle a\rangle)\,0) \Downarrow^0 \lambda a.((\lambda x.\langle a\rangle)\,0)}\quad \dfrac{}{5 \Downarrow^0 5}\quad \dfrac{}{\lambda x.\langle 5\rangle \Downarrow^0 \lambda x.\langle 5\rangle}\quad \dfrac{}{0 \Downarrow^0 0}\quad \dfrac{\dfrac{}{5 \Downarrow^1 5}}{\langle 5\rangle \Downarrow^0 \langle 5\rangle}}{!\langle \lambda a.\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0\rangle \Downarrow^0 \lambda a.((\lambda x.\langle a\rangle)\,0)}\quad \dfrac{}{(\lambda x.\langle 5\rangle)\,0 \Downarrow^0 \langle 5\rangle}}{!\langle \lambda a.\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0\rangle\,5 \Downarrow^0 \langle 5\rangle}$$

Figure 2.1: Evaluation of $!\langle \lambda a.\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0\rangle\,5$ in Substitutional Natural Semantics of MetaML.

because

$$!\langle \lambda a. \sim ((\lambda x.\langle x \rangle)\,(\lambda x.\langle a \rangle))\,0 \rangle\; 5 \Downarrow^0 \langle 5 \rangle$$

as shown in Figure 2.1 and $\langle 5 \rangle \in \text{VALUE}^0$.

2.2.3 Substitutional Structural Operational Semantics

Previously, we have studied the substitutional natural semantics of MetaML. The main semantics refinement problem is to define an environmental abstract machine for MetaML, which we call the MEK machine. The MEK machine is a small-step operational semantics. As the first step in the process of deriving the MEK machine, we refine the substitutional natural semantics of MetaML to a substitutional structural operational semantics.

Natural semantics relates a term to its final result of computation. Structural operational semantics relates a term to its next small step of computation on the way to its final result. One advantage of the structural operational semantics is that it allows reasoning about programs that do not terminate such as an operating system. In contrast, natural semantics is only defined for programs that can produce a final result.

We lay out the substitutional structural operational semantics through a family of level-indexed single-step relations and a family of level-indexed multi-step relations.

Definition 36 (Level-indexed Single-step Relations)**.** For any $i \in \mathbb{N}$, let the level-indexed single-step relation $\longrightarrow^i$ be a binary relation between the set of terms at level i and the set of terms at level i.

$$\longrightarrow^i \subseteq \text{TERM}^i \times \text{TERM}^i$$

$$\textit{No (lambda-0)} \qquad \frac{t_1^{i+1} \longrightarrow^{i+1} t_2^{i+1}}{\lambda x.t_1^{i+1} \longrightarrow^{i+1} \lambda x.t_2^{i+1}}\ \text{(lambda-(i+1))}$$

$$\frac{t_{11}^i \longrightarrow^i t_{12}^i}{t_{11}^i\; t_2^i \longrightarrow^i t_{12}^i\; t_2^i}\ \text{(appL-i)} \qquad \frac{t_{21}^i \longrightarrow^i t_{22}^i}{v_1^i\; t_{21}^i \longrightarrow^i v_1^i\; t_{22}^i}\ \text{(appR-i)} \qquad \frac{}{(\lambda x.t^0)\; v^0 \longrightarrow^0 t^0[v^0/x]}\ \text{(app-0)}$$

$$\frac{}{!\langle v^1 \rangle \longrightarrow^0 v^1}\ \text{(run-0)} \qquad \frac{t_1^i \longrightarrow^i t_2^i}{!t_1^i \longrightarrow^i !t_2^i}\ \text{(run-i)}$$

$$\frac{t_1^{i+1} \longrightarrow^{i+1} t_2^{i+1}}{\langle t_1^{i+1} \rangle \longrightarrow^i \langle t_2^{i+1} \rangle}\ \text{(code-i)}$$

$$\textit{No (splice-0)} \qquad \frac{}{\sim\langle v^1 \rangle \longrightarrow^1 v^1}\ \text{(splice-1)} \qquad \frac{t_1^i \longrightarrow^i t_2^i}{\sim t_1^i \longrightarrow^{i+1} \sim t_2^i}\ \text{(splice-(i+1))}$$

$$\textit{No (ref-i)}$$

$$\textit{No (num-i)}$$

$$\frac{t_{11}^i \longrightarrow^i t_{12}^i}{t_{11}^i + t_2^i \longrightarrow^i t_{12}^i + t_2^i}\ \text{(plusL-i)} \qquad \frac{t_{21}^i \longrightarrow^i t_{22}^i}{v_1^i + t_{21}^i \longrightarrow^i v_1^i + t_{22}^i}\ \text{(plusR-i)}$$

$$\frac{}{n_1 + n_2 \longrightarrow^0 n}\ \text{where } n = n_1 + n_2\ \text{(plus-0)}$$

The single-step relation $t_1^i \longrightarrow^i t_2^i$ reads as "t_1 single-steps to t_2 at level i".

Recall that a term can be a variable x, an application $t_1\ t_2$, a lambda abstraction $\lambda x.t$, a code operation $\langle t \rangle$, a splice operation $\sim t$, a run operation $!t$, a natural number n and an addition operation $t_1 + t_2$. We explain how an arbitrary term gets evaluated by the above relations.

Given a variable x, we get stuck because there is no (ref-i) rule.

Given an application $t_1\ t_2$, we may evaluate it using the (appL-i) rule, the (appR-i) rule or the (app-0) rule depending on the current level and whether its operator/operand is a value. We first repeatedly apply the (appL-i) rule to evaluate the operator t_1 and finally get a value v_1. Then we use repeatedly apply the (appR-i) rule to evaluate the operand t_2 and finally get a value v_2. If we have been evaluating at level 0, we expect v_1 to be a lambda abstraction and we then perform the application $v_1\ v_2$ using the (app-0) rule. If we have been evaluating at level $i+1$, the resulting application $v_1\ v_2$ is a value at that level.

Given a lambda abstraction $\lambda x.t$, we may evaluate it using the (lambda-(i+1)) rule depending on its level and whether the body t is a value. (1) If the lambda abstraction is at level 0, then we get stuck because there is no (lambda-0) rule. (2) If the lambda abstraction is at level $i+1$, we repeatedly apply the (lambda-(i+1)) rule to evaluate its body t at level $i+1$ and finally get a value v. The resulting application $\lambda x.v$ is a value at level $i+1$.

Given a code operation $\langle t \rangle$, we may evaluate it using the (code-i) rule depending on whether its operand is a value. We repeatedly apply the (code-i) rule to evaluate its operand t and finally get a value v. The resulting code operation $\langle v \rangle$ is a value .

Given a splice operation $\sim t$, we may evaluate it using the (splice-1) rule or the (splice-(i+1)) rule depending on the current level and whether its operand is a value. (1) If the splice operation is at level 0, we get stuck because there is no (splice-0) rule. (2) If the splice operation is at level $i+1$, we repeatedly apply the (splice-(i+1)) rule to evaluate its operand t and finally get a value v. (2.1) If we have been evaluating at level 1, we expect v to be code operation and we apply (splice-1) to merge the code into the context. (2.2) If we have been evaluating at level $i+2$, the resulting splice operation $\sim v$ is a value at that level.

Given a run operation $!t$, we may evaluate it using the (run-0) rule or the (run-i) rule depending on the current level and whether its operand is a value. We first repeatedly apply the (run-i) rule to evaluate its operand t and finally get a value v. (1) If we have been evaluating at level 0, we expect v to be code operation and we apply (run-0) to execute the code. (2) If we have been evaluating at level $i+1$, the resulting code operation $!v$ is a value at that level.

Given a natural number n, we get stuck because there is no (num-i) rule.

Given an addition operation $t_1 + t_2$, we may evaluate it using the (plusL-i) rule, the (plusR-i) rule or the (plus-0) rule depending on the current level and whether its first or second operand is a value. We first

repeatedly apply the (appL-i) rule to evaluate the first operand t_1 and finally get a value v_1. Then we use repeatedly apply the (plusR-i) rule to evaluate the second operand t_2 and finally get a value v_2. If we have been evaluating at level 0, we expect v_1 and v_2 to be two natural numbers and we perform the addition $v_1 + v_2$ using the (plus-0) rule. If we have been evaluating at level $i+1$, the resulting addition $v_1 + v_2$ is a value at that level.

Observations. The rules of the single-step relations can be classified into two categories. (1) The rules (app-0), (run-0), (splice-1) and (plus-0) are *reduction* rules which perform a real step of computation. (2) The other rules are *structural* rules which define how to evaluate a term with respect to its sub-terms. There are no value rules in the single-step relations as opposed to the big-step relations. This is because stepping always does work and there is no work left to do for values.

The single-step relations tell that the language is call-by-value. The (app-0) rule restricts the operand of the application to be a value.

The big-step relations do not specify which argument to a two-argument operation, like function application or addition, must be evaluated first. The single-step relations, on the other hand, force evaluation to proceed from left to right, as indicated by (appL-i), (appR-i), (plusL-i) and (plusR-i). We say that the single-step relations are tailored to leftmost reduction.

The single-step relations have fewer rules than the big-step relations. (1) There are no (lambda-0), (ref-(i+1)) and (num-i) because values do not reduce in the single-step relations but evaluate to themselves in the big-step relations. (2) There is no (ref-0), analogous to the fact that evaluating a free variable in a single-stage programming language is disallowed. (3) There is no (splice-0) because a splice operation is always at some level higher than 0.

Intuitively, a level-indexed single-step relation $\longrightarrow^i$ defines a single step of computation at level i. To represent multiple (zero or more) steps of computation at level i, we define the level-indexed multi-step relation $\longrightarrow^{*i}$.

Definition 37 (Level-indexed Multi-step Relation)**.** Define the level-indexed multi-step relation $\longrightarrow^{i*}$ to be the reflexive-transitive closure of the level-indexed single-step relation $\longrightarrow^i$.

$$\longrightarrow^{i*} \subseteq \textsc{Term}^i \times \textsc{Term}^i$$

$$\frac{}{t_1^i \longrightarrow^{i*} t_2^i} \text{ where } t_1^i \longrightarrow^i t_2^i \text{ (step)} \qquad \frac{}{t^i \longrightarrow^{i*} t^i} \text{ (refl)} \qquad \frac{t_1^i \longrightarrow^{i*} t_2^i \quad t_2^i \longrightarrow^{i*} t_3^i}{t_1^i \longrightarrow^{i*} t_3^i} \text{ (trans)}$$

The multi-step relation $t_1^i \longrightarrow^{i*} t_2$ reads as "t_1 multi-steps to t_2 at level i". The (step) rule implies that the multi-step relation respects the single-step relation with the same level index. The (refl) rule implies that the multi-step relation is reflexive. The (trans) rule implies that the multi-step relation is transitive.

Examples. To get familiar with the substitutional structural operational semantics, consider the following examples. These examples are the same as the ones in we presented for substitutional natural semantics.

Example 38. Evaluate $\langle\sim\langle 1\rangle\rangle$ at level 0 using the substitutional structural operational semantics.

We have:

$$\dfrac{\dfrac{}{\sim\langle 1\rangle \longrightarrow^1 1}\ \text{(splice-1)}}{\langle\sim\langle 1\rangle\rangle \longrightarrow^0 \langle 1\rangle}\ \text{(code-i)}$$

Observe $\langle 1\rangle \not\longrightarrow^0$ because:

$$\dfrac{1 \longrightarrow^1 \blacktriangle}{\langle 1\rangle \longrightarrow^0}\ \text{(code-i)}$$

The evaluation gets stuck at $\blacktriangle$ because there is no (num-i) rule. Furthermore, we have $\langle 1\rangle \in \text{VALUE}^0$. By the (step) rule of the multi-step relation, $\langle\sim\langle 1\rangle\rangle \longrightarrow^{0*} \langle 1\rangle$.

Example 39. Evaluate $\langle\lambda x.x\rangle$ at level 0 using the substitutional structural operational semantics.

Observe $\langle\lambda x.x\rangle \not\longrightarrow^0$ because

$$\dfrac{\dfrac{x \longrightarrow^1 \blacktriangle}{\lambda x.x \longrightarrow^1}\ \text{(lambda-(i+1))}}{\langle\lambda x.x\rangle \longrightarrow^0}\ \text{(code-i)}$$

The evaluation gets stuck at $\blacktriangle$ because there is no (ref-i) rule. Furthermore, we have $\langle\lambda x.x\rangle \in \text{VALUE}^0$. By the (refl) rule of the multi-step relation, $\langle\lambda x.x\rangle \longrightarrow^0 \langle\lambda x.x\rangle$.

Example 40. Evaluate $\langle\lambda x.\sim\langle x\rangle\rangle$ at level 0 using the substitutional structural operational semantics.

We have:

$$\dfrac{\dfrac{\dfrac{}{\sim\langle x\rangle \longrightarrow^1 x}\ \text{(splice-1)}}{\lambda x.\sim\langle x\rangle \longrightarrow^1 \lambda x.x}\ \text{(lambda-(i+1))}}{\langle\lambda x.\sim\langle x\rangle\rangle \longrightarrow^0 \langle\lambda x.x\rangle}\ \text{(code-i)}$$

Observe $\langle\lambda x.x\rangle \not\longrightarrow^0$ because

$$\dfrac{\dfrac{x \longrightarrow^1 \blacktriangle}{\lambda x.x \longrightarrow^1}\ \text{(lambda-(i+1))}}{\langle\lambda x.x\rangle \longrightarrow^0}\ \text{(code-i)}$$

The evaluation gets stuck at because there is no (ref-i) rule. Furthermore, we have $\langle\lambda x.x\rangle \in \text{VALUE}^0$. By the (step) rule of the multi-step relation, $\langle\lambda x.\sim\langle x\rangle\rangle \longrightarrow^{0*} \langle\lambda x.x\rangle$.

Example 41. Evaluate $\langle\lambda x.\sim x\rangle$ at level 0 using the substitutional structural operational semantics.

Observe $\langle\lambda x.\sim x\rangle \not\longrightarrow^0$ because:

$$\dfrac{\dfrac{\dfrac{x \longrightarrow^0 \blacktriangle}{\sim x \longrightarrow^1}\ \text{(splice-1)}}{\lambda x.\sim x \longrightarrow^1}\ \text{(lambda-(i+1))}}{\langle\lambda x.\sim x\rangle \longrightarrow^0}\ \text{(code-i)}$$

The evaluation gets stuck at ▲ because there is no (ref-i) rule. Furthermore, we have $\langle \lambda x.{\sim}x \rangle \notin \text{VALUE}^0$ and $\langle \lambda x.{\sim}x \rangle$ is a stuck term.

By the (refl) rule of the multi-step relation, $\langle \lambda x.{\sim}x \rangle \longrightarrow^{0*} \langle \lambda x.{\sim}x \rangle$.

Example 42. Evaluate $\langle \lambda x.{\sim}(1+1) \rangle$ at level 0 using the substitutional structural operational semantics.

We have:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{1+1 \longrightarrow^0 2}\text{(plus-0)}}{{\sim}(1+1) \longrightarrow^1 {\sim}2}\text{(splice-1)}}{\lambda x.{\sim}(1+1) \longrightarrow^1 \lambda x.{\sim}2}\text{(lambda-(i+1))}}{\langle \lambda x.{\sim}(1+1) \rangle \longrightarrow^0 \langle \lambda x.{\sim}2 \rangle}\text{(code-i)}$$

Observe $\langle \lambda x.{\sim}2 \rangle \not\longrightarrow^0$ because:

$$\dfrac{\dfrac{\dfrac{2 \longrightarrow^0 \blacktriangle}{{\sim}2 \longrightarrow^1}\text{(splice-(i+1))}}{\lambda x.{\sim}2 \longrightarrow^1}\text{(lambda-(i+1))}}{\langle \lambda x.{\sim}2 \rangle \longrightarrow^0}\text{(code-i)}$$

The evaluation gets stuck at ▲ because there is no (num-i) rule. Furthermore, we have $\langle \lambda x.{\sim}2 \rangle \notin \text{VALUE}^0$ and $\langle \lambda x.{\sim}2 \rangle$ is a stuck term.

By the (step) rule of the multi-step relation, $\langle \lambda x.{\sim}(1+1) \rangle \longrightarrow^{0*} \langle \lambda x.{\sim}2 \rangle$.

Evaluator. We now define an evaluator in terms of the substitutional natural semantics of MetaML. Given a program t, the evaluator applies the multi-step relations on t at level 0. If the program multi-steps to a natural number, then the evaluator outputs the number. Otherwise, the evaluator indicates the class of value that the program multi-steps to, i.e., either `function` or `code`. The evaluator is undefined if the program gets stuck or does not terminate.

Definition 43 (Evaluator based on Substitutional Structural Operational Semantics)**.** Define the evaluator $eval_{\text{MetaML:SubSOS}}$ to be a partial function from the set of programs $\text{PRGM}_{\text{MetaML}}$ to the set of answers $\text{ANS}_{\text{MetaML}}$.

$$eval_{\text{MetaML:SubSOS}} : \text{PRGM}_{\text{MetaML}} \rightharpoonup \text{ANS}_{\text{MetaML}}$$

$$eval_{\text{MetaML:SubSOS}}(t) = \begin{cases} \texttt{function} & \text{if } t \longrightarrow^{0*} \lambda x.t'^{0} \\ \texttt{code} & \text{if } t \longrightarrow^{0*} \langle v^1 \rangle \\ n & \text{if } t \longrightarrow^{0*} n \end{cases}$$

This evaluator is defined in terms of the substitutional structural operational semantics. The subscript "$_{\text{MetaML:SubSOS}}$" in $eval_{\text{MetaML:SubSOS}}$ denotes the substitutional structural operational semantics of MetaML.

We demonstrate how the evaluator works by evaluating the same puzzle program as Example 35.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{(\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle) \longrightarrow^0 \langle \lambda x.\langle a\rangle\rangle}\text{ where } \langle x\rangle \in \text{TERM}^0 \text{ and } \lambda x.\langle a\rangle \in \text{VALUE}^0}{\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle)) \longrightarrow^1 \sim\langle \lambda x.\langle a\rangle\rangle}}{\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0 \longrightarrow^1 \sim\langle \lambda x.\langle a\rangle\rangle\,0}}{\lambda a.\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0 \longrightarrow^1 \lambda a.\sim\langle \lambda x.\langle a\rangle\rangle\,0}}{\langle\lambda a.\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0\rangle \longrightarrow^0 \langle\lambda a.\sim\langle \lambda x.\langle a\rangle\rangle\,0\rangle}}{!\langle\lambda a.\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0\rangle \longrightarrow^0 !\langle\lambda a.\sim\langle \lambda x.\langle a\rangle\rangle\,0\rangle}}{!\langle\lambda a.\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0\rangle\,5 \longrightarrow^0 !\langle\lambda a.\sim\langle \lambda x.\langle a\rangle\rangle\,0\rangle\,5}$$

(a) Derivation of $!\langle\lambda a.\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0\rangle\,5 \longrightarrow^0 !\langle\lambda a.\sim\langle\lambda x.\langle a\rangle\rangle\,0\rangle\,5$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\sim\langle\lambda x.\langle a\rangle\rangle \longrightarrow^1 \lambda x.\langle a\rangle}\text{ where } \lambda x.\langle a\rangle \in \text{VALUE}^1}{\sim\langle\lambda x.\langle a\rangle\rangle\,0 \longrightarrow^1 (\lambda x.\langle a\rangle)\,0}}{\lambda a.\sim\langle\lambda x.\langle a\rangle\rangle\,0 \longrightarrow^1 \lambda a.(\lambda x.\langle a\rangle)\,0}}{\langle\lambda a.\sim\langle\lambda x.\langle a\rangle\,0\rangle \longrightarrow^0 \langle\lambda a.(\lambda x.\langle a\rangle)\,0\rangle}}{!\langle\lambda a.\sim\langle\lambda x.\langle a\rangle\,0\rangle \longrightarrow^0 !\langle\lambda a.(\lambda x.\langle a\rangle)\,0\rangle}}{!\langle\lambda a.\sim\langle\lambda x.\langle a\rangle\rangle\,0\rangle\,5 \longrightarrow^0 !\langle\lambda a.(\lambda x.\langle a\rangle)\,0\rangle\,5}$$

(b) Derivation of $!\langle\lambda a.\sim\langle\lambda x.\langle a\rangle\rangle\,0\rangle\,5 \longrightarrow^0 !\langle\lambda a.(\lambda x.\langle a\rangle)\,0\rangle\,5$.

$$\dfrac{\dfrac{\dfrac{}{!\langle\lambda a.(\lambda x.\langle a\rangle)\,0\rangle \longrightarrow^0 \lambda a.(\lambda x.\langle a\rangle)\,0}\text{ where } \lambda a.(\lambda x.\langle a\rangle)\,0 \in \text{VALUE}^1}{!\langle\lambda a.(\lambda x.\langle a\rangle)\,0\rangle\,5 \longrightarrow^0 (\lambda a.(\lambda x.\langle a\rangle)\,0)\,5}$$

(c) Derivation of $!\langle\lambda a.(\lambda x.\langle a\rangle)\,0\rangle\,5 \longrightarrow^0 (\lambda a.(\lambda x.\langle a\rangle)\,0)\,5$.

$$\dfrac{\dfrac{\dfrac{}{(\lambda x.\langle a\rangle)\,0 \longrightarrow^0 \langle a\rangle}\text{ where } \langle a\rangle \in \text{TERM}^0 \text{ and } 0 \in \text{VALUE}^0}{\lambda a.(\lambda x.\langle a\rangle)\,0 \longrightarrow^0 \lambda a.\langle a\rangle}}{(\lambda a.(\lambda x.\langle a\rangle)\,0)\,5 \longrightarrow^0 (\lambda a.\langle a\rangle)\,5}$$

(d) Derivation of $(\lambda a.(\lambda x.\langle a\rangle)\,0)\,5 \longrightarrow^0 (\lambda a.\langle a\rangle)\,5$.

$$\dfrac{}{(\lambda a.\langle a\rangle)\,5 \longrightarrow^0 \langle 5\rangle}\text{ where } \langle a\rangle \in \text{TERM}^0 \text{ and } 5 \in \text{VALUE}^0$$

(e) Derivation of $(\lambda a.\langle a\rangle)\,5 \longrightarrow^0 \langle 5\rangle$.

Figure 2.2: Evaluation of $!\langle\lambda a.\sim((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0\rangle\,5$ in Substitutional Structural Operational Semantics of MetaML.

Example 44. We have

$$eval_{\text{MetaML:SubSOS}}(!\langle \lambda a. \sim ((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0\rangle\,5) = \texttt{code}$$

because

$$!\langle \lambda a. \sim ((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0\rangle\,5 \longrightarrow^{0*} \langle 5\rangle$$

as shown in Figure 2.2 and $\langle 5\rangle \in \text{VALUE}^0$.

Examples 35 and 44 show that two evaluators we have defined so far agree on the evaluation of the puzzle program $!\langle \lambda a. \sim ((\lambda x.\langle x\rangle)\,(\lambda x.\langle a\rangle))\,0\rangle\,5$. In fact, these two evaluators agree on all programs.

Theorem 45 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{MetaML}}$, $eval_{\text{MetaML:SubNat}}(t)$ *is Kleene equal to* $eval_{\text{MetaML:SubSOS}}(t)$.

The above theorem uses Kleene Equality introduced in [Kle52]: for any expressions A and B, A is *Kleene equal* to B if and only if (1) both A and B are defined and are equal or (2) both A and B are undefined. We prove the theorem in the appendices.

2.3 Chapter Summary

We first introduced three staging annotations and informally discussed how a multi-stage program evaluates. Then we studied the pre-existing substitutional natural semantics of MetaML and derived a substitutional structural operational semantics for MetaML. We defined evaluators for both semantics and demonstrated their equivalence.

Chapter 3

Refining Semantics for ISWIM: Developing the CEK Machine

Following the first dimension of simplifying the main semantics refinement problem, we study how to stepwise develop an environmental abstract machine for the single-stage language ISWIM rather than the multi-stage language MetaML. The problem is restated as follows.

> Can we refine the substitutional structural operational semantics of ISWIM to a corresponding environmental abstract machine, which is known as the CEK machine, and demonstrate their equivalence?

We tackle this problem progressively in several manageable steps through several intermediate semantics.

3.1 ISWIM

ISWIM, whose acronym stands for "if you see what I mean", was originally developed by [Lan66]. It was introduced to understand and design the whole landscape of programming languages [Lan64]. It has influenced the development of functional programming languages such as ML and Haskell.

We consider a variant of ISWIM that has natural numbers and addition. We sometimes call what is presented in this section *Substitutional* ISWIM in order to differentiate it from the subsequent dialects, i.e., Explicit ISWIM, Suspended ISWIM and Environmental ISWIM.

3.1.1 Syntax

We first define the basic syntax of the language: terms, values and denotable terms. Then we define the free variable function, the substitution function and the alpha equivalence relation. We finally present the commutativity of substitutions.

3.1.1.1 Terms, Values and Denotable Terms

We start with two sets: the set of variables, VAR, and the set of natural numbers, $\mathbb{N}$.

Definition 46 (Terms, Values and Denotable Terms)**.** Let (1) TERM be the set of terms, (2) VALUE be the set of values, and (3) DENOTABLE be the set of denotable terms.

$x \in \text{VAR}$, $n \in \mathbb{N}$, $t \in \text{TERM}$, $v \in \text{VALUE}$, $w \in \text{DENOTABLE}$

$$\begin{array}{rcl} t & ::= & x \mid t\,t \mid \lambda x.t \mid n \mid t+t \\ v & ::= & \lambda x.t \mid n \\ w & ::= & x \mid v \end{array}$$

ISWIM can be viewed as a single-stage restricted form of MetaML. Given the syntax of MetaML, if we remove all terms that contain any staging annotation or is not at level 0, we get ISWIM.

3.1.1.2 Free Variable Function

We define the free variable function as follows.

Definition 47 (Free Variable Function)**.** Define the free variable function FV to be a total function from the set of terms to the power set of variables.

$$FV : \text{TERM} \longrightarrow \mathscr{P}(\text{VAR})$$

$$\begin{array}{rcll} FV(x) & = & x & (1) \\ FV(t_1\,t_2) & = & FV(t_1) \cup FV(t_2) & (2) \\ FV(\lambda x.t) & = & FV(t) \backslash \{x\} & (3) \\ FV(n) & = & \emptyset & (4) \\ FV(t_1+t_2) & = & FV(t_1) \cup FV(t_2) & (5) \end{array}$$

The free variable function for ISWIM is the same as for MetaML but restricted to the single-stage part.

Definition 48 (Closed Terms)**.** A term t is *closed* if and only if $FV(t) = \emptyset$.

3.1.1.3 Substitution Function

We define the substitution function as follows.

Definition 49 (Substitution Function)**.** Define the substitution function $\cdot[\cdot/\cdot]$ to be a total function from the 3-tuple of the set of terms, the set of denotable terms and the set of variables, to the set of terms.

$$\cdot[\cdot/\cdot] : (\text{TERM} \times \text{DENOTABLE} \times \text{VAR}) \longrightarrow \text{TERM}$$

$$\begin{array}{rcll} x_1[w/x_2] & = & w \text{ where } x_1 \equiv x_2 & (1) \\ x_1[w/x_2] & = & x_1 \text{ where } x_1 \not\equiv x_2 & (2) \\ (t_1\,t_2)[w/x] & = & (t_1[w/x])\,(t_2[w/x]) & (3) \\ (\lambda x_1.t_0)[w/x_2] & = & \lambda x_3.t_0[x_3/x_1][w/x_2] & \\ & & \text{where } x_3 \notin FV(\lambda x_1.t_0) \cup FV(w) \cup \{x_2\} & (4) \\ n[w/x] & = & n & (5) \\ (t_1+t_2)[w/x] & = & (t_1[w/x]) + (t_2[w/x]) & (6) \end{array}$$

The substitution function for ISWIM is the same as for MetaML but restricted to the single-stage part.

3.1.1.4 Alpha Equivalence Relation

Alpha equivalence reflects that the particular choice of the bound variable in a lambda abstraction does not matter. Two terms are α-equivalent if and only if they are identical except for renaming bound variables.

Definition 50 (Alpha Equivalence Relation)**.** Define the alpha equivalence relation $\sim_\alpha$ to be a binary relation between the set of terms and the set of terms.

$$\sim_\alpha \subseteq \textsc{Term} \times \textsc{Term}$$

$$\frac{}{x \sim_\alpha x}\ (\text{var}) \qquad \frac{t_{11} \sim_\alpha t_{21} \quad t_{12} \sim_\alpha t_{22}}{(t_{11}\ t_{12}) \sim_\alpha (t_{21}\ t_{22})}\ (\text{app})$$

$$\frac{t_1[x_3/x_1] \sim_\alpha t_2[x_3/x_2]}{(\lambda x_1.t_1) \sim_\alpha (\lambda x_2.t_2)}\ \text{where } x_3 \notin FV(t_1) \cup FV(t_2)\ (\text{lam})$$

$$\frac{t_{11} \sim_\alpha t_{21} \quad t_{12} \sim_\alpha t_{22}}{(t_{11} + t_{12}) \sim_\alpha (t_{21} + t_{22})}\ (\text{plus}) \qquad \frac{}{n \sim_\alpha n}\ (\text{num})$$

We did not introduce the alpha equivalence relation for MetaML in the previous chapter. This is because it is usually not difficult to demonstrate equivalence of two semantics of the exact same language. However, in this chapter and later chapters, we need to prove equivalence of semantics of different dialects of ISWIM or MetaML. It is not uncommon that two semantics evaluate the same term to two syntactically different values that have the same meaning semantically. Since our evaluators only concern observational results, such values should be equated. For example, $\lambda x.x$ and $\lambda y.y$ represent the same lambda abstraction. They are related by the alpha equivalence relation. Furthermore, the alpha equivalence relation makes our proofs easier. In many cases, we may replace a term by its alpha equivalent term in our proof at our convenience.

Example 51. We have $((\lambda x.x\ x)\ (\lambda x.x)) \sim_\alpha ((\lambda y.y\ y)\ (\lambda z.z))$. The left-hand side and right-hand side of the relation represent the lambda abstractions that only differ naming lambda bound variables.

3.1.1.5 Commutativity of Substitutions

We make several observations with respect to the commutativity of substitutions, which are useful in justifying the design of the succeeding semantics. (1) After substituting a variable with a denotable term that does not contain it, further substitutions for the same variable have no meaningful effect, so they can be dropped. (2) Two non-clashing substitutions can commute with one another. (3) More generally, two substitutions that arise in practice during an evaluation commute which involves more work to ensure no clash happens.

Proposition 52. *If $x \notin FV(w_1)$, then $t[w_1/x][w_2/x] \sim_\alpha t[w_1/x]$.*

Proposition 53. *If $x_1 \not\equiv x_2$, $x_1 \notin FV(w_2)$ and $x_2 \notin FV(w_1)$, then $t[w_1/x_1][w_2/x_2] \sim_\alpha t[w_2/x_2][w_1/x_1]$.*

Proposition 54. *$t[w_1/x_1][w_2/x_2] \sim_\alpha t[x_3/x_1][w_2/x_2][w_1[w_2/x_2]/x_3]$ where $x_3 \notin FV(\lambda x_1.t_0) \cup FV(w) \cup \{x_2\}$.*

3.1.2 Substitutional Structural Operational Semantics

We lay out the substitutional structural operational semantics through the single-step relation $\longrightarrow$ and the multi-step relation $\longrightarrow^*$.

Definition 55 (Single-step Relation)**.** Let the single-step relation $\longrightarrow$ be a binary relation between the set of terms and the set of terms.

$$\longrightarrow \subseteq \textsc{Term} \times \textsc{Term}$$

$$\frac{t_{11} \longrightarrow t_{12}}{t_{11}\ t_2 \longrightarrow t_{12}\ t_2}\ (\text{appL}) \qquad \frac{t_{21} \longrightarrow t_{22}}{v_1\ t_{21} \longrightarrow v_1\ t_{22}}\ (\text{appR}) \qquad \frac{}{(\lambda x.t)\ v \longrightarrow t[v/x]}\ (\text{app})$$

$$\frac{t_{11} \longrightarrow t_{12}}{t_{11}+t_2 \longrightarrow t_{12}+t_2}\ (\text{plusL}) \qquad \frac{t_{21} \longrightarrow t_{22}}{v_1+t_{21} \longrightarrow v_1+t_{22}}\ (\text{plusR}) \qquad \frac{}{n_1+n_2 \longrightarrow n}\ \text{where } n = n_1+n_2\ (\text{plus})$$

The single-step relation $t_1 \longrightarrow t_2$ reads as "t_1 single-steps to t_2".

Definition 56 (Multi-step Relation)**.** Let the multi-step relation $\longrightarrow^*$ be the reflexive-transitive closure of the single-step relation $\longrightarrow$.

$$\longrightarrow^* \subseteq \textsc{Term} \times \textsc{Term}$$

$$\frac{}{t_1 \longrightarrow^* t_2}\ \text{where } t_1 \longrightarrow t_2\ (\text{step}) \qquad \frac{}{t \longrightarrow^* t}\ (\text{refl}) \qquad \frac{t_1 \longrightarrow^* t_2 \quad t_2 \longrightarrow^* t_3}{t_1 \longrightarrow^* t_3}\ (\text{trans})$$

The multi-step relation $t_1 \longrightarrow^* t_2$ reads as "t_1 multi-steps to t_2".

The single-step relation and multi-step relation for the substitutional structural operational semantics of ISWIM are the same as for the substitutional structural operational semantics of MetaML but restricted to the single-stage part.

Example 57. Consider $((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4$ where $x_1 \not\equiv x_2$.

By the substitutional structural operational semantics of ISWIM, we have:

$$\begin{array}{lll}
 & ((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4 & (1)\\
\longrightarrow & (\lambda x_2.x_1)[7/x_1]\ 4 & (2)\\
= & (\lambda x_2.x_1[x_2/x_2][7/x_1])\ 4 \quad \text{where } x_2 \not\equiv x_1 & (3)\\
= & (\lambda x_2.x_1[7/x_1])\ 4 & (4)\\
= & (\lambda x_2.7)\ 4 & (5)\\
\longrightarrow & 7[4/x_2] & (6)\\
= & 7 & (7)
\end{array}$$

The first single-step is indeed from (1) to (5), including applying the substitution $[7/x_1]$ on the lambda abstraction $\lambda x_2.x_1$. The substitution is performed in the meta-language and does not count as any additional reduction step.

Properties. We observe the following properties that are useful in proving semantics equivalence. (1) The single-step relation preserves alpha equivalence. During an evaluation, we may conveniently replace a term by its alpha equivalent term without changing the observational result of the evaluation. (2) The multi-step relation preserves the closedness of a term. Evaluating a program never produces the error of evaluating a free variable. This ensures that the behaviour of a program does not depend on the outside world.

Proposition 58. *If* $t_{a_1} \sim_\alpha t_{b_1}$ *and* $t_{a_1} \longrightarrow t_{a_2}$, *then* $t_{b_1} \longrightarrow t_{b_2}$ *and* $t_{a_2} \sim_\alpha t_{b_2}$.

Proposition 59. *If* $FV(t_1) = \emptyset$ *and* $t_1 \longrightarrow^* t_2$, *then* $FV(t_2) = \emptyset$.

Programs and Answers. To define an evaluator for ISWIM, we first programs and answers.

Closed terms are programs in ISWIM.

Definition 60 (Programs)**.** Let the set of programs $\textsc{Prgm}_{\text{IWSIM}}$ be the set of closed terms.

$$\textsc{Prgm}_{\text{ISWIM}} = \{t \in \textsc{Term} \mid FV(t) = \emptyset\}.$$

An answers is the text `function` or a natural number.

Definition 61 (Answers)**.** Let the set of answers $\textsc{Ans}_{\text{ISWIM}}$ be the union of the set $\{\texttt{function}\}$ and the set of natural numbers.

$$\textsc{Ans}_{\text{ISWIM}} = \{\texttt{function}\} \cup \mathbb{N}.$$

The set of programs $\textsc{Prgm}_{\text{ISWIM}}$ and the set of answers $\textsc{Ans}_{\text{ISWIM}}$ are defined for ISWIM, not for any particular dialect or semantics of ISWIM.

Evaluator. We now define an evaluator in terms of the substitutional structural operational semantics of ISWIM.

Definition 62 (Evaluator based on Substitutional ISWIM)**.** Let the evaluator $eval_{\text{ISWIM:SubSOS}}$ be a partial function from the set of programs $\textsc{Prgm}_{\text{ISWIM}}$ to the set of answers $\textsc{Ans}_{\text{ISWIM}}$.

$$eval_{\text{ISWIM:SubSOS}} : \textsc{Prgm}_{\text{ISWIM}} \rightharpoonup \textsc{Ans}_{\text{ISWIM}}$$

$$eval_{\text{ISWIM:SubSOS}}(t) = \begin{cases} \texttt{function} & \text{if } t \longrightarrow^* \lambda x.t' \\ n & \text{if } t \longrightarrow^* n \end{cases}$$

This evaluator is defined in terms of the substitutional structural operational semantics of ISWIM. The subscript "$_{\text{ISWIM:SubSOS}}$" in $eval_{\text{ISWIM:SubSOS}}$ denotes the substitutional structural operational semantics of ISWIM. The evaluator is essentially the same as the single-stage subset of the evaluator defined in terms of the substitutional structural operational semantics of MetaML.

3.2 Explicit ISWIM

Consider again the (app) rule of the substitutional structural operational semantics of ISWIM.

$$\frac{}{(\lambda x.t)\ v \longrightarrow t[v/x]}\ \text{(app)}$$

Because the substitution function $\cdot[\cdot/\cdot]$ is defined as equations in the meta-language, the (app) rule says that in the expression $t[v/x]$ each free occurrence of the variable x is *immediately* replaced by the value v in the term t. Evaluating a substitution does not take any additional step regardless of how complicated the substitution is. It is not evident how to attain our objective of this chapter, i.e., how to develop an environmental abstract machine for ISWIM, based directly on the substitutional structural operational semantics of ISWIM.

We propose to turn the big gap of what is less clear into several small moves of what is more evident, each of which leads to an intermediate semantics. As the very first move, we integrate the percolation of substitutions into the structural operational semantics, leading to explicit substitutions [Cur85, ACCL91]. As a result, the (app) rule becomes

$$\frac{}{(\lambda x.t)\ v \longrightarrow t[x := v]}\ \text{(app)}$$

where $[x := v]$ is an explicit substitution. Depending on how complex the term t is, it may take several steps to percolate the substitution $[x := v]$ through the term t.

We call the resulting dialect Explicit ISWIM and present its structural operational semantics.

3.2.1 Syntax

We first define the basic syntax of Explicit ISWIM: terms, values and denotable terms. Then we define the free variable function, the substitution function and the alpha equivalence relation.

3.2.1.1 Terms, Values and Denotable Terms

Definition 63 (Terms, Values and Denotable Terms)**.** Let (1) TERM be the set of terms, (2) VALUE be the set of values, and (3) DENOTABLE be the set of denotable terms.

$x \in \text{VAR}$, $n \in \mathbb{N}$, $t \in \text{TERM}$, $v \in \text{VALUE}$, $w \in \text{DENOTABLE}$

$$\begin{array}{lcl} t & := & x \mid t\,t \mid \lambda x.t \mid n \mid t + t \mid \boxed{t[x := w]} \\ v & := & \lambda x.t \mid n \\ w & := & x \mid v \end{array}$$

The language has been enhanced with a term surrounded by an explicit substitution $t[x := w]$, which means that each free occurrence of the variable x in the term t *needs* to be substituted by the denotable term w. Evaluating an explicit substitution takes steps. In contrast, an implicit substitution $t[w/x]$ used in Substitutional ISWIM represents the *result* of substituting the denotable term w for each free occurrence of the variable x in the term t.

3.2.1.2 Free Variable Function

We define the free variable function by extending Definition 47 to accommodate explicit substitutions.

Definition 64 (Free Variable Function)**.** Let the free variable function FV be a total function from the set of terms to the power set of variables.

$$FV : \text{TERM} \longrightarrow \mathscr{P}(\text{VAR})$$

$$\begin{array}{rcll} \vdots & \vdots & \vdots & \vdots \\ FV(t[x := w]) & = & (FV(t) \backslash \{x\}) \cup FV(w) & (6) \end{array}$$

Equations (1)-(5) are the same as Definition 47 in Substitutional ISWIM. Recall that $t[x := w]$ is introduced to represent what an application $(\lambda x.t)\ w$ evaluates to. Observe that there is no free variable introduced or eliminated during the evaluation. We have $FV(t[x := w]) = FV((\lambda x.t)\ w) = (FV(t) \backslash \{x\}) \cup FV(w)$.

3.2.1.3 Substitution Function

We define the substitution function by extending Definition 49 to accommodate explicit substitutions.

Definition 65 (Substitution Function)**.** Let the substitution function $\cdot[\cdot/\cdot]$ be a total function from the 3-tuple of the set of terms, the set of denotable terms and the set of variables, to the set of terms.

$$\cdot[\cdot/\cdot] : (\text{TERM} \times \text{DENOTABLE} \times \text{VAR}) \longrightarrow \text{TERM}$$

$$\begin{array}{rcll} \vdots & \vdots & \vdots & \vdots \\ (t_0[x_1 := w_1])[w_2/x_2] & = & t_0[x_3/x_1][w_2/x_2][x_3 := w_1[w_2/x_2]] & \\ & & \text{where } x_3 \notin FV(\lambda x_1.t_0) \cup FV(w) \cup \{x_2\} & (7) \end{array}$$

Equations (1)-(6) are the same as Definition 49 in Substitutional ISWIM. Recall that Proposition 54 says $t[w_1/x_1][w_2/x_2] \sim_\alpha t[x_3/x_1][w_2/x_2][w_1[w_2/x_2]/x_3]$ where $x_3 \notin FV(\lambda x_1.t_0) \cup FV(w) \cup \{x_2\}$, corresponding to Equation (7).

3.2.1.4 Alpha Equivalence Relation

We define the alpha equivalence relation by extending Definition 50 to accommodate explicit substitutions.

Definition 66 (Alpha Equivalence Relation)**.** Define the alpha equivalence relation $\sim_\alpha$ to be a binary relation between the set of terms and the set of terms.

$$\sim_\alpha \subseteq \text{TERM} \times \text{TERM}$$

$$\vdots$$

$$\frac{w_1 \sim_\alpha w_2 \quad t_1[x_3/x_1] \sim_\alpha t_2[x_3/x_2]}{(t_1[x_1 := w_1]) \sim_\alpha (t_2[x_2 := w_2])} \text{ where } x_3 \notin FV(t_1) \cup FV(t_2) \text{ (sub)}$$

All rules except the (sub) rule are the same as Definition 50 in Substitutional ISWIM. Recall that $t[x := w]$ is introduced to represent what an application $(\lambda x.t)\ w$ evaluates to. If we can show $(\lambda x_1.t_1)\ w_1 \sim_\alpha (\lambda x_2.t_2)\ w_2$, then we should be able to show $(t_1[x_1 := w_1]) \sim_\alpha (t_2[x_2 := w_2])$. The premise of the (sub) rule indeed shows $(\lambda x_1.t_1)\ w_1 \sim_\alpha (\lambda x_2.t_2)\ w_2$.

3.2.2 Structural Operational Semantics

We lay out the structural operational semantics through the single-step relation $\longrightarrow$, the single-step substitution reduction relation $\longrightarrow^{\mathrm{x}}$, the multi-step substitution reduction relation $\longrightarrow^{\mathrm{x}*}$ and the multi-step relation $\longrightarrow^{*}$.

Definition 67 (Single-step Relation)**.** Define the single-step relation $\longrightarrow$ to be a binary relation between the set of terms and the set of terms.

$$\longrightarrow \subseteq \textsc{Term} \times \textsc{Term}$$

$$\frac{t_{11} \longrightarrow t_{12}}{t_{11}\ t_2 \longrightarrow t_{12}\ t_2}\ \text{(appL)} \qquad \frac{t_{21} \longrightarrow t_{22}}{v_1\ t_{21} \longrightarrow v_1\ t_{22}}\ \text{(appR)} \qquad \boxed{\frac{}{(\lambda x.t)\ v \longrightarrow t[x := v]}\ \text{(app)}}$$

$$\frac{t_{11} \longrightarrow t_{12}}{t_{11} + t_2 \longrightarrow t_{12} + t_2}\ \text{(plusL)} \qquad \frac{t_{21} \longrightarrow t_{22}}{v_1 + t_{21} \longrightarrow v_1 + t_{22}}\ \text{(plusR)} \qquad \frac{}{n_1 + n_2 \longrightarrow n}\ \text{where } n = n_1 + n_2\ \text{(plus)}$$

...

The only rule that is different from Substitutional ISWIM is (app) which replaces the meta-language substitution $[v/x]$ with the explicit substitution $[x := v]$. The definition of the single-step relation is currently incomplete because how explicit substitutions percolate has not been defined yet.

To specify how explicit substitutions percolate, we define a new relation $t[x := w] \longrightarrow^{\mathrm{x}} t$. This relation ensures that explicit substitutions percolate deterministically.

Definition 68 (Single-step Substitution Reduction Relation)**.** Let the single-step substitution reduction relation $\longrightarrow^{\mathrm{x}}$ be a binary relation between the set of terms and the set of terms.

$$\longrightarrow^{\mathrm{x}} \subseteq \textsc{Term} \times \textsc{Term}$$

$$\frac{}{x[x := w] \longrightarrow^{\mathrm{x}} w}\ \text{(var-eq-subst)} \qquad \frac{}{x_1[x_2 := w] \longrightarrow^{\mathrm{x}} x_1}\ \text{where } x_1 \not\equiv x_2\ \text{(var-df-subst)}$$

$$\frac{}{n[x := w] \longrightarrow^{\mathrm{x}} n}\ \text{(num-subst)}$$

$$\frac{}{(t_1\ t_2)[x := w] \longrightarrow^{\mathrm{x}} (t_1[x := w])\ (t_2[x := w])}\ \text{(app-subst)}$$

$$\frac{}{(t_1 + t_2)[x := w] \longrightarrow^{\mathrm{x}} (t_1[x := w]) + (t_2[x := w])}\ \text{(plus-subst)}$$

$$\frac{}{(\lambda x_1.t)[x_2 := w] \longrightarrow^{\mathrm{x}} \lambda x_3.t[x_1 := x_3][x_2 := w]}\ \text{where } x_3 \notin FV(\lambda x_1.t) \cup FV(w) \cup \{x_2\}\ \text{(lam-subst)}$$

$$\frac{t_1[x_1 := w_1] \longrightarrow^{\mathrm{x}} t_2}{t_1[x_1 := w_1][x_2 := w_2] \longrightarrow^{\mathrm{x}} t_2[x_2 := w_2]}\ \text{(subst-subst)}$$

Most of the rules describe how explicit substitutions behave when encountering other terms in the language. Rules (var-eq-subst), (var-df-subst), (app-subst), (lam-subst), (num-subst) and (plus-subst) correspond to Equations (1) to (5) of Substitutional ISWIM's substitution function (Definition 49) respectively. The (subst-subst) rule implies that only a single-step of substitution reduction may happen underneath an explicit substitution.

Every single-step substitution reduction counts as a single step of computation. We add the following rule to the definition of the single-step relation, Definition 67.

$$\frac{}{t_1[x := w] \longrightarrow t_2} \text{ where } t_1[x := w] \longrightarrow^{\mathrm{x}} t_2 \text{ (subst)}$$

Definition 69 (Multi-step Substitution Reduction Relation)**.** Define the multi-step substitution reduction relation $\longrightarrow^{\mathrm{x}*}$ to be the reflexive-transitive closure relation on the single-step substitution reduction relation $\longrightarrow^{\mathrm{x}}$.

Definition 70 (Multi-step Relation)**.** Define the multi-step relation $\longrightarrow^*$ to be the reflexive-transitive closure relation on the single-step relation $\longrightarrow$.

Example 71. Let $x_1 \not\equiv x_2$. We observe that

$$((\lambda x_1.x_2)\ 5)[x_2 := 2] \not\longrightarrow (x_2[x_1 := 5])[x_2 := 2].$$

As $(\lambda x_1.x_2)\ 5 \longrightarrow x_2[x_1 := 5]$ is merely a single-step computation but not a substitution reduction, it cannot be performed underneath the explicit substitution $[x_2 := 2]$.

The explicit substitution $[x_2 := 2]$ has to propagate first. Correctly, we have:

$$\begin{array}{lll}
 & ((\lambda x_1.x_2)\ 5)[x_2 := 2] & \\
\longrightarrow & (\lambda x_1.x_2)[x_2 := 2]\ 5[x_2 := 2] & \\
\longrightarrow & (\lambda x_1.x_2[x_1 := x_1][x_2 := 2])\ 5[x_2 := 2] & \text{where } x_1 \not\equiv x_2 \\
\longrightarrow & (\lambda x_1.x_2[x_1 := x_1][x_2 := 2])\ 5 & \\
\longrightarrow & x_2[x_1 := x_1][x_2 := 2][x_1 := 5] & \\
\longrightarrow & x_2[x_2 := 2][x_1 := 5] & \\
\longrightarrow & 2[x_1 := 5] & \\
\longrightarrow & 2 &
\end{array}$$

Example 72. Let $x_1 \not\equiv x_2$. By the structural operational semantics of Explicit ISWIM, we have:

$$\begin{array}{lll}
 & (\lambda x_1.\lambda x_2.x_1)\ 7 & \\
\longrightarrow & (\lambda x_2.x_1)[x_1 := 7] & \\
\longrightarrow & \lambda x_2.x_1[x_2 := x_2][x_1 := 7] & \text{where } x_2 \not\equiv x_1
\end{array}$$

In contrast, by the substitutional structural operational semantics of ISWIM, we have:

$$\begin{array}{lll}
 & (\lambda x_1.\lambda x_2.x_1)\ 7 & \\
\longrightarrow & (\lambda x_2.x_1)[7/x_1] & \\
= & \lambda x_2.x_1[x_2/x_2][7/x_1] & \text{where } x_2 \not\equiv x_1 \\
= & \lambda x_2.x_1[7/x_1] & \\
= & \lambda x_2.7 &
\end{array}$$

Explicit ISWIM takes two single-steps while Substitutional ISWIM completes the execution in one single-step. Explicit ISWIM terminates at a lambda abstraction where its body is a term surrounded by explicit substitutions. Explicit ISWIM reduces substitutions in a lazy fashion in the sense that it only pushes explicit substitutions to the body of a lambda abstraction but does not perform any substitution reduction on the body of a lambda abstraction.

Properties. We observe the following properties that are useful in proving semantics equivalence. (1) The single-step relation preserves alpha equivalence. (2) The multi-step relation preserves the closedness of a term. These are the same properties that Substitutional ISWIM holds.

Proposition 73. *If* $t_{a_1} \sim_\alpha t_{b_1}$ *and* $t_{a_1} \longrightarrow t_{a_2}$*, then* $t_{b_1} \longrightarrow t_{b_2}$ *and* $t_{a_2} \sim_\alpha t_{b_2}$.

Proposition 74. *If* $FV(t_1) = \emptyset$ *and* $t_1 \longrightarrow^* t_2$*, then* $FV(t_2) = \emptyset$.

Evaluator. We now define an evaluator in terms of the structural operational semantics of Explicit ISWIM. The evaluator is analogous to the one defined in terms of Substitutional ISWIM.

Definition 75 (Evaluator based on Structural Operational Semantics of Explicit ISWIM)**.** Define the evaluator $eval_{\text{ISWIM:ExpSOS}}$ to be a partial function from the set of programs $\text{PRGM}_{\text{ISWIM}}$ to the set of answers $\text{ANS}_{\text{ISWIM}}$.

$$eval_{\text{ISWIM:ExpSOS}} : \text{PRGM}_{\text{ISWIM}} \rightharpoonup \text{ANS}_{\text{ISWIM}}$$

$$eval_{\text{ISWIM:ExpSOS}}(t) = \begin{cases} \texttt{function} & \text{if } t \longrightarrow^* \lambda x.t' \\ n & \text{if } t \longrightarrow^* n \end{cases}$$

This evaluator is defined in terms of the structural operational semantics of Explicit ISWIM. The subscript "$_{\text{ISWIM:ExpSOS}}$" in $eval_{\text{ISWIM:ExpSOS}}$ denotes the structural operational semantics of Explicit ISWIM.

We claim that the evaluators defined in terms of the Substitutional ISWIM and Explicit ISWIM are equivalent.

Theorem 76 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{ISWIM}}$, $eval_{\text{ISWIM:SubSOS}}(t)$ *is Kleene equal to* $eval_{\text{ISWIM:ExpSOS}}(t)$.

We prove the theorem in the appendices.

3.3 Suspended ISWIM

Explicit ISWIM models how substitutions percolate at the semantical level. However, the integration has introduced unnecessary or unconventional computation. Consider the (lam-subst) rule of the structural operational semantics of Explicit ISWIM.

$$\frac{}{(\lambda x_1.t)[x_2 := w] \longrightarrow^{\text{x}} \lambda x_3.t[x_1 := x_3][x_2 := w]} \text{ where } x_3 \notin FV(\lambda x_1.t) \cup FV(w) \cup \{x_2\} \text{ (lam-subst)}$$

Pushing an explicit substitution into a lambda abstraction requires rewriting the body of the lambda abstraction. Furthermore, when an explicit substitution is pushed into a lambda abstraction, a new explicit substitution to rename the lambda bound variable is created, which may get pushed downward.

We propose to delay explicit substitutions outside of any lambda abstraction until the lambda abstraction is called in an application. The resulting dialect is Suspended ISWIM.

3.3.1 Syntax

We first define the basic syntax of the Suspended ISWIM: terms, values and denotable terms. Then we define the free variable function, the substitution function and the alpha equivalence relation.

3.3.1.1 Terms, Values and Denotable Terms

Definition 77 (Terms, Values and Denotable Terms)**.** Let (1) TERM be the set of terms, (2) VALUE be the set of values, and (3) DENOTABLE be the set of denotable terms.

$x \in \text{VAR}$, $n \in \mathbb{N}$, $t \in \text{TERM}$, $v \in \text{VALUE}$, $w \in \text{DENOTABLE}$

$$\begin{array}{lcl} t & ::= & x \mid t\,t \mid \lambda x.t \mid n \mid t+t \mid t[x := w] \\ v & ::= & \boxed{(\lambda x.t)\overline{[x := w]}} \mid n \\ w & ::= & x \mid v \end{array}$$

The definition uses $(\lambda x.t)\overline{[x := w]}$ to represent that the lambda abstraction $\lambda x.t$ is surrounded by zero or more explicit substitutions $\overline{[x := w]}$.[2] Since we delay explicit substitutions outside of lambda abstractions, $(\lambda x.t)\overline{[x := w]}$ is a value in Suspended ISWIM.

A term surrounded by explicit substitutions, $t[x_1 := w_1][x_2 := w_2]...[x_n := w_n]$, truly represents a term surrounded by explicit substitutions cascadedly, $(...((t[x_1 := w_1])[x_2 := w_2])...)[x_n := w_n]$. We usually omit the parentheses for convenience.

3.3.1.2 Free Variable Function, Substitution Function and Alpha Equivalence Relation

The free variable function, substitution function and alpha equivalence relation are the same as Explicit ISWIM (Section 3.2).

3.3.2 Structural Operational Semantics

We lay out the structural operational semantics through the single-step relation $\longrightarrow$, the single-step substitution reduction relation $\longrightarrow^{\text{x}}$, the multi-step substitution reduction relation $\longrightarrow^{\text{x}*}$ and the multi-step relation $\longrightarrow^{*}$.

Definition 78 (Single-step Relation)**.** Let the single-step relation $\longrightarrow$ be a binary relation between the set of terms and the set of terms.

[2] We state several syntactic conventions. (1) $\overline{[x_i := w_i]}$ denotes zero or more explicit substitutions. (2) $\overline{[x_i := w_i]}_{i=1}^{n}$ denotes either zero explicit substitutions or n explicit substitutions of the form $[x_1 := w_1][x_2 := w_2]...[x_n := w_n]$. (3) $\overline{[x_i := w_i]}_{i=-1}^{-n}$ denotes either zero explicit substitutions or n explicit substitutions of the form $[x_{-1} := w_{-1}][x_{-2} := w_{-2}]...[x_{-n} := w_{-n}]$. (4) $\overline{[x_i := w_i]}^{+}$ denotes one or more explicit substitutions. $\overline{[x_i := w_i]}_{i=-1}^{+-n}$ denotes n explicit substitutions of the form $[x_{-1} := w_{-1}][x_{-2} := w_{-2}]...[x_{-n} := w_{-n}]$.

$$\longrightarrow \subseteq \text{TERM} \times \text{TERM}$$

$$\frac{t_{11} \longrightarrow t_{12}}{t_{11}\ t_2 \longrightarrow t_{12}\ t_2}\ \text{(appL)} \qquad \frac{t_{21} \longrightarrow t_{22}}{v_1\ t_{21} \longrightarrow v_1\ t_{22}}\ \text{(appR)} \qquad \boxed{\frac{}{(\lambda x.t)\overline{[x_i := w_i]}\ v \longrightarrow t[x := v]\overline{[x_i := w_i]}}\ \text{(app)}}$$

$$\frac{t_{11} \longrightarrow t_{12}}{t_{11}+t_2 \longrightarrow t_{12}+t_2}\ \text{(plusL)} \qquad \frac{t_{21} \longrightarrow t_{22}}{v_1+t_{21} \longrightarrow v_1+t_{22}}\ \text{(plusR)} \qquad \frac{}{n_1+n_2 \longrightarrow n}\ \text{where } n = n_1+n_2\ \text{(plus)}$$

$$\frac{}{t_1[x := w] \longrightarrow t_2}\ \text{where } t_1[x := w] \longrightarrow^{\text{x}} t_2\ \text{(subst)}$$

The only rule that is different from Explicit ISWIM is the (app) rule.

Someone may attempt to use the following one as the (app) rule.

$$\frac{}{(\lambda x.t)\overline{[x_i := w_i]}\ v \longrightarrow t\overline{[x_i := w_i]}[x := v]}\ \text{(app-incorrect)}$$

This rule is incorrect. For example, if (app-incorrect) is used, we have $((\lambda x.x)[x := 9])\ 7 \longrightarrow x[x := 9][x := 7] \longrightarrow 9[x := 7] \longrightarrow 9$. In contrast, Explicit ISWIM evaluates $((\lambda x.x)[x := 9])\ 7$ to 7, and Substitutional ISWIM evaluates $((\lambda x.x)[9/x])\ 7$ to 7 as well.

Someone may attempt to use the following one as the (app) rule.

$$\frac{}{(\lambda x.t)\overline{[x_i := w_i]}\ v \longrightarrow t[x := x_0]\overline{[x_i := w_i]}[x_0 := v]}\ \text{where } x_0 \notin FV(\lambda x.t) \cup \bigcup_i (FV(w_i) \cup \{x_i\})\ \text{(app-optional)}$$

This rule is correct but not ideal. We want to eliminate all renamings in Suspended ISWIM, but (app-optional) still renames the lambda bound variable.

Keep in mind that this chapter aims to develop an environmental operational semantics for ISWIM. The (app) rule promotes the substitution for the lambda bound variable to the front, overwriting any existing explicit substitution for that variable, which is close to the operation of updating an environment. We shall keep the (app) rule.

To specify how explicit substitutions percolate, we define the single-step substitution reduction relation $\longrightarrow^{\text{x}}$.

Definition 79 (Single-step Substitution Reduction Relation)**.** Let the single-step substitution reduction relation $\longrightarrow^{\text{x}}$ be a binary relation between the set of terms and the set of terms.

$$\longrightarrow^{\mathrm{x}} \subseteq \textsc{Term} \times \textsc{Term}$$

$$\frac{}{x[x := w] \longrightarrow^{\mathrm{x}} w}\ \text{(var-eq-subst)} \qquad \frac{}{x_1[x_2 := w] \longrightarrow^{\mathrm{x}} x_1}\ \text{where } x_1 \not\equiv x_2\ \text{(var-df-subst)}$$

$$\frac{}{n[x := w] \longrightarrow^{\mathrm{x}} n}\ \text{(num-subst)}$$

$$\frac{}{(t_1\ t_2)[x := w] \longrightarrow^{\mathrm{x}} (t_1[x := w])\ (t_2[x := w])}\ \text{(app-subst)}$$

$$\frac{}{(t_1 + t_2)[x := w] \longrightarrow^{\mathrm{x}} (t_1[x := w]) + (t_2[x := w])}\ \text{(plus-subst)}$$

no (lam-subst)

$$\frac{t_1[x_1 := w_1] \longrightarrow^{\mathrm{x}} t_2}{t_1[x_1 := w_1][x_2 := w_2] \longrightarrow^{\mathrm{x}} t_2[x_2 := w_2]}\ \text{(subst-subst)}$$

All rules are the same as the single-step substitution reduction relation of Explicit ISWIM. The (lam-subst) rule no longer exists in Suspended ISWIM because a lambda abstraction surrounded by explicit substitutions is a value.

The above two definitions have fully eliminated variable renamings. The denotable term w in an arbitrary substitution $[x := w]$ must be a value. As a result, the definition of denotable terms in Definition 77

$$w := x \mid v$$

shall be replaced by

$$w := v$$

where $w \in \textsc{Denotable}$, $x \in \textsc{Var}$ and $v \in \textsc{Value}$. For convenience, we may use $[x := v]$ or keep using $[x := w]$ to represent an explicit substitution in Suspended ISWIM.

Definition 80 (Multi-step Substitution Reduction Relation)**.** Define the multi-step substitution reduction relation $\longrightarrow^{\mathrm{x}*}$ to be the reflexive-transitive closure relation on the single-step substitution reduction relation $\longrightarrow^{\mathrm{x}}$.

Definition 81 (Multi-step Relation)**.** Define the multi-step relation $\longrightarrow^{*}$ to be the reflexive-transitive closure relation on the single-step relation $\longrightarrow$.

Example 82. Consider $((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4$ where $x_1 \not\equiv x_2$. By the structural operational semantics of Suspended ISWIM, we have:

$$\begin{aligned}
& ((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4 \\
\longrightarrow\ & ((\lambda x_2.x_1)[x_1 := 7])\ 4 \\
\longrightarrow\ & x_1[x_2 := 4][x_1 := 7] \\
\longrightarrow\ & x_1[x_1 := 7] \\
\longrightarrow\ & 7.
\end{aligned}$$

In contrast, by the structural operational semantics of Explicit ISWIM, we have:

$$\begin{array}{ll} & ((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4 \\ \longrightarrow & ((\lambda x_2.x_1)[x_1 := 7])\ 4 \\ \longrightarrow & (\lambda x_2.x_1[x_2 := x_2][x_1 := 7])\ 4 \quad \text{where } x_2 \not\equiv x_1 \\ \longrightarrow & x_1[x_2 := x_2][x_1 := 7][x_2 := 4] \\ \longrightarrow & x_1[x_1 := 7][x_2 := 4] \\ \longrightarrow & 7[x_2 := 4] \\ \longrightarrow & 7 \end{array}$$

Suspended ISWIM takes two fewer steps than Explicit ISWIM. Given $(\lambda x_2.x_1)[x_1 := 7]$, Suspended ISWIM does not push the substitution $[x_1 := 7]$ into the lambda abstraction $\lambda x_2.x_1$. Instead, the substitution is suspended until the lambda abstraction is called in an application. Given $((\lambda x_2.x_1)[x_1 := 7])\ 4$, Suspended ISWIM promotes the substitution for the lambda bound variable to the front, resulting in $x_1[x_2 := 4][x_1 := 7]$.

Properties. We observe the following properties that are useful in proving semantics equivalence. (1) The single-step relation preserves alpha equivalence. (2) The multi-step relation preserves the closedness of a term. These are the same properties that Substitutional ISWIM and Explicit ISWIM hold.

Proposition 83. *If $t_{a_1} \sim_\alpha t_{b_1}$ and $t_{a_1} \longrightarrow t_{a_2}$, then $t_{b_1} \longrightarrow t_{b_2}$ and $t_{a_2} \sim_\alpha t_{b_2}$.*

Proposition 84. *If $FV(t_1) = \emptyset$ and $t_1 \longrightarrow^* t_2$, then $FV(t_2) = \emptyset$.*

Evaluator. We now define an evaluator in terms of the substitutional structural operational semantics of Suspended ISWIM. The evaluator is analogous to be one defined for Explicit ISWIM.

Definition 85 (Evaluator based on Structural Operational Semantics of Suspended ISWIM)**.** Let the evaluator $eval_{\text{ISWIM:SusSOS}}$ be a partial function from the set of programs $\text{PRGM}_{\text{ISWIM}}$ to the set of answers $\text{ANS}_{\text{ISWIM}}$.

$$eval_{\text{ISWIM:SusSOS}} : \text{PRGM}_{\text{ISWIM}} \rightharpoonup \text{ANS}_{\text{ISWIM}}$$

$$eval_{\text{ISWIM:SusSOS}}(t) = \begin{cases} \texttt{function} & \text{if } t \longrightarrow^* (\lambda x.t')\overline{[x_i := w_i]} \\ n & \text{if } t \longrightarrow^* n \end{cases}$$

This evaluator is defined in terms of the structural operational semantics of Suspended ISWIM. The subscript "$_{\text{ISWIM:SusSOS}}$" in $eval_{\text{ISWIM:SusSOS}}$ denotes the structural operational semantics of Suspended ISWIM.

We claim that the evaluators defined in terms of the Substitutional ISWIM and Suspended ISWIM are equivalent.

Theorem 86 (Kleene Equality of Evaluators)**.** *For any $t \in \text{PRGM}_{\text{ISWIM}}$, $eval_{\text{ISWIM:SubSOS}}(t)$ is Kleene equal to $eval_{\text{ISWIM:SusSOS}}(t)$.*

We prove the theorem in the appendices.

3.4 Environmental ISWIM - Structural Operational Semantics

Suspended ISWIM is peculiar in the sense that the top-level structure of a lambda abstraction surrounded by zero or more explicit substitutions is not immediately recognisable. Consider $(\lambda x.t)\overline{[x_i := w_i]}_{i=1}^{n}$ which truly represents $(...(((\lambda x.t)[x_1 := w_1])[x_2 := w_2])...)[x_n := w_n]$. To ensure that its top-level structure is a lambda abstraction, we have to dive down through the cascaded explicit substitutions to search for a lambda abstraction.

Furthermore, since our evaluator only concerns closed terms, we can safely claim that all denotable terms substituting variables in cascaded explicit substitutions are closed. Then a cascade of explicit substitutions can be viewed as a whole when operating on a term and its subterms. For example, consider a cascade of explicit substitutions operating on a variable, $x\overline{[x_i := w_i]}_{i=1}^{n}$. We compare x against x_i where $i = 1, 2, ..., n$ in order. There are two possibilities. (1) If none of x_i's refers to x, the cascaded explicit substitutions disappear, i.e., $x\overline{[x_i := w_i]}_{i=1}^{n}$ steps to x. (2) If we find the leftmost x_p where $1 \leq \mathrm{p} \leq n$ such that x_p refers to x, then x is replaced by w_p and the remaining cascaded substitutions $\overline{[x_i := w_i]}_{i=\mathrm{p}+1}^{n}$ disappear, i.e., $x\overline{[x_i := w_i]}_{i=1}^{n}$ steps to w_p where x_p is the leftmost x_i such that $x \equiv x_\mathrm{p}$. We make these two observations because the denotable term w_p is closed and remains unchanged when encountering any explicit substitution. After further analysis, we realise that there will always be a cascade of explicit substitutions that reach a variable in Suspended ISWIM. It is a natural step to treat a cascade of explicit substitutions as a whole and replace it by an environment, leading to Environmental ISWIM.

3.4.1 Syntax

We first define the basic syntax of the Environmental ISWIM: terms, values, denotable terms, configurations and environments. Then we define the free variable function.

3.4.1.1 Terms, Values, Denotable Terms, Configurations and Environments

Definition 87 (Terms, Values, Denotable Terms and Configurations)**.** Let (1) TERM be the set of terms, (2) VALUE be the set of values, (3) DENOTABLE be the set of denotable terms, (4) CONF be the set of configurations, and (5) ENV be a finite partial function from the set of variables to the set of denotable terms.

$x \in \text{VAR}$, $n \in \mathbb{N}$, $t \in \text{TERM}$, $v \in \text{VALUE}$, $w \in \text{DENOTABLE}$, $c \in \text{CONF}$, $\rho \in \text{ENV} = \text{VAR} \xrightarrow{\text{fin}} \text{DENOTABLE}$

$$
\begin{array}{lcl}
t & := & x \mid t\ t \mid \lambda x.t \mid n \mid t + t \\
v & := & n \mid \triangleleft \lambda x.t,\ \rho \triangleright \text{ where } FV(\lambda x.t) \subseteq \mathrm{dom}(\rho) \\
w & := & v \\
c & := & v \mid c\ c \mid c + c \mid \langle t,\ \rho \rangle \text{ where } FV(t) \subseteq \mathrm{dom}(\rho)
\end{array}
$$

A pair of a term and an environment where the former is closed by the latter, i.e., $\langle t,\ \rho \rangle$ where $FV(t) \subseteq \mathrm{dom}(\rho)$ or $\triangleleft \lambda x.t,\ \rho \triangleright$ where $FV(\lambda x.t) \subseteq \mathrm{dom}(\rho)$, is called a *closure*. A pair of a lambda abstraction and an environment where the former is closed by the latter, i.e., $\triangleleft \lambda x.t,\ \rho \triangleright$ where $FV(\lambda x.t) \subseteq \mathrm{dom}(\rho)$, is called a *closure value*.

A closure makes its top-level structure immediately evident. For example, given a closure $\langle \lambda x.t,\ \rho \rangle$, it is immediately recognisable that its top-level structure is a lambda abstraction $\lambda x.t$ without having to dive into the environment ρ. In contrast, in Suspended ISWIM, to check the top-level structure of $(\lambda x.t)\overline{[x := w]}$, we have to dive down through the cascaded explicit substitutions $\overline{[x := w]}$ until reaching the lambda abstraction $\lambda x.t$.

Unlike the previous dialects of ISWIM, Environmental ISWIM deems the set of configurations rather than the set of terms to be the fundamental set on which the operational semantics is defined.

Definition 88 (Environments)**.** An environment $\rho \in \text{ENV}$ is a finite partial function from the set of variables to the set of denotable terms. Let $\text{dom}(\rho)$ be the domain of the environment ρ and $\text{rng}(\rho)$ be the range of the environment ρ. Let $\rho[x \mapsto w]$ be an environment update and $\rho(x)$ be an environment lookup. We have:

$$\rho[x \mapsto w](y) = \begin{cases} w & \text{if } x \equiv y \\ \rho(y) & \text{if } x \not\equiv y \end{cases}$$

Suppose an environment ρ maps x_1 to w_1, x_2 to w_2, ..., x_n to w_n, where $x_i \not\equiv x_j$ for any i, j such that $i \neq j$, and ρ has no other mapping. The environment ρ can be represented as a finite set $\{(x_1, w_1), (x_2, w_2), ..., (x_n, w_n)\}$. The domain of the environment ρ is $\text{dom}(\rho) = \{x_1, x_2, ..., x_n\}$ and the range of the environment ρ is $\text{rng}(\rho) = \{w_1, w_2, ..., w_n\}$.

3.4.1.2 Free Variable Function

We define the free variable function by extending Definition 47 to accommodate configurations.

Definition 89 (Free Variable Function)**.** Let the free variable function FV be a total function from the set of configurations to the power set of variables.

$$FV : \text{CONF} \longrightarrow \mathscr{P}(\text{VAR})$$

$$\begin{aligned}
FV(x) &= x && (1)\\
FV(t_1\ t_2) &= FV(t_1) \cup FV(t_2) && (2)\\
FV(\lambda x.t) &= FV(t) \backslash \{x\} && (3)\\
FV(n) &= \emptyset && (4)\\
FV(t_1 + t_2) &= FV(t_1) \cup FV(t_2) && (5)\\
FV(\langle t,\ \rho \rangle) &= \emptyset && (6)\\
FV(\triangleleft \lambda x.t,\ \rho \triangleright) &= FV(\langle \lambda x.t,\ \rho \rangle) && (7)\\
FV(c_1\ c_2) &= FV(c_1) \cup FV(c_2) && (8)\\
FV(c_1 + c_2) &= FV(c_1) \cup FV(c_2) && (9)
\end{aligned}$$

Equations (1)-(5) are the same as Definition 47 in Substitutional ISWIM. Equations (6) and (7) are based on the definitions of closures and closure values. Equations (8) and (9) are analogous to Equations (2) and (5).

Definition 90 (Closed Configurations)**.** A configuration c is *closed* if and only if $FV(c) = \emptyset$.

3.4.2 Structural Operational Semantics

We lay out the structural operational semantics of Environmental ISWIM through the single-step relation $\longrightarrow$ and the multi-step relation $\longrightarrow^*$.

Definition 91 (Single-step Relation)**.** Define the single-step relation $\longrightarrow$ to be a binary relation between the set of configurations and the set of configurations.

$$\longrightarrow \subseteq \textsc{Conf} \times \textsc{Conf}$$

$$\frac{c_{11} \longrightarrow c_{12}}{c_{11}\ c_2 \longrightarrow c_{12}\ c_2}\ \text{(appL)} \qquad \frac{c_{21} \longrightarrow c_{22}}{v_1\ c_{21} \longrightarrow v_1\ c_{22}}\ \text{(appR)} \qquad \frac{}{\lhd(\lambda x.t),\ \rho \rhd\ v \longrightarrow \langle t,\ \rho[x \mapsto v]\rangle}\ \text{(app)}$$

$$\frac{c_{11} \longrightarrow c_{12}}{c_{11} + c_2 \longrightarrow c_{12} + c_2}\ \text{(plusL)} \qquad \frac{c_{21} \longrightarrow c_{22}}{v_1 + c_{21} \longrightarrow v_1 + c_{22}}\ \text{(plusR)}$$

$$\frac{}{n_1 + n_2 \longrightarrow n}\ \text{where } n = n_1 + n_2\ \text{(plus)}$$

$$\frac{}{\langle(\lambda x.t),\ \rho\rangle \longrightarrow \lhd(\lambda x.t),\ \rho\rhd}\ \text{(clos-env)}$$

$$\frac{}{\langle x,\ \rho\rangle \longrightarrow w}\ \text{where } \rho(x) = w\ \text{(var-env)}$$

$$\frac{}{\langle n,\ \rho\rangle \longrightarrow n}\ \text{(num-env)}$$

$$\frac{}{\langle(t_1\ t_2),\ \rho\rangle \longrightarrow \langle t_1,\ \rho\rangle\ \langle t_2,\ \rho\rangle}\ \text{(app-env)}$$

$$\frac{}{\langle(t_1 + t_2),\ \rho\rangle \longrightarrow \langle t_1,\ \rho\rangle + \langle t_2,\ \rho\rangle}\ \text{(plus-env)}$$

The single-step relation $c_1 \longrightarrow c_2$ reads as "c_1 single-steps to c_2". (1) Rules (appL), (appR), (plusL), (plusR) and (plus) are analogous to the rules of the same names in Suspended ISWIM's single-step relation. The only difference is that Suspended ISWIM defined the relation on terms but we now define the relation on configurations. (2) The (app) rule models performing an application by environment updating, which completes the unfinished job of Suspended ISWIM's (app) rule. (3) The other rules discuss how to evaluate a closure. The (clos-env) rule turns a closure to a closure value. Other (*-env) rules correspond to Suspended ISWIM's single-step substitution reduction relation.

Rules of the single-step relation can be categorised into reduction rules and structural rules. Structural rules are (appL), (appR), (plusL) and (plusR). The others are reduction rules.

Definition 92 (Multi-step Relation)**.** Define the multi-step relation $\longrightarrow^*$ to be the reflexive-transitive closure of the single-step relation $\longrightarrow$.

Example. To get familiar with the structural operational semantics, consider the following example.

Example 93. Consider $(\lambda x_1.\lambda x_2.x_1)\ 7)\ 4$ where $x_1 \not\equiv x_2$.

We first construct a configuration that pairs the above term with an empty environment, i.e., $\langle((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4,\ \emptyset\rangle$. By the structural operational semantics of Environmental ISWIM, we have:

$$
\begin{array}{lll}
 & \langle((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4,\ \emptyset\rangle & (1)\\
\longrightarrow & \langle(\lambda x_1.\lambda x_2.x_1)\ 7,\ \emptyset\rangle\ \langle 4,\ \emptyset\rangle & (2)\\
\longrightarrow & (\langle(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rangle\ \langle 7,\ \emptyset\rangle)\ \langle 4,\ \emptyset\rangle & (3)\\
\longrightarrow & (\lhd(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rhd\ \langle 7,\ \emptyset\rangle)\ \langle 4,\ \emptyset\rangle & (4)\\
\longrightarrow & (\lhd(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rhd\ 7)\ \langle 4,\ \emptyset\rangle & (5)\\
\longrightarrow & \langle(\lambda x_2.x_1),\ \{(x_1,7)\}\rangle\ \langle 4,\ \emptyset\rangle & (6)\\
\longrightarrow & \lhd(\lambda x_2.x_1),\{(x_1,7)\}\rhd\ \langle 4,\ \emptyset\rangle & (7)\\
\longrightarrow & \lhd(\lambda x_2.x_1),\{(x_1,7)\}\rhd\ 4 & (8)\\
\longrightarrow & \langle x_1,\{(x_2,4),(x_1,7)\}\rangle & (9)\\
\longrightarrow & 7. & (10)
\end{array}
$$

As shown in Line (1), we always evaluate a program with an empty environment.

Property. We observe the following property that is useful in proving semantics equivalence. The multi-step relation preserves the closedness of a term. This is the same property that Substitutional ISWIM, Explicit ISWIM and Suspended ISWIM hold.

Proposition 94. *If $FV(c_1) = \emptyset$ and $c_1 \longrightarrow^* c_2$, then $FV(c_2) = \emptyset$.*

Evaluator. We now define an evaluator in terms of the structural operational semantics of Environmental ISWIM. Environmental ISWIM's multi-step relation is defined on sets of configurations rather than sets of terms. Given a program t, the evaluator applies the multi-step relation on the configuration $\langle t,\ \emptyset\rangle$ in which the program is associated with an empty environment. The evalutor is otherwise analogous to the evaluator defined for Suspended ISWIM.

Definition 95 (Evaluator based on Structural Operational Semantics of Environmental ISWIM)**.** Let the evaluator $eval_{\text{ISWIM:EnvSOS}}$ to be a partial function from the set of programs $\text{PRGM}_{\text{ISWIM}}$ to the set of answers $\text{ANS}_{\text{ISWIM}}$.

$$
eval_{\text{ISWIM:EnvSOS}} : \text{PRGM}_{\text{ISWIM}} \rightharpoonup \text{ANS}_{\text{ISWIM}}
$$

$$
eval_{\text{ISWIM:EnvSOS}}(t) = \begin{cases} \texttt{function} & \text{if } \langle t,\ \emptyset\rangle \longrightarrow^* \lhd(\lambda x.t'),\ \rho\rhd \\ n & \text{if } \langle t,\ \emptyset\rangle \longrightarrow^* n \end{cases}
$$

This evaluator is defined in terms of the structural operational semantics of Environmental ISWIM. The subscript "$_{\text{ISWIM:EnvSOS}}$" in $eval_{\text{ISWIM:EnvSOS}}$ denotes the structural operational semantics of Environmental ISWIM.

We claim that the evaluators defined in terms of Substitutional ISWIM and Environmental ISWIM are equivalent.

Theorem 96 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{ISWIM}}$, $eval_{\text{ISWIM:SubSOS}}(t)$ *is Kleene equal to* $eval_{\text{ISWIM:EnvSOS}}(t)$.

We prove the theorem in appendices.

3.5 Environmental ISWIM - Reduction Semantics

This chapter aims to develop an environmental abstract machine for ISWIM. Since a reduction semantics can be viewed as a concise representation of an abstract machine, we develop a reduction semantics for Environmental ISWIM, based on which we develop an abstract machine in the next section.

3.5.1 Syntax

The definitions of terms, values, denotable terms, configurations and environments are the same as Section 3.4.

3.5.1.1 Evaluation Contexts

We now define evaluation contexts to regulate the only places where an arbitrary reduction may happen.

Definition 97 (Evalutation Contexts: Inside-out)**.** Let ECXT be the set of evaluation contexts.

$$E \in \text{ECXT},\ c \in \text{CONF},\ v \in \text{VALUE}$$

$$\frac{}{\Box \in \text{ECXT}}\ (\text{ept})$$

$$\frac{E \in \text{ECXT}}{E[\Box\ c] \in \text{ECXT}}\ (\text{appL}) \qquad \frac{E \in \text{ECXT}}{E[v\ \Box] \in \text{ECXT}}\ (\text{appR})$$

$$\frac{E \in \text{ECXT}}{E[\Box + c] \in \text{ECXT}}\ (\text{plusL}) \qquad \frac{E \in \text{ECXT}}{E[v + \Box] \in \text{ECXT}}\ (\text{plusR})$$

The sole hole $\Box$ in an evaluation context can be filled by a configuration. $E[c]$ is a configuration constructed by filling the sole hole of the evaluation context E by the configuration c.

There is a correspondence between evaluation contexts and structural rule of the single-step relation defined as Definition 91. (1) The evaluation context $E[\Box\ c]$ allows reduction at the operator position of an application, corresponding to the (appL) rule of the semantics. (2) The evaluation context $E[v\ \Box]$ allows reduction at the operand position of an application, corresponding to the (appR) rule of the semantics. (3) The evaluation context $E[\Box + c]$ allows reduction at the first operand position of an addition, corresponding to the (plusL) rule of the semantics. (4) The evaluation context $E[v + \Box]$ allows reduction at second operand position of an addition, corresponding to the (plusR) rule of the semantics. (5) The evaluation context $\Box$ allows reduction immediately.

This definition is called *inside-out* because it makes the innermost structure of an evaluation context the most evident. An alternative but equivalent definition of evaluation contexts is provided in Definition 106.

3.5.2 Reduction Semantics

We lay out the reduction semantics of Environmental ISWIM through the notions of reduction $\mathscr{R}$, the reduction relation $\longmapsto$ and the multi-reduction relation $\longmapsto^*$.

Definition 98 (Notions of Reduction)**.** Let the notions of reduction, $\mathscr{R}$, be a binary relation between the set of configurations and the set of configurations.

$$\mathscr{R} \subseteq \textsc{Conf} \times \textsc{Conf}$$

$$\begin{array}{rcll}
\lhd(\lambda x.t),\, \rho \rhd v & \mathscr{R} & \langle t,\, \rho[x \mapsto v]\rangle & \text{(app)} \\
n_1 + n_2 & \mathscr{R} & n \text{ where } n = n_1 + n_2 & \text{(plus)} \\
\langle(\lambda x.t),\, \rho\rangle & \mathscr{R} & \lhd(\lambda x.t),\, \rho\rhd & \text{(conf-lam)} \\
\langle x,\, \rho\rangle & \mathscr{R} & w \text{ where } \rho(x) = w & \text{(conf-var)} \\
\langle n,\, \rho\rangle & \mathscr{R} & n & \text{(conf-num)} \\
\langle(t_1\, t_2),\, \rho\rangle & \mathscr{R} & \langle t_1,\, \rho\rangle\, \langle t_2,\, \rho\rangle & \text{(conf-app)} \\
\langle(t_1 + t_2),\, \rho\rangle & \mathscr{R} & \langle t_1,\, \rho\rangle + \langle t_2,\, \rho\rangle & \text{(conf-plus)}
\end{array}$$

The notion of reduction $c_1 \mathscr{R} c_2$ reads as "c_1 reduces to c_2". Each notion corresponds to one reduction rule of the single-step relation defined as Definition 91.

Definition 99. If $c_1 \mathscr{R} c_2$, then c_1 is a *redex* and c_2 is a *contractum*.

Definition 100 (Reduction Relation)**.** Let the reduction relation $\longmapsto$ be a binary relation between the set of configurations and the set of configurations directly based on the notions of reduction $\mathscr{R}$.

$$\longmapsto \subseteq \textsc{Conf} \times \textsc{Conf}$$

$$\frac{c_1 \mathscr{R} c_2}{E[c_1] \longmapsto E[c_2]}$$

The reduction relation $c_1 \longmapsto c_2$ reads as "c_1 single-reduces to c_2". The above definition states that the reduction relation respects performing any notion of reduction in an evaluation context.

Intuitively, the reduction relation $\longmapsto$ defines a single step of computation. We define $\longmapsto^*$ to represent multiple (zero or more) steps of computation.

Definition 101 (Multi-reduction Relation)**.** Define the multi-reduction relation $\longmapsto^*$ to be the reflexive-transitive closure of the reduction relation $\longmapsto$.

The multi-reduction relation $c_1 \longmapsto^* c_2$ reads as "c_1 multi-reduces to c_2".

Example 102. Consider $(\lambda x_1.\lambda x_2.x_1)\ 7)\ 4$ where $x_1 \not\equiv x_2$.

We first pair above term with an empty environment, constructing the configuration $\langle((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4,\, \emptyset\rangle$. By the reduction semantics of Environmental ISWIM, we have:

$$\langle((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4,\, \emptyset\rangle \longmapsto^* 7$$

as demonstrated in Figure 3.1.

$$
\begin{array}{lll}
 & \langle((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4,\ \emptyset\rangle & \\
= & \Box[\langle((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4,\ \emptyset\rangle] & \\
\longmapsto & \Box[\langle(\lambda x_1.\lambda x_2.x_1)\ 7,\ \emptyset\rangle\ \langle 4,\ \emptyset\rangle] & \text{where } \langle((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4,\ \emptyset\rangle\ \mathscr{R}\ \langle(\lambda x_1.\lambda x_2.x_1)\ 7,\ \emptyset\rangle\ \langle 4,\ \emptyset\rangle \\
= & \langle(\lambda x_1.\lambda x_2.x_1)\ 7,\ \emptyset\rangle\ \langle 4,\ \emptyset\rangle & \\
= & \Box[\Box\ \langle 4,\ \emptyset\rangle][\langle(\lambda x_1.\lambda x_2.x_1)\ 7,\ \emptyset\rangle] & \\
\longmapsto & \Box[\Box\ \langle 4,\ \emptyset\rangle][\langle(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rangle\ \langle 7,\ \emptyset\rangle] & \text{where } \langle(\lambda x_1.\lambda x_2.x_1)\ 7,\ \emptyset\rangle\ \mathscr{R}\ \langle(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rangle\ \langle 7,\ \emptyset\rangle \\
= & (\langle(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rangle\ \langle 7,\ \emptyset\rangle)\ \langle 4,\ \emptyset\rangle & \\
= & \Box[\Box\ \langle 4,\ \emptyset\rangle][\Box\ \langle 7,\ \emptyset\rangle][\langle(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rangle] & \\
\longmapsto & \Box[\Box\ \langle 4,\ \emptyset\rangle][\Box\ \langle 7,\ \emptyset\rangle][\lhd(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rhd] & \text{where } \langle(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rangle\ \mathscr{R}\ \lhd(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rhd \\
= & (\lhd(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rhd\ \langle 7,\ \emptyset\rangle)\ \langle 4,\ \emptyset\rangle & \\
= & \Box[\Box\ \langle 4,\ \emptyset\rangle][\lhd(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rhd\ \Box][\langle 7,\ \emptyset\rangle] & \\
\longmapsto & \Box[\Box\ \langle 4,\ \emptyset\rangle][\lhd(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rhd\ \Box][7] & \text{where } \langle 7,\ \emptyset\rangle\ \mathscr{R}\ 7 \\
= & (\lhd(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rhd\ 7)\ \langle 4,\ \emptyset\rangle & \\
= & \Box[\Box\ \langle 4,\ \emptyset\rangle][\lhd(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rhd\ 7] & \\
\longmapsto & \Box[\Box\ \langle 4,\ \emptyset\rangle][\langle(\lambda x_2.x_1),\ \{(x_1,7)\}\rangle] & \text{where } \lhd(\lambda x_1.\lambda x_2.x_1),\ \emptyset\rhd\ 7\ \mathscr{R}\ \langle(\lambda x_2.x_1),\ \emptyset[x_1 \mapsto 7]\rangle \\
= & \langle(\lambda x_2.x_1),\ \{(x_1,7)\}\rangle\ \langle 4,\ \emptyset\rangle & \\
= & \Box[\Box\ \langle 4,\ \emptyset\rangle][\langle(\lambda x_2.x_1),\ \{(x_1,7)\}\rangle] & \\
\longmapsto & \Box[\Box\ \langle 4,\ \emptyset\rangle][\lhd(\lambda x_2.x_1),\ \{(x_1,7)\}\rhd] & \text{where } \langle(\lambda x_2.x_1),\ \{(x_1,7)\}\rangle\ \mathscr{R}\ \lhd(\lambda x_2.x_1),\ \{(x_1,7)\}\rhd \\
= & \lhd(\lambda x_2.x_1),\{(x_1,7)\}\rhd\ \langle 4,\ \emptyset\rangle & \\
= & \Box[\lhd(\lambda x_2.x_1),\{(x_1,7)\}\rhd\ \Box][\langle 4,\ \emptyset\rangle] & \\
\longmapsto & \Box[\lhd(\lambda x_2.x_1),\{(x_1,7)\}\rhd\ \Box][4] & \text{where } \langle 4,\ \emptyset\rangle\ \mathscr{R}\ 4 \\
= & \lhd(\lambda x_2.x_1),\{(x_1,7)\}\rhd\ 4 & \\
= & \Box[\lhd(\lambda x_2.x_1),\{(x_1,7)\}\rhd\ 4] & \\
\longmapsto & \Box[\langle x_1,\{(x_2,4),(x_1,7)\}\rangle] & \text{where } \lhd(\lambda x_2.x_1),\{(x_1,7)\}\rhd\ 4\ \mathscr{R}\ \langle x_1,\{(x_1,7)\}[x_2 \mapsto 4]\rangle \\
= & \langle x_1,\{(x_2,4),(x_1,7)\}\rangle & \\
= & \Box[\langle x_1,\{(x_2,4),(x_1,7)\}\rangle] & \\
\longmapsto & \Box[7] & \text{where } \langle x_1,\{(x_2,4),(x_1,7)\}\rangle\ \mathscr{R}\ 7 \\
= & 7. &
\end{array}
$$

Figure 3.1: Evaluation of $((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4$ in Reduction Semantics of Environmental ISWIM.

To apply the reduction semantics on a term, follow the following pattern repeatedly until the resulting term is a value.

1. Break the term into an evaluation context and a redex.
2. Apply a notion of reduction on the redex and get a contractum.
3. Plug the contractum into the evaluation context and get a new term.

However, the reduction semantics does not tell how to break a term into an evaluation context and a redex. In other words, the reduction semantics does not encode a systematic strategy to search for an evaluation context and a redex.

Property. We observe the following property that is useful in proving semantics equivalence. The multi-step relation preserves the closedness of a term. This is the same property that the previos ISWIM dialects hold.

Proposition 103. *If* $FV(c_1) = \emptyset$ *and* $c_1 \longmapsto^* c_2$*, then* $FV(c_2) = \emptyset$.

Evaluator. We now define an evaluator in terms of the reduction semantics of Environmental ISWIM. The evalutor is analogous to the evaluator defined in terms of the structural operational semantics of Environmental ISWIM.

Definition 104 (Evaluator based on Reduction Semantics of Environmental ISWIM)**.** Let the evaluator $eval_{\text{ISWIM:EnvRed}}$ be a partial function from the set of programs $\text{PRGM}_{\text{ISWIM}}$ to the set of answers $\text{ANS}_{\text{ISWIM}}$.

$$eval_{\text{ISWIM:EnvRed}} : \text{PRGM}_{\text{ISWIM}} \rightharpoonup \text{ANS}_{\text{ISWIM}}$$

$$eval_{\text{ISWIM:EnvRed}}(t) = \begin{cases} \texttt{function} & \text{if } \langle t, \emptyset \rangle \longmapsto^* \triangleleft(\lambda x.t'), \rho\triangleright \\ n & \text{if } \langle t, \emptyset \rangle \longmapsto^* n \end{cases}$$

This evaluator is defined in terms of the reduction semantics of Environmental ISWIM. The subscript "$_{\text{ISWIM:EnvRed}}$" in $eval_{\text{ISWIM:EnvRed}}$ denotes the reduction semantics of Environmental ISWIM.

We claim that the evaluators defined in terms of the structural operational semantics and the reduction semantics of Environmental ISWIM are equivalent.

Theorem 105 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{ISWIM}}$, $eval_{\text{ISWIM:EnvSOS}}(t)$ *is Kleene equal to* $eval_{\text{ISWIM:EnvRed}}(t)$.

We prove the theorem in appendices.

3.6 Environmental ISWIM - CEK Abstract Machine

Reduction semantics can be viewed as a concise representation of an abstract machine in the sense that it abstracts away the search for an evaluation context and a redex. In contrast, an abstract machine encodes a systematic strategy to search for an evaluation context and a redex. We finish refining semantics for ISWIM by developing an abstract machine for Environmental ISWIM. The environmental abstract machine is also known as the CEK machine.

3.6.1 Syntax

The definitions of terms, values, denotable terms, configurations and environments are the same as Sections 3.4 and 3.5.

3.6.1.1 Evaluation Contexts

We provide two definitions of evaluation contexts. The first definition is the same as the one used in Section 3.5. The second one is as follows. These two definitions are equivalent and are used interchangeably at our convenience.

Definition 106 (Evaluation Contexts: Outside-in)**.** Let ECXT be the set of evaluation contexts.

$$E \in \text{ECXT},\ c \in \text{CONF},\ v \in \text{VALUE}$$

$$\frac{}{\Box \in \text{ECXT}}\ \text{(ept)}$$

$$\frac{E \in \text{ECXT}}{(E\ c) \in \text{ECXT}}\ \text{(appL)} \qquad \frac{E \in \text{ECXT}}{(v\ E) \in \text{ECXT}}\ \text{(appR)}$$

$$\frac{E \in \text{ECXT}}{(E + c) \in \text{ECXT}}\ \text{(plusL)} \qquad \frac{E \in \text{ECXT}}{(v + E) \in \text{ECXT}}\ \text{(plusR)}$$

This definition is called *outside-in* because it makes the outermost structure of an evaluation context the most evident.

3.6.1.2 Machine Configurations

The states of an abstract machine are represented by machine configurations.

Definition 107 (Machine Configurations)**.** Let CFG be the set of machine configurations.

$$C \in \text{CFG},\ c \in \text{CONF},\ v \in \text{VALUE},\ E \in \text{ECXT}$$

$$\begin{aligned} C \quad ::= \quad & v \\ & \mid \langle E,\ c\rangle_{\mathrm{r}} \\ & \mid \langle E,\ c\rangle_{\mathrm{f}} \\ & \mid \langle E,\ v\rangle_{\mathrm{b}} \end{aligned}$$

The machine operates in four modes: the *value* mode v, the *reduce* mode $\langle E,\ c\rangle_{\mathrm{r}}$, the *focus* mode $\langle E,\ c\rangle_{\mathrm{f}}$ and the *build* mode $\langle E,\ v\rangle_{\mathrm{b}}$.

A machine configuration $\langle E,\ c\rangle_?$ where $? \in \{\mathrm{r},\mathrm{f},\mathrm{b}\}$ unloads to the configuration $E[c]$. Precisely, the configuration c in the machine configuration $\langle E,\ c\rangle_\mathrm{r}$ needs to be a redex.

3.6.2 CEK Abstract Machine

We lay out an abstract machine of Environmental ISWIM, which is known as the CEK machine, through the reduction relation $\longmapsto_{\mathrm{cek}}$ and the multi-reduction relation $\longmapsto^*_{\mathrm{cek}}$.

Definition 108 (Reduction Relation)**.** Let the reduction relation $\longmapsto_{\mathrm{cek}}$ be a binary relation between the set of machine configurations and the set of machine configurations.

$$\longmapsto_{\mathrm{cek}} \subseteq \textsc{Cfg} \times \textsc{Cfg}$$

Reduce rules:

$$\begin{array}{rcll}
\langle E,\ \lhd(\lambda x.t),\ \rho \rhd v\rangle_\mathrm{r} & \longmapsto_{\mathrm{cek}} & \langle E,\ \langle t,\ \rho[x \mapsto v]\rangle\rangle_\mathrm{f} & \text{(r-app)} \\
\langle E,\ n_1 + n_2\rangle_\mathrm{r} & \longmapsto_{\mathrm{cek}} & \langle E,\ n\rangle_\mathrm{f} \text{ where } n = n_1 + n_2 & \text{(r-plus)} \\
\langle E,\ \langle(\lambda x.t),\ \rho\rangle\rangle_\mathrm{r} & \longmapsto_{\mathrm{cek}} & \langle E,\ \lhd(\lambda x.t),\ \rho\rhd\rangle_\mathrm{f} & \text{(r-conf-lam)} \\
\langle E,\ \langle x,\ \rho\rangle\rangle_\mathrm{r} & \longmapsto_{\mathrm{cek}} & \langle E,\ w\rangle_\mathrm{f} \text{ where } \rho(x) = w & \text{(r-conf-var)} \\
\langle E,\ \langle n,\ \rho\rangle\rangle_\mathrm{f} & \longmapsto_{\mathrm{cek}} & \langle E,\ n\rangle_\mathrm{f} & \text{(r-conf-num)} \\
\langle E,\ \langle(t_1\ t_2),\ \rho\rangle\rangle_\mathrm{r} & \longmapsto_{\mathrm{cek}} & \langle E,\ \langle t_1,\ \rho\rangle\ \langle t_2,\ \rho\rangle\rangle_\mathrm{f} & \text{(r-conf-app)} \\
\langle E,\ \langle(t_1 + t_2),\ \rho\rangle\rangle_\mathrm{r} & \longmapsto_{\mathrm{cek}} & \langle E,\ \langle t_1,\ \rho\rangle + \langle t_2,\ \rho\rangle\rangle_\mathrm{f} & \text{(r-conf-plus)}
\end{array}$$

Focus rules:

$$\begin{array}{rcll}
\langle E,\ \langle t,\ \rho\rangle\rangle_\mathrm{f} & \longmapsto_{\mathrm{cek}} & \langle E,\ \langle t,\ \rho\rangle\rangle_\mathrm{r} & \text{(f-conf)} \\
\langle E,\ c_1\ c_2\rangle_\mathrm{f} & \longmapsto_{\mathrm{cek}} & \langle E[\Box\ c_2],\ c_1\rangle_\mathrm{f} & \text{(f-app)} \\
\langle E,\ \lhd(\lambda x.t),\ \rho\rhd\rangle_\mathrm{f} & \longmapsto_{\mathrm{cek}} & \langle E,\ \lhd(\lambda x.t),\ \rho\rhd\rangle_\mathrm{b} & \text{(f-lam)} \\
\langle E,\ n\rangle_\mathrm{f} & \longmapsto_{\mathrm{cek}} & \langle E,\ n\rangle_\mathrm{b} & \text{(f-num)} \\
\langle E,\ c_1 + c_2\rangle_\mathrm{f} & \longmapsto_{\mathrm{cek}} & \langle E[\Box + c_2],\ c_1\rangle_\mathrm{f} & \text{(f-plus)}
\end{array}$$

Build rules:

$$\begin{array}{rcll}
\langle \Box,\ v\rangle_\mathrm{b} & \longmapsto_{\mathrm{cek}} & v & \text{(b-val)} \\
\langle E[\Box\ c_2],\ v_1\rangle_\mathrm{b} & \longmapsto_{\mathrm{cek}} & \langle E[v_1\ \Box],\ c_2\rangle_\mathrm{f} & \text{(b-appL)} \\
\langle E[v_1\ \Box],\ v_2\rangle_\mathrm{b} & \longmapsto_{\mathrm{cek}} & \langle E,\ v_1\ v_2\rangle_\mathrm{r} & \text{(b-appR)} \\
\langle E[\Box + c_2],\ v_1\rangle_\mathrm{b} & \longmapsto_{\mathrm{cek}} & \langle E[v_1 + \Box],\ c_2\rangle_\mathrm{f} & \text{(b-plusL)} \\
\langle E[v_1 + \Box],\ v_2\rangle_\mathrm{b} & \longmapsto_{\mathrm{cek}} & \langle E,\ v_1 + v_2\rangle_\mathrm{r} & \text{(b-plusR)}
\end{array}$$

The reduction relation $C_1 \longmapsto_{\mathrm{cek}} C_2$ reads as "C_1 single-reduces to C_2".

A machine configuration at the reduce mode, $\langle E,\ c\rangle_\mathrm{r}$, signifies that a proper notion of reduction can be applied on the redex c. A machine configuration at the focus mode, $\langle E,\ c\rangle_\mathrm{f}$, indicates searching downward into the configuration c for a redex to reduce. A machine configuration at the build mode, $\langle E,\ v\rangle_\mathrm{b}$, returns the value v to the current evaluation context E. A machine configuration at the value mode, v, represents that the value v is the result of executing the machine.

Intuitively, the reduction relation $\longmapsto_{\text{cek}}$ defines a single step of computation. We define $\longmapsto^*_{\text{cek}}$ to represent multiple (zero or more) steps of computation.

Definition 109 (Multi-reduction Relation)**.** Let the multi-reduction relation $\longmapsto^*_{\text{cek}}$ be the reflexive-transitive closure of the reduction relation $\longmapsto_{\text{cek}}$.

The multi-reduction relation $C_1 \longrightarrow^*_{\text{cek}} C_2$ reads as "C_1 multi-reduces to C_2".

The abstract machine defined above is also known as the CEK machine. C stands for control, i.e., the configuration under evaluation, E stands for environment, and K stands for continuation, i.e., the evaluation context.

Example 110. Consider $(\lambda x_1.\lambda x_2.x_1)\ 7)\ 4$ where $x_1 \not\equiv x_2$.

We first construct a machine configuration that contains the above term with an empty evaluation context and an empty environment and start running the machine configuration at focus mode.

By the CEK machine, we have

$$\langle \square,\ \langle ((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4,\ \emptyset \rangle \rangle_{\text{f}} \longmapsto^*_{\text{cek}} 7$$

as demonstrated in Figure 3.2.

Evaluator. We now define an evaluator in terms of the CEK machine. CEK machine's multi-reduction relation is defined on sets of machine configurations. Given a program t, the evaluator applies the multi-transformation relation on the machine configuration $\langle \square,\ \langle t,\ \emptyset \rangle \rangle_{\text{f}}$ in which the program is associated with an empty environment and an empty evaluation context. The evaluator is otherwise analogous to the one defined in terms of the reduction semantics of Environmental ISWIM.

Definition 111 (Evaluator based on CEK Machine)**.** Let the evaluator $eval_{\text{ISWIM:CEK}}$ be a partial function from the set of programs $\text{PRGM}_{\text{ISWIM}}$ to the set of answers $\text{ANS}_{\text{ISWIM}}$.

$$eval_{\text{ISWIM:CEK}} : \text{PRGM}_{\text{ISWIM}} \rightharpoonup \text{ANS}_{\text{ISWIM}}$$

$$eval_{\text{ISWIM:CEK}}(t) = \begin{cases} \texttt{function} & \text{if } \langle \square,\ \langle t,\ \emptyset \rangle \rangle_{\text{f}} \longmapsto^*_{\text{cek}} \lhd(\lambda x.t'),\ \rho \rhd \\ n & \text{if } \langle \square,\ \langle t,\ \emptyset \rangle \rangle_{\text{f}} \longmapsto^*_{\text{cek}} n \end{cases}$$

This evaluator is defined in terms of the CEK machine. The subscript "$_{\text{ISWIM:CEK}}$" in $eval_{\text{ISWIM:CEK}}$ denotes the CEK machine of ISWIM.

We claim that the evaluators defined in terms of the reduction semantics and based on the CEK abstract machine of Environmental ISWIM are equivalent.

Theorem 112 (Kleene Equality of Evaluators)**.** *For any $t \in \text{PRGM}_{\text{ISWIM}}$, $eval_{\text{ISWIM:EnvRed}}(t)$ is Kleene equal to $eval_{\text{ISWIM:CEK}}(t)$.*

$$
\begin{array}{lll}
 & \langle \Box, & \langle ((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4,\ \emptyset \rangle \quad \rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box, & \langle ((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4,\ \emptyset \rangle \quad \rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box, & \langle (\lambda x_1.\lambda x_2.x_1)\ 7,\ \emptyset \rangle\ \langle 4,\ \emptyset \rangle \quad \rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle], & \langle (\lambda x_1.\lambda x_2.x_1)\ 7,\ \emptyset \rangle \quad \rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle], & \langle (\lambda x_1.\lambda x_2.x_1)\ 7,\ \emptyset \rangle \quad \rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle], & \langle \lambda x_1.\lambda x_2.x_1,\ \emptyset \rangle\ \langle 7,\ \emptyset \rangle \quad \rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle][\Box\ \langle 7,\ \emptyset \rangle], & \langle \lambda x_1.\lambda x_2.x_1,\ \emptyset \rangle \quad \rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle][\Box\ \langle 7,\ \emptyset \rangle], & \langle \lambda x_1.\lambda x_2.x_1,\ \emptyset \rangle \quad \rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle][\Box\ \langle 7,\ \emptyset \rangle], & \triangleleft \lambda x_1.\lambda x_2.x_1,\ \emptyset \triangleright \quad \rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle][\Box\ \langle 7,\ \emptyset \rangle], & \triangleleft \lambda x_1.\lambda x_2.x_1,\ \emptyset \triangleright \quad \rangle_{\mathrm{b}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle][\triangleleft \lambda x_1.\lambda x_2.x_1,\ \emptyset \triangleright\ \Box], & \langle 7,\ \emptyset \rangle \quad \rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle][\triangleleft \lambda x_1.\lambda x_2.x_1,\ \emptyset \triangleright\ \Box], & \langle 7,\ \emptyset \rangle \quad \rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle][\triangleleft \lambda x_1.\lambda x_2.x_1,\ \emptyset \triangleright\ \Box], & 7 \quad \rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle][\triangleleft \lambda x_1.\lambda x_2.x_1,\ \emptyset \triangleright\ \Box], & 7 \quad \rangle_{\mathrm{b}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle], & \triangleleft \lambda x_1.\lambda x_2.x_1,\ \emptyset \triangleright\ 7 \quad \rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle], & \langle \lambda x_2.x_1,\ \{(x_1,7)\} \rangle \quad \rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle], & \langle \lambda x_2.x_1,\ \{(x_1,7)\} \rangle \quad \rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle], & \triangleleft \lambda x_2.x_1,\ \{(x_1,7)\} \triangleright \quad \rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\Box\ \langle 4,\ \emptyset \rangle], & \triangleleft \lambda x_2.x_1,\ \{(x_1,7)\} \triangleright \quad \rangle_{\mathrm{b}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\triangleleft \lambda x_2.x_1,\ \{(x_1,7)\} \triangleright\ \Box], & \langle 4,\ \emptyset \rangle \quad \rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\triangleleft \lambda x_2.x_1,\ \{(x_1,7)\} \triangleright\ \Box], & \langle 4,\ \emptyset \rangle \quad \rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\triangleleft \lambda x_2.x_1,\ \{(x_1,7)\} \triangleright\ \Box], & 4 \quad \rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box[\triangleleft \lambda x_2.x_1,\ \{(x_1,7)\} \triangleright\ \Box], & 4 \quad \rangle_{\mathrm{b}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box, & \triangleleft \lambda x_2.x_1,\ \{(x_1,7)\} \triangleright\ 4 \quad \rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box, & \langle x_1,\ \{(x_1,7),(x_2,4)\} \rangle \quad \rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box, & \langle x_1,\ \{(x_1,7),(x_2,4)\} \rangle \quad \rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box, & 7 \quad \rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle \Box, & 7 \quad \rangle_{\mathrm{b}} \\
\longmapsto_{\mathrm{cek}} & 7 &
\end{array}
$$

Figure 3.2: Evaluation of $((\lambda x_1.\lambda x_2.x_1)\ 7)\ 4$ in the CEK Machine.

We prove the above theorem in appendices.

As a corollary, the evaluators defined in terms of the Substitutional ISWIM and the CEK machine are equivalent.

Corollary 113 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{ISWIM}}$*,* $eval_{\text{ISWIM:SubSOS}}(t)$ *is Kleene equal to* $eval_{\text{ISWIM:CEK}}(t)$*.*

Proof. It immediately follows from Theorems 96, 105 and 112 by the transitivity of Kleene equality. □

3.7 Chapter Summary

Following the first dimension of the semantics refinement problem, this chapter solved the following problem that is less complicated than the main semantics refinement problem.

> Can we refine the substitutional structural operational semantics of ISWIM to a corresponding substitutional abstract machine, which is known as the CEK machine, and demonstrate their equivalence?

We accomplished the development progressively in several manageable steps, each of which led to an intermediate semantics. We first studied the substitutional structural operational semantics of ISWIM. Then we successively developed the structural operational semantics of Explicit ISWIM, the structural operational semantics of Suspended ISWIM, the structural operational semantics of Environmental ISWIM, the reduction semantics of Environmental ISWIM, and finally derived the abstract machine of Environmental ISWIM. The abstract machine of Environmental ISWIM is also known as the CEK machine.

We defined an evaluator based on each semantics. By proving the equivalence of every two adjacent semantics, we finally showed that the CEK machine is equivalent to the substitutional structural operational semantics of ISWIM.

Chapter 4

Refining Semantics for MetaML: Developing the MK Machine

Following the second dimension of simplifying the main semantics refinement problem, we study how to stepwise develop an substitutional abstract machine rather than an environmental abstract machine for the multi-stage language MetaML. The problem is restated as follows.

> Can we refine the substitutional structural operational semantics of MetaML to a corresponding substitutional abstract machine, which we call the MK machine, and demonstrate their equivalence?

Recall that Sections 3.4, 3.5 and 3.6 have shown an approach to refining a structural operational semantics to a reduction semantics and finally to an abstract machine. We adopt the same strategy in deriving the MK machine.

4.1 MetaML - Substitutional Reduction Semantics

MetaML's substitutional structural operational semantics has been presented in Section 2.2. Following the path of refining a structural operational semantics to a reduction semantics as shown in Section 3.5, we derive a substitutional reduction semantics for MetaML.

4.1.1 Syntax

The definitions of terms, values and denotable terms are the same as Section 2.2.1.

4.1.1.1 Evaluation Contexts

Section 3.5 defined evaluation contexts to regulate the only places where an arbitrary reduction may happen in a single-stage language. We now extend the definition to accommodate the multi-stage setting.

Definition 114 (Level-indexed Evaluation Contexts: Inside-out)**.** Let $\textsc{Ecxt}^{i-\circ j}$ be the set of evaluation contexts with inner level i and outer level j.

$$E^{i\multimap j} \in \text{ECXT}^{i\multimap j},\ t^i \in \text{TERM}^i,\ v^i \in \text{VALUE}^i$$

$$\frac{}{\Box \in \text{ECXT}^{i\multimap i}}\ \text{(ept-i)}$$

$$\frac{E \in \text{ECXT}^{i\multimap j}}{E[\Box\ t_2^i] \in \text{ECXT}^{i\multimap j}}\ \text{(appL-i)} \qquad \frac{E \in \text{ECXT}^{i\multimap j}}{E[v_1^i\ \Box] \in \text{ECXT}^{i\multimap j}}\ \text{(appR-i)}$$

$$\frac{E \in \text{ECXT}^{(i+1)\multimap j}}{E[\lambda x.\Box] \in \text{ECXT}^{(i+1)\multimap j}}\ \text{(lambda-(i+1))}$$

$$\frac{E \in \text{ECXT}^{i\multimap j}}{E[\langle\Box\rangle] \in \text{ECXT}^{(i+1)\multimap j}}\ \text{(code-i)} \qquad \frac{E \in \text{ECXT}^{(i+1)\multimap j}}{E[\sim\Box] \in \text{ECXT}^{i\multimap j}}\ \text{(splice-(i+1))} \qquad \frac{E \in \text{ECXT}^{i\multimap j}}{E[!\Box] \in \text{ECXT}^{i\multimap j}}\ \text{(run-i)}$$

$$\frac{E \in \text{ECXT}^{i\multimap j}}{E[\Box + t_2^i] \in \text{ECXT}^{i\multimap j}}\ \text{(plusL-i)} \qquad \frac{E \in \text{ECXT}^{i\multimap j}}{E[v_1^i + \Box] \in \text{ECXT}^{i\multimap j}}\ \text{(plusR-i)}$$

An evaluation context $E^{i\multimap j}$ comes with an inner level i and an outer level j. The inner level i is the level of the hole of the context, indicating the level of terms that can fill the hole. The outer level j is the level of the term produced by the context when the hole of the context is filled.

The sole hole $\Box^{i\multimap i}$ in an evaluation context can be filled by a level-i term. $E^{i\multimap j}[t^i]$ is a level-j term constructed by filling the sole hole of the evaluation context $E^{i\multimap j}$ by a level-i term t^i. The levels i and j in an evaluation context $E^{i\multimap j}$ can be related in any of the following three ways. (1) $i > j$. For example, a level-0 term $!\langle\lambda x.x\rangle$ can be represented as $(\Box[!\Box][\langle\Box\rangle][\lambda x.\Box])^{1\multimap 0}[x]$. (2) $i = j$. For example, a level-0 term $(\lambda x.x)\ ((\lambda x.x)\ 7)$ can be represented as $(\Box[(\lambda x.x)\ \Box])^{0\multimap 0}[(\lambda x.x)\ 7]$. (3) $i < j$. For example, a level-1 term $\sim((\lambda x.x)\ \langle\lambda x.x\rangle)$ can be represented as $(\Box[\sim\Box])^{0\multimap 1}[(\lambda x.x)\ \langle\lambda x.x\rangle]$.

The definition of evaluation contexts is motivated by the structural rules of the single-step relations (Definition 36). For example, the (code-i) rule

$$\frac{t_1^{i+1} \longrightarrow^{i+1} t_2^{i+1}}{\langle t_1^{i+1}\rangle \longrightarrow^{i} \langle t_2^{i+1}\rangle}\ \text{(code-i)}$$

tells that a code operation at level i can be evaluated by reducing its operand at level $i+1$. Since an evaluation context defines where a reduction may happen, we may replace the (code-i) rule by an evaluation context $(\Box[\langle\Box\rangle])^{(i+1)\multimap i}$ and allow any level-$(i+1)$ reduction to happen at the hole of the context. We observe the following correspondences between evaluation contexts and structural rule of the single-step relations (Definition 36). (1) The evaluation context $E[\Box\ t_2^i]$ corresponds to (appL-i). (2) The evaluation context $E[v_1^i\ \Box]$ corresponds to (appR-i). (3) The evaluation context $E[\lambda x.\Box]$ corresponds to (lambda-(i+1)). (4) The evaluation context $E[!\Box]$ corresponds to (run-i). (5) The evaluation context $E[\langle\Box\rangle]$ corresponds to (code-i). (6) The evaluation context $E[\sim\Box]$ corresponds to (splice-i). (7) The evaluation context $E[\Box + t_2^i]$ corresponds to (plusL-i). (8) The evaluation context $E[v_1^i + \Box]$ corresponds to (plusR-i). To get familiar with

the above correspondences, consider the following example:

$$\dfrac{\dfrac{\dfrac{}{\sim\langle x\rangle \longrightarrow^1 x}\ \text{(splice-1)}}{\lambda x.\sim\langle x\rangle \longrightarrow^1 \lambda x.x}\ \text{(lambda-(i+1))}}{\langle\lambda x.\sim\langle x\rangle\rangle \longrightarrow^0 \langle\lambda x.x\rangle}\ \text{(code-i)}$$

The level-0 term $\langle\lambda x.\sim\langle x\rangle\rangle$ can be represented as $(\Box[\langle\Box\rangle][\lambda x.\Box])^{1\multimap 0}[\sim\langle x\rangle]$. The evaluation context $(\Box[\langle\Box\rangle][\lambda x.\Box])^{1\multimap 0}$ corresponds to the structural rules (code-i) and (lambda-(i+1)).

We can get the evaluation contexts that are suitable for a single-stage language such as ISWIM if we remove the rules that involve any multi-stage annotation and repeatedly apply the rules that define evaluation contexts with an inner level 0 and an outer level 0.

This definition is inside-out. An outside-in definition is provided in Definition 121.

4.1.2 Substitutional Reduction Semantics

We lay out the substitutional reduction semantics through a family of level-indexed notions of reduction $\mathscr{R}^i$, a family of level-indexed reduction relations $\longmapsto^i$ and a family of level-indexed multi-reduction relations $\longmapsto^{i*}$.

Definition 115 (Level-indexed Notions of Reduction)**.** For any $i \in \{0,1\}$, let the level-indexed notions of reduction $\mathscr{R}^i$ be a binary relation between the set of terms at level i and the set of terms at level i.

$$\mathscr{R}^i \subseteq \text{TERM}^i \times \text{TERM}^i$$

$$\begin{array}{rcll} (\lambda x.t^0)\ v^0 & \mathscr{R}^0 & t^0[v^0/x] & \text{(app-0)} \\ !\langle v^1\rangle & \mathscr{R}^0 & v^1 & \text{(run-0)} \\ \sim\langle v^1\rangle & \mathscr{R}^1 & v^1 & \text{(splice-1)} \\ n_1+n_2 & \mathscr{R}^0 & n \text{ where } n=n_1+n_2 & \text{(plus-0)} \end{array}$$

The notion of reduction $t_1^i\ \mathscr{R}^i\ t_2^i$ reads as "t_1 reduces to t_2 at level i". Each notion corresponds to one reduction rule of the single-step relations presented in Definition 36.

Definition 116 (Level-indexed Reduction Relations)**.** For any $i \in \mathbb{N}$, let the level-indexed reduction relation $\longmapsto^i$ be a binary relation between the set of terms and the set of terms directly based on the notions of reduction $\mathscr{R}^j$.

$$\longmapsto^i \subseteq \text{TERM}^i \times \text{TERM}^i$$

$$\dfrac{t_1^j\ \mathscr{R}^j\ t_2^j}{E^{j\multimap i}[t_1^j] \longmapsto^i E^{j\multimap i}[t_2^j]}$$

The reduction relation $t_1^i \longmapsto^i t_2^i$ reads as "t_1 single-reduces to t_2 at level i". The above definition states that the reduction relation respects performing any notion of reduction in an evaluation context.

Intuitively, a level-indexed reduction relation $\longmapsto^i$ defines a single step of computation at level i. We define the level-indexed multi-reduction relation $\longmapsto^{i*}$ to represent multiple (zero or more) steps of computation at level i.

$$
\begin{array}{lll}
 & !\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle & \\
= & (\Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0][\sim\Box])^{0\multimap 0}[(\lambda x.\langle x\rangle)\ (\lambda x.y)] & \\
\longmapsto^0 & (\Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0][\sim\Box])^{0\multimap 0}[\langle \lambda x.y\rangle] & \text{where } (\lambda x.\langle x\rangle)\ (\lambda x.y)\ \mathscr{R}^0\ \langle \lambda x.y\rangle \\
= & !\langle \lambda y.(\sim\langle \lambda x.y\rangle\ 0)\rangle & \\
= & (\Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0])^{1\multimap 0}[\sim\langle \lambda x.y\rangle] & \\
\longmapsto^0 & (\Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0])^{1\multimap 0}[\lambda x.y] & \text{where } \sim\langle \lambda x.y\rangle\ \mathscr{R}^1\ \lambda x.y \\
= & !\langle \lambda y.((\lambda x.y)\ 0)\rangle & \\
= & \Box^{0\multimap 0}[!\langle \lambda y.((\lambda x.y)\ 0)\rangle] & \\
\longmapsto^0 & \Box^{0\multimap 0}[\lambda y.((\lambda x.y)\ 0)] & \text{where } !\langle \lambda y.((\lambda x.y)\ 0)\rangle\ \mathscr{R}^0\ \lambda y.((\lambda x.y)\ 0) \\
= & \lambda y.((\lambda x.y)\ 0) &
\end{array}
$$

Figure 4.1: Evaluation of $!\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle$ in Substitutional Reduction Semantics of MetaML.

Definition 117 (Level-indexed Multi-reduction Relations)**.** For any $i \in \mathbb{N}$, let the level-indexed multi-reduction relation $\longmapsto^{i*}$ be the reflexive-transitive closure of the reduction relation $\longmapsto^i$.

The multi-reduction relation $t_1^i \longmapsto^{i*} t_2^i$ reads as "t_1 multi-reduces to t_2 at level i".

Example 118. Consider $!\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle$.

By the substitutional reduction semantics of MetaML, we have:

$$!\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle \longmapsto^{0*} \lambda y.((\lambda x.y)\ 0)$$

as demonstrated in Figure 4.1.

A comparison of Figures 4.1 and 3.1 tells that MetaML's substitutional reduction semantics follows the exact same three-step break-apply-plug pattern of evaluating a program as Environmental ISWIM's reduction semantics.

Evaluator We now define an evaluator in terms of the substitutional reduction semantics of MetaML. The evaluator is analogous to the evaluator defined in terms of the substitutional structural operational semantics of MetaML.

Definition 119 (Evaluator based on Substitutional Reduction Semantics of MetaML)**.** Let the evaluator $eval_{\text{MetaML:SubRed}}$ be a partial function from the set of programs $\text{PRGM}_{\text{MetaML}}$ to the set of answers $\text{ANS}_{\text{MetaML}}$.

$$eval_{\text{MetaML:SubRed}} : \text{PRGM}_{\text{MetaML}} \rightharpoonup \text{ANS}_{\text{MetaML}}$$

$$eval_{\text{MetaML:SubRed}}(t) = \begin{cases} \texttt{function} & \text{if } t \longmapsto^{0*} \lambda x.t'^0 \\ \texttt{code} & \text{if } t \longmapsto^{0*} \langle v^1\rangle \\ n & \text{if } t \longmapsto^{0*} n \end{cases}$$

This evaluator is defined in terms of the substitutional reduction semantics. The subscript "$_{\text{MetaML:SubRed}}$" in $eval_{\text{MetaML:SubRed}}$ denotes the substitutional reduction semantics of MetaML.

We claim that the evaluators defined in terms of the substitutional structural operational semantics and the substitutional reduction semantics of MetaML are equivalent.

Theorem 120 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{MetaML}}$*,* $eval_{\text{MetaML:SubSOS}}(t)$ *is Kleene equal to* $eval_{\text{MetaML:SubRed}}(t)$*.*

We prove the theorem in appendices.

4.2 MetaML - MK Abstract Machine

With a systematic strategy to search for an evaluation context and a redex, we can refine MetaML's substitutional reduction semantics to a corresponding substitutional abstract machine which we call the MK machine. This section mostly follows the path of refining Environmental ISWIM's reduction semantics to the CEK abstract machine as presented in Section 3.6.

4.2.1 Syntax

The definitions of terms, values and denotable terms are the same as Sections 2.2.1 and 4.1.

4.2.1.1 Evaluation Contexts

Evaluation contexts have been defined in Section 4.1 as inside-out. We provide an outside-in definition as follows. These two definitions are equivalent and are used interchangeably at our convenience.

Definition 121 (Level-indexed Evaluation Contexts: Outside-in)**.** Let $i, j \in \mathbb{N}$. Define $\text{ECXT}^{i-\circ j}$ to be the set of evaluation contexts with inner level i and outer level j.

$$\frac{}{\Box \in \text{ECXT}^{j-\circ j}} \text{ (ept-j)}$$

$$\frac{E \in \text{ECXT}^{i-\circ j}}{(E\ t_2^j) \in \text{ECXT}^{i-\circ j}} \text{ (appL-j)} \qquad \frac{E \in \text{ECXT}^{i-\circ j}}{(v_1^j\ E) \in \text{ECXT}^{i-\circ j}} \text{ (appR-j)}$$

$$\frac{E \in \text{ECXT}^{i-\circ (j+1)}}{\lambda x.E \in \text{ECXT}^{i-\circ (j+1)}} \text{ (lambda-(j+1))}$$

$$\frac{E \in \text{ECXT}^{i-\circ (j+1)}}{\langle E \rangle \in \text{ECXT}^{i-\circ j}} \text{ (code-j)} \qquad \frac{E \in \text{ECXT}^{i-\circ j}}{\sim E \in \text{ECXT}^{i-\circ (j+1)}} \text{ (splice-(j+1))} \qquad \frac{E \in \text{ECXT}^{i-\circ j}}{!E \in \text{ECXT}^{i-\circ j}} \text{ (run-j)}$$

$$\frac{E \in \text{ECXT}^{i-\circ j}}{(E + t_2^j) \in \text{ECXT}^{i-\circ j}} \text{ (plusL-j)} \qquad \frac{E \in \text{ECXT}^{i-\circ j}}{(v_1^j + E) \in \text{ECXT}^{i-\circ j}} \text{ (plusR-j)}$$

4.2.1.2 Machine Configurations

Section 3.6 defined the states of the CEK machine through four modes of machine configurations. We extend the four-mode definition to accommodate multiple stages.

Definition 122 (Machine Configurations)**.** Let CFG be the set of machine configurations

$$C \in \textsc{Cfg},\ t^i \in \textsc{Term}^i,\ v^i \in \textsc{Value}^i,\ E^{i\multimap j} \in \textsc{Ecxt}^{i\multimap j}$$

$$\begin{array}{rcl} C & ::= & v^0 \\ & & \mid \langle i,\ E^{i\multimap 0},\ t^i\rangle_{\mathrm{r}} \\ & & \mid \langle i,\ E^{i\multimap 0},\ t^i\rangle_{\mathrm{f}} \\ & & \mid \langle i,\ E^{i\multimap 0},\ v^i\rangle_{\mathrm{b}} \end{array}$$

The machine operates in four modes: the *value* mode v^0, the *reduce* mode $\langle i,\ E^{i\multimap 0},\ t^i\rangle_{\mathrm{r}}$, the *focus* mode $\langle i,\ E^{i\multimap 0},\ t^i\rangle_{\mathrm{f}}$, the *build* mode $\langle i,\ E^{i\multimap 0},\ v^i\rangle_{\mathrm{b}}$.

A machine configuration $\langle i,\ E^{i\multimap 0},\ t^i\rangle_?$ where $? \in \{\mathrm{r},\mathrm{f},\mathrm{b}\}$ unloads to the configuration $E^{i\multimap 0}[t^i]$.

4.2.2 MK Abstract Machine

We lay out the substitutional abstract machine, i.e., the MK machine, through the reduction relation $\longmapsto_{\mathrm{mk}}$ and the multi-reduction relation $\longmapsto^*_{\mathrm{mk}}$.

Definition 123 (Reduction Relation)**.** Let the reduction relation $\longmapsto_{\mathrm{mk}}$ be a binary relation between the set of machine configurations and the set of machine configurations.

$$\longmapsto_{\mathrm{mk}} \subseteq \textsc{Cfg} \times \textsc{Cfg}$$

Reduce rules: $\langle i,\ E^{i\multimap 0},\ t^i\rangle_{\mathrm{r}}$

$$\begin{array}{rcll} \langle 0,\ E,\ (\lambda x.t^0)\ v^0\rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mk}} & \langle 0,\ E,\ t^0[v^0/x]\rangle_{\mathrm{f}} & \text{(r-app-0)} \\ \langle 0,\ E,\ !\langle v^1\rangle\rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mk}} & \langle 0,\ E,\ v^1\rangle_{\mathrm{f}} & \text{(r-run-0)} \\ \langle 1,\ E,\ \sim\langle v^1\rangle\rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mk}} & \langle 1,\ E,\ v^1\rangle_{\mathrm{f}} & \text{(r-splice-1)} \\ \langle 0,\ E,\ n_1+n_2\rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mk}} & \langle 0,\ E,\ n\rangle_{\mathrm{f}} \text{ where } n = n_1+n_2 & \text{(r-plus-0)} \end{array}$$

Focus rules: $\langle i,\ E^{i\multimap 0},\ t^i\rangle_{\mathrm{f}}$

$$\begin{array}{rcll} \langle i+1,\ E,\ x\rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mk}} & \langle i+1,\ E,\ x\rangle_{\mathrm{b}} & \text{(f-var-(i+1))} \\ \langle i,\ E,\ t_1\ t_2\rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mk}} & \langle i,\ E[\Box\ t_2],\ t_1\rangle_{\mathrm{f}} & \text{(f-appL-i)} \\ \langle 0,\ E,\ \lambda x.t\rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mk}} & \langle 0,\ E,\ \lambda x.t\rangle_{\mathrm{b}} & \text{(f-lambda-0)} \\ \langle i+1,\ E,\ \lambda x.t\rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mk}} & \langle i+1,\ E[\lambda x.\Box],\ t\rangle_{\mathrm{f}} & \text{(f-lambda-(i+1))} \\ \langle i,\ E,\ \langle t\rangle\rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mk}} & \langle i+1,\ E[\langle\Box\rangle],\ t\rangle_{\mathrm{f}} & \text{(f-code-i)} \\ \langle i+1,\ E,\ \sim t\rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mk}} & \langle i,\ E[\sim\Box],\ t\rangle_{\mathrm{f}} & \text{(f-splice-(i+1))} \\ \langle i,\ E,\ !t\rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mk}} & \langle i,\ E[!\Box],\ t\rangle_{\mathrm{f}} & \text{(f-run-i)} \\ \langle i,\ E,\ n\rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mk}} & \langle i,\ E,\ n\rangle_{\mathrm{b}} & \text{(f-num-i)} \\ \langle i,\ E,\ t_1+t_2\rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mk}} & \langle i,\ E[\Box+t_2],\ t_1\rangle_{\mathrm{f}} & \text{(f-plusL-i)} \end{array}$$

Build rules: $\langle i,\ E^{i\multimap 0},\ v^i\rangle_{\mathrm{b}}$

$$
\begin{array}{rcll}
\langle 0,\, \Box,\, v\rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mk}} & v & \text{(b-value-0)} \\
\langle i,\, E[\Box\, t_2],\, v_1\rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mk}} & \langle i,\, E[v_1\, \Box],\, t_2\rangle_{\mathrm{f}} & \text{(b-appL-i)} \\
\langle 0,\, E[v_1\, \Box],\, v_2\rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mk}} & \langle 0,\, E,\, v_1\, v_2\rangle_{\mathrm{r}} & \text{(b-appR-0)} \\
\langle i+1,\, E[v_1\, \Box],\, v_2\rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mk}} & \langle i+1,\, E,\, v_1\, v_2\rangle_{\mathrm{b}} & \text{(b-appR-(i+1))} \\
\langle i+1,\, E[\lambda x.\Box],\, v\rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mk}} & \langle i+1,\, E,\, \lambda x.v\rangle_{\mathrm{b}} & \text{(b-lambda-(i+1))} \\
\langle i+1,\, E[\langle\Box\rangle],\, v\rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mk}} & \langle i,\, E,\, \langle v\rangle\rangle_{\mathrm{b}} & \text{(b-code-(i+1))} \\
\langle 0,\, E[\sim\Box],\, v\rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mk}} & \langle 1,\, E,\, \sim v\rangle_{\mathrm{r}} & \text{(b-splice-0)} \\
\langle i+1,\, E[\sim\Box],\, v\rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mk}} & \langle i+2,\, E,\, \sim v\rangle_{\mathrm{b}} & \text{(b-splice-(i+1))} \\
\langle 0,\, E[!\Box],\, v\rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mk}} & \langle 0,\, E,\, !v\rangle_{\mathrm{r}} & \text{(b-run-0)} \\
\langle i+1,\, E[!\Box],\, v\rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mk}} & \langle i+1,\, E,\, !v\rangle_{\mathrm{b}} & \text{(b-run-(i+1))} \\
\langle i,\, E[\Box+t_2],\, v_1\rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mk}} & \langle i,\, E[v_1+\Box],\, t_2\rangle_{\mathrm{f}} & \text{(b-plusL-i)} \\
\langle 0,\, E[v_1+\Box],\, v_2\rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mk}} & \langle 0,\, E,\, v_1+v_2\rangle_{\mathrm{r}} & \text{(b-plusR-0)} \\
\langle i+1,\, E[v_1+\Box],\, v_2\rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mk}} & \langle i+1,\, E,\, v_1+v_2\rangle_{\mathrm{b}} & \text{(b-plusR-(i+1))}
\end{array}
$$

The reduction relation $C_1 \longmapsto_{\mathrm{mk}} C_2$ reads as "C_1 single-reduces to C_2".

The intuition behind the above relation is analogous to that of CEK machine's reduction relation. See comments below Definition 108.

Intuitively, the reduction relation $\longmapsto_{\mathrm{mk}}$ defines a single step of computation. We define $\longmapsto^*_{\mathrm{mk}}$ to represent multiple (zero or more) steps of computation.

Definition 124 (Multi-reduction Relation)**.** Let the multi-reduction relation $\longmapsto^*_{\mathrm{mk}}$ be the reflexive-transitive closure of the reduction relation $\longmapsto_{\mathrm{mk}}$.

The multi-transformation relation $C_1 \longmapsto^*_{\mathrm{mk}} C_2$ reads as "C_1 multi-reduces to C_2".

The abstract machine defined above is also known as the MK machine. M stands for multi-stage and K stands for continuation, i.e., the evaluation context.

Evaluator. We now define an evaluator in terms of the MK machine. The MK machine's multi-reduction relation is defined on machine configurations. Given a program t, the evaluator applies the multi-reduction relation on the machine configuration $\langle 0,\, \Box,\, t\rangle_{\mathrm{f}}$ in which the program is associated with an empty evaluation context and is evaluated at level 0. The evaluator is otherwise analogous to the one defined in terms of the substitution reduction semantics of MetaML.

Definition 125 (Evaluator based on MK Machine)**.** Let the evaluator $eval_{\mathrm{MetaML:MK}}$ be a partial function from the set of programs $\textsc{Prgm}_{\mathrm{MetaML}}$ to the set of answers $\textsc{Ans}_{\mathrm{MetaML}}$.

$$
eval_{\mathrm{MetaML:MK}} : \textsc{Prgm}_{\mathrm{MetaML}} \rightharpoonup \textsc{Ans}_{\mathrm{MetaML}}
$$

$$
eval_{\mathrm{MetaML:MK}}(t) = \begin{cases} \texttt{function} & \text{if } \langle 0,\, \Box,\, t\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} \lambda x.t'^{0} \\ \texttt{code} & \text{if } \langle 0,\, \Box,\, t\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} \langle v^1\rangle \\ n & \text{if } \langle 0,\, \Box,\, t\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} n \end{cases}
$$

$$
\begin{array}{lll}
 & \langle 0, & \Box, \\
\longmapsto_{\mathrm{mk}} & \langle 0, & \Box[!\Box], \\
\longmapsto_{\mathrm{mk}} & \langle 1, & \Box[!\Box][\langle\Box\rangle], \\
\longmapsto_{\mathrm{mk}} & \langle 1, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box], \\
\longmapsto_{\mathrm{mk}} & \langle 1, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0], \\
\longmapsto_{\mathrm{mk}} & \langle 0, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0][\sim\Box], \\
\longmapsto_{\mathrm{mk}} & \langle 0, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0][\sim\Box][\Box\ (\lambda x.y)], \\
\longmapsto_{\mathrm{mk}} & \langle 0, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0][\sim\Box][\Box\ (\lambda x.y)], \\
\longmapsto_{\mathrm{mk}} & \langle 0, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0][\sim\Box][(\lambda x.\langle x\rangle)\ \Box], \\
\longmapsto_{\mathrm{mk}} & \langle 0, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0][\sim\Box][(\lambda x.\langle x\rangle)\ \Box], \\
\longmapsto_{\mathrm{mk}} & \langle 0, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0][\sim\Box], \\
\longmapsto_{\mathrm{mk}} & \langle 0, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0][\sim\Box], \\
\longmapsto_{\mathrm{mk}} & \langle 0, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0][\sim\Box], \\
\longmapsto_{\mathrm{mk}} & \langle 1, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0][\sim\Box][\langle\Box\rangle], \\
\longmapsto_{\mathrm{mk}} & \langle 1, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0][\sim\Box][\langle\Box\rangle], \\
\longmapsto_{\mathrm{mk}} & \langle 0, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0][\sim\Box], \\
\longmapsto_{\mathrm{mk}} & \langle 1, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0], \\
\longmapsto_{\mathrm{mk}} & \langle 1, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0], \\
\longmapsto_{\mathrm{mk}} & \langle 1, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][\Box\ 0], \\
\longmapsto_{\mathrm{mk}} & \langle 1, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][(\lambda x.y)\ \Box], \\
\longmapsto_{\mathrm{mk}} & \langle 1, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box][(\lambda x.y)\ \Box], \\
\longmapsto_{\mathrm{mk}} & \langle 1, & \Box[!\Box][\langle\Box\rangle][\lambda y.\Box], \\
\longmapsto_{\mathrm{mk}} & \langle 1, & \Box[!\Box][\langle\Box\rangle], \\
\longmapsto_{\mathrm{mk}} & \langle 0, & \Box[!\Box], \\
\longmapsto_{\mathrm{mk}} & \langle 0, & \Box, \\
\longmapsto_{\mathrm{mk}} & \langle 0, & \Box, \\
\longmapsto_{\mathrm{mk}} & \langle 0, & \Box, \\
\longmapsto_{\mathrm{mk}} & \lambda y.((\lambda x.y)\ 0) &
\end{array}
\qquad
\begin{array}{l}
!\langle\lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle\rangle_{\mathrm{f}} \\
\langle\lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle\rangle_{\mathrm{f}} \\
\lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle_{\mathrm{f}} \\
\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0\rangle_{\mathrm{f}} \\
\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\rangle_{\mathrm{f}} \\
(\lambda x.\langle x\rangle)\ (\lambda x.y)\rangle_{\mathrm{f}} \\
\lambda x.\langle x\rangle\rangle_{\mathrm{f}} \\
\lambda x.\langle x\rangle\rangle_{\mathrm{b}} \\
\lambda x.y\rangle_{\mathrm{f}} \\
\lambda x.y\rangle_{\mathrm{b}} \\
(\lambda x.\langle x\rangle)\ (\lambda x.y)\rangle_{\mathrm{r}} \\
\langle\lambda x.y\rangle\rangle_{\mathrm{r}} \\
\langle\lambda x.y\rangle\rangle_{\mathrm{f}} \\
\lambda x.y\rangle_{\mathrm{f}} \\
\lambda x.y\rangle_{\mathrm{b}} \\
\langle\lambda x.y\rangle\rangle_{\mathrm{b}} \\
\sim\langle\lambda x.y\rangle\rangle_{\mathrm{r}} \\
\lambda x.y\rangle_{\mathrm{f}} \\
\lambda x.y\rangle_{\mathrm{b}} \\
0\rangle_{\mathrm{f}} \\
0\rangle_{\mathrm{b}} \\
(\lambda x.y)\ 0\rangle_{\mathrm{b}} \\
\lambda y.((\lambda x.y)\ 0)\rangle_{\mathrm{b}} \\
\langle\lambda y.((\lambda x.y)\ 0)\rangle\rangle_{\mathrm{b}} \\
!\langle\lambda y.((\lambda x.y)\ 0)\rangle\rangle_{\mathrm{r}} \\
\lambda y.((\lambda x.y)\ 0)\rangle_{\mathrm{f}} \\
\lambda y.((\lambda x.y)\ 0)\rangle_{\mathrm{b}}
\end{array}
$$

Figure 4.2: Evaluation of $!\langle\lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle$ in the MK Machine.

This evaluator is defined in terms of the MK machine. The subscript "$_{\text{MetaML:MK}}$" in $eval_{\text{MetaML:MK}}$ denotes the MK machine of MetaML.

Example 126. Consider $!\langle\lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle$.

We have:

$$eval_{\text{MetaML:MK}}(!\langle\lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle) = \texttt{function}$$

This is because

$$\langle 0,\ \Box,\ !\langle\lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle\rangle_{\mathrm{f}} \longmapsto^{*}_{\mathrm{mk}} \lambda y.((\lambda x.y)\ 0)$$

as demonstrated in Figure 4.2.

The above example has appeared as Example 118. By MetaML's substitutional reduction semantics, we have:

$$eval_{\text{MetaML:SubRed}}(!\langle\lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle) = \texttt{function}$$

The evaluators defined in terms of the substitutional reduction semantics of MetaML and the MK machine

agree on evaluating the the evaluation of the puzzle program $!\langle \lambda a. \sim ((\lambda x.\langle x \rangle)\,(\lambda x.\langle a \rangle))\,0 \rangle$. In fact, these two evaluators agree on all programs.

Theorem 127 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{MetaML}}$*,* $eval_{\text{MetaML:SubRed}}(t)$ *is Kleene equal to* $eval_{\text{MetaML:MK}}(t)$*.*

We prove the above theorem in appendices.

As a corollary, the evaluators defined in terms of the substitutional structural operational semantics of MetaML and based on the MK machine are equivalent.

Corollary 128 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{MetaML}}$*,* $eval_{\text{MetaML:SubSOS}}(t)$ *is Kleene equal to* $eval_{\text{MetaML:MK}}(t)$*.*

Proof. It immediately follows from Theorems 120 and 127 by the transitivity of Kleene equality. □

4.3 Chapter Summary

Following the second dimension of the semantics refinement problem, this chapter solved the following problem that is less complicated than the main semantics refinement problem.

> Can we refine the substitutional structural operational semantics of MetaML to a corresponding substitutional abstract machine, which we call the MK machine, and demonstrate their equivalence?

Given the substitutional structural operational semantics of MetaML, as a manageable step towards developing the MK machine, we developed an equivalent substitutional reduction semantics. Then based on the substitutional reduction semantics, we developed an equivalent substitutional abstract machine, the MK machine.

We defined an evaluator for each semantics. By proving the equivalence of every two adjacent semantics, we finally showed that the MK machine is equivalent to the substitutional structural operational semantics of MetaML.

Chapter 5

Refining Semantics for MetaML: Developing the MEK Machine

Developing an environmental abstract machine for the multi-stage language MetaML is not straightforward. On the one hand, even for a single-stage language such as ISWIM, it is challenging to refine a substitutional structural operational semantics to a corresponding environmental abstract machine. On the other hand, it is uneasy to refine semantics for a multi-stage language such as MetaML even if substitutions are not replaced with environments.

Chapter 3 studies how to develop an environmental abstract machine for the single-stage language ISWIM. We split the problem into two subproblems. The first subproblem is to stepwise refine the known substitutional structural operational semantics to a corresponding environmental structural operational semantics. The crucial points of our approach include replacing meta-language substitutions by explicit substitutions, modelling how an explicit substitution percolates through a term at the semantical level, delaying explicit substitutions outside lambda abstractions and replacing cascaded explicit substitutions by environments. The second subproblem is to stepwise refine the environmental structural operational semantics to a corresponding environmental abstract machine. The key to our approach is relating these two semantics by an environmental reduction semantics because a reduction semantics is a concise representation of an abstract machine. At this point, we are unclear whether our approach to refining semantics for ISWIM is applicable to MetaML.

To understand how refining semantics for MetaML is different from ISWIM and whether our approach to refining semantics for ISWIM is applicable to MetaML, Chapter 4 studies how to develop a substitutional abstract machine for the multi-stage language MetaML. This problem is analogous to the second subproblem of refining semantics for ISWIM. Taking the same approach, we successfully derive a substitutional abstract machine for MetaML. After Chapter 4, we are now familiar with refining semantics for MetaML and we believe refining semantics for MetaML is analogous to refining semantics for ISWIM.

Utilising the experience of solving two less complicated semantics refinement problems in Chapters 3 and 4, we concentrate on the main semantics refinement problem in this chapter. We study how to stepwise develop an environmental abstract machine for the multi-stage language MetaML. The problem is restated as follows.

> Can we refine the substitutional structural operational semantics of MetaML to a corresponding environmental abstract machine, which we call the MEK machine, and demonstrate their equivalence?

Taking the approach to refining semantics in Chapter 3, we propose to first stepwise derive an environmental

structural operational semantics for MetaML and then refine the semantics to an environmental abstract machine. We also draw lessons from Chapter 4 for refining a structural operational semantics to an abstract machine for MetaML.

5.1 MetaML

For convenience, this section revisits the substitutional structural operational semantics of MetaML, which has been intensively studied in Section 2.2.

Terms, Values and Denotable Terms. Terms, values and denotable terms have been defined in Section 2.2 using inductive rules. We present an equivalent definition in Backus Naur Form (BNF) as follows.

Definition 129 (Terms, Values and Denotable Terms)**.** For any $i \in \mathbb{N}$, let TERM^i be the set of level-indexed terms at level i and VALUE^i be the set of level-indexed values at level i. Let DENOTABLE be the set of denotable terms.

$$x \in \text{VAR},\ i,n \in \mathbb{N},\ t^i \in \text{TERM}^i,\ v^i \in \text{VALUE}^i,\ w \in \text{DENOTABLE}$$

$$\begin{array}{lcl}
t^0 & ::= & x \mid t^0\ t^0 \mid \lambda x.t^0 \mid \langle t^1 \rangle \mid !t^0 \mid n \mid t^0 + t^0 \\
t^{i+1} & ::= & x \mid t^{i+1}\ t^{i+1} \mid \lambda x.t^{i+1} \mid \langle t^{i+2} \rangle \mid {\sim}t^i \mid !t^{i+1} \mid n \mid t^{i+1} + t^{i+1} \\
\\
v^0 & ::= & \lambda x.t^0 \mid \langle v^1 \rangle \mid n \\
v^1 & ::= & x \mid v^1\ v^1 \mid \lambda x.v^1 \mid \langle v^2 \rangle \mid !v^1 \mid n \mid v^1 + v^1 \\
v^{i+2} & ::= & x \mid v^{i+2}\ v^{i+2} \mid \lambda x.v^{i+2} \mid \langle v^{i+3} \rangle \mid {\sim}v^{i+1} \mid !v^{i+2} \mid n \mid v^{i+2} + v^{i+2} \\
\\
w & ::= & x \mid v^0
\end{array}$$

Given how concise the above BNF definition is, we can more clearly present the syntax changes between different dialects of MetaML.

Alpha Equivalence Relation. Section 3.1.1.4 presents the alpha equivalence for the single-language ISWIM and justifies why such a relation is necessary for proving equivalence of semantics of different dialects of ISWIM or MetaML. We extend the definition of alpha equivalence for ISWIM (Definition 50) to accommodate the multi-stage setting.

Definition 130 (Alpha Equivalence Relation)**.** Let the alpha equivalence relation $\sim_\alpha$ be a binary relation on terms.

$$\sim_\alpha \subseteq \text{TERM} \times \text{TERM}$$

$$\frac{}{x \sim_\alpha x}\ \text{(var)} \qquad \frac{t_{11} \sim_\alpha t_{21} \quad t_{12} \sim_\alpha t_{22}}{(t_{11}\ t_{12}) \sim_\alpha (t_{21}\ t_{22})}\ \text{(app)}$$

$$\frac{t_1[x_3/x_1] \sim_\alpha t_2[x_3/x_2]}{(\lambda x_1.t_1) \sim_\alpha (\lambda x_2.t_2)}\ \text{where } x_3 \notin FV(t_1) \cup FV(t_2)\ \text{(lam)}$$

$$\frac{t_1 \sim_\alpha t_2}{\langle t_1 \rangle \sim_\alpha \langle t_2 \rangle}\ \text{(code)} \qquad \frac{t_1 \sim_\alpha t_2}{\sim t_1 \sim_\alpha \sim t_2}\ \text{(splice)} \qquad \frac{t_1 \sim_\alpha t_2}{!t_1 \sim_\alpha !t_2}\ \text{(run)}$$

$$\frac{t_{11} \sim_\alpha t_{21} \quad t_{12} \sim_\alpha t_{22}}{(t_{11} + t_{12}) \sim_\alpha (t_{21} + t_{22})}\ \text{(plus)} \qquad \frac{}{n \sim_\alpha n}\ \text{(num)}$$

The alpha equivalence relation makes proving equivalence of semantics easier. In many cases, we may replace a term by its alpha equivalent term in a proof at our convenience.

Properties. We observe the following properties that are useful in proving semantics equivalence. (1) The single-step relation preserves alpha equivalence. (2) The multi-step relation preserves the closedness of a term. They are the same properties that Substitutional ISWIM holds.

Proposition 131. *If $t_1^i \sim_\alpha t_2^i$ and $t_1^i \longrightarrow^i t_{11}^i$, then $t_2^i \longrightarrow^i t_{21}^i$ and $t_{11}^i \sim_\alpha t_{21}^i$.*

Proposition 132. *If $FV(t_1^i) = \emptyset$ and $t_1^i \longrightarrow^{*i} t_2^i$, then $FV(t_2^i) = \emptyset$.*

5.2 Explicit MetaML

Following the path of refining Substitutional ISWIM to Explicit ISWIM in Section 3.2, we refine Substitutional MetaML to Explicit MetaML. Explicit MetaML replaces Substitutional MetaML's meta-language substitutions (e.g., $t[w/x]$) by explicit substitutions (e.g., $t[x := w]$) and models how an explicit substitution percolates through a term at the semantical level.

5.2.1 Syntax

We first define the basic syntax of Explicit MetaML: source terms, runtime terms, values and denotable terms. Then we define the free variable function, the substitution function and the alpha equivalence relation.

5.2.1.1 Source Terms, Runtime Terms, Values and Denotable Terms

Definition 133 (Source Terms, Runtime Terms, Values and Denotable Terms)**.** For any $i \in \mathbb{N}$, let STERM^i be the set of level-indexed source terms at level i, RTERM^i be the set of level-indexed runtime terms at level i and VALUE^i be the set of level-indexed values at level i. Let DENOTABLE be the set of denotable terms.

$x \in \text{VAR},\ i,n \in \mathbb{N},\ t_{\text{s}}^i \in \text{STERM}^i,\ t^i \in \text{RTERM}^i,\ v^i \in \text{VALUE}^i,\ w \in \text{DENOTABLE}$

$$
\begin{array}{lcl}
t_{\text{s}}^0 & := & x \mid t_{\text{s}}^0\ t_{\text{s}}^0 \mid \lambda x.t_{\text{s}}^0 \mid \langle t_{\text{s}}^1 \rangle \mid !t_{\text{s}}^0 \mid n \mid t_{\text{s}}^0 + t_{\text{s}}^0 \\
t_{\text{s}}^{i+1} & := & x \mid t_{\text{s}}^{i+1}\ t_{\text{s}}^{i+1} \mid \lambda x.t_{\text{s}}^{i+1} \mid \langle t_{\text{s}}^{i+2} \rangle \mid {\sim}t_{\text{s}}^i \mid !t_{\text{s}}^{i+1} \mid n \mid t_{\text{s}}^{i+1} + t_{\text{s}}^{i+1} \\
 & & \\
t^0 & := & x \mid t^0\ t^0 \mid \lambda x.t^0 \mid \langle t^1 \rangle \mid !t^0 \mid n \mid t^0 + t^0 \mid \boxed{\underline{\lambda} x.t^0} \mid \boxed{t^0[x := w]} \\
t^{i+1} & := & x \mid t^{i+1}\ t^{i+1} \mid \lambda x.t^{i+1} \mid \langle t^{i+2} \rangle \mid {\sim}t^i \mid !t^{i+1} \mid n \mid t^{i+1} + t^{i+1} \mid \boxed{\underline{\lambda} x.t^0} \mid \boxed{t^i[x := w]} \\
 & & \\
v^0 & := & \langle v^1 \rangle \mid n \mid \boxed{\underline{\lambda} x.t^0} \\
v^1 & := & x \mid v^1\ v^1 \mid \lambda x.v^1 \mid \langle v^2 \rangle \mid !v^1 \mid n \mid v^1 + v^1 \mid \boxed{\underline{\lambda} x.t^0} \\
v^{i+2} & := & x \mid v^{i+2}\ v^{i+2} \mid \lambda x.v^{i+2} \mid \langle v^{i+3} \rangle \mid {\sim}v^{i+1} \mid !v^{i+2} \mid n \mid v^{i+2} + v^{i+2} \mid \boxed{\underline{\lambda} x.t^0} \\
 & & \\
w & := & x \mid v^0
\end{array}
$$

The set of source terms of Explicit MetaML is the same as the set of terms of Substitutional MetaML. Explicit MetaML has been enhanced with an explicit substitution $t^i[x := w]$, which means that each free occurrence of the variable x in the term t^i *needs* to be substituted by the denotable term w. Evaluating an explicit substitution takes steps.

To make Explicit MetaML be consistent with Substitutional MetaML, we must ensure that a value at one level must be a value at any higher level. To preserve this property in Explicit MetaML, we introduce an underlined lambda abstraction, $\underline{\lambda} x.t^0$. To understand the necessity of this change of syntax, consider the example of substituting a lambda abstraction for a variable where the variable is at a level higher than 0. Since a lambda abstraction acts as a denotable term, it must be a level-0 value. When the substitution is performed, the lambda abstraction is at a level higher than 0. We only perform substitution reduction for the body of a lambda abstraction at a level higher than 0. Although the lambda abstraction is a level-0 value, it may not be a value at a level higher than 0. We explain the reason in more detail through Example 142 after presenting the semantics.

Explicit MetaML differs from Substitutional MetaML in what terms count as values. A (conventional) lambda abstraction $\lambda x.t^0$ is no longer a value at level 0. An underlined lambda abstraction $\underline{\lambda} x.t^0$ is a value at any level. We usually call $\lambda x.v^{i+1}$ a level-$(i+1)$ lambda value and call $\underline{\lambda} x.t^0$ a level-0 lambda value.

5.2.1.2 Free Variable Function

We define the free variable function by extending Substitutional MetaML's Definition 24 to accommodate underlined lambda abstractions and explicit substitutions.

Definition 134 (Free Variable Function)**.** Let the free variable function FV be a total function from the set of runtime terms to the power set of variables.

$$FV \ : \ \text{RTERM} \longrightarrow \mathscr{P}(\text{VAR})$$

$$\begin{array}{rcll} \vdots & \vdots & \vdots & \vdots \\ FV(\underline{\lambda}x.t) & = & FV(\lambda x.t) & (10) \\ FV(t[x := w]) & = & (FV(t) \backslash \{x\}) \cup FV(w) & (11) \end{array}$$

Equations (1)-(9) are the same as Definition 20 in Substitutional MetaML. Equation (10) is trivial. Equation (11) is analogous to Equation (6) of Explicit ISWIM's free variable function (Definition 64).

5.2.1.3 Substitution Function

We define the substitution function by extending Substitutional MetaML's Definition 24 to accommodate underlined lambda abstractions and explicit substitutions.

Definition 135 (Substitution Function)**.** Let the substitution function $\cdot[\cdot/\cdot]$ be a partial function from the 3-tuple of the set of runtime terms, the set of denotable terms and the set of variables, to the set of runtime terms.

$$\cdot[\cdot/\cdot] \ : \ (\text{RTERM} \times \text{DENOTABLE} \times \text{VAR}) \rightarrow \text{RTERM}$$

$$\begin{array}{rcll} \vdots & \vdots & \vdots & \vdots \\ (\underline{\lambda}x_1.t_0)[w/x_2] & = & \underline{\lambda}x_3.t_0[x_3/x_1][w/x_2] & \\ & & \text{where } x_3 \notin FV(\underline{\lambda}x_1.t_0) \cup FV(w) \cup \{x_2\} & (10) \\ (t_0[x_1 := w_1])[w_2/x_2] & = & t_0[x_3/x_1][w_2/x_2][x_3 := w_1[w_2/x_2]] & \\ & & \text{where } x_3 \notin FV(\lambda x_1.t_0) \cup FV(w) \cup \{x_2\} & (11) \end{array}$$

Equations (1)-(9) are the same as Definition 24 in Substitutional MetaML. Equation (10) is analogous to Equation (4). Equation (11) is analogous to Equation (6) of Explicit ISWIM's Definition 65.

5.2.1.4 Alpha Equivalence Relation

We define the alpha equivalence relation by extending Substitutional MetaML's Definition 130 to accommodate underlined lambda abstractions and explicit substitutions.

Definition 136 (Alpha Equivalence Relation)**.** Let the alpha equivalence relation $\sim_\alpha$ be a binary relation on runtime terms.

$$\sim_\alpha \subseteq \text{RTERM} \times \text{RTERM}$$

$$\vdots$$

$$\frac{t_1[x_3/x_1] \sim_\alpha t_2[x_3/x_2]}{(\underline{\lambda} x_1.t_1) \sim_\alpha (\underline{\lambda} x_2.t_2)} \text{ where } x_3 \notin FV(t_1) \cup FV(t_2) \text{ (lamu)}$$

$$\frac{w_1 \sim_\alpha w_2 \quad t_1[x_3/x_1] \sim_\alpha t_2[x_3/x_2]}{(t_1[x_1 := w_1]) \sim_\alpha (t_2[x_2 := w_2])} \text{ where } x_3 \notin FV(t_1) \cup FV(t_2) \text{ (sub)}$$

All rules except the (sub) rule are the same as Definition 130 in Substitutional MetaML. The (lamu) rule is analogous to the (lam) rule. The (sub) rule is analogous to the (sub) rule of Explicit ISWIM's Definition 66.

5.2.2 Structural Operational Semantics

We lay out the structural operational semantics of Explicit MetaML through the level-indexed single-step relations $\longrightarrow^i$, the level-indexed single-step substitution reduction relations $\longrightarrow^{\text{x}i}$, the level-indexed multi-step substitution reduction relations $\longrightarrow^{\text{x}i*}$ and the level-indexed multi-step relations $\longrightarrow^{i*}$.

Definition 137 (Level-indexed Single-step Relations)**.** For any $i \in \mathbb{N}$, let the level-indexed single-step relation $\longrightarrow^i$ be a binary relation between the set of runtime terms at level i and the set of runtime terms at level i.

$$\longrightarrow^i \subseteq \text{RTERM}^i \times \text{RTERM}^i$$

$$\frac{}{\lambda x.t^0 \longrightarrow^0 \underline{\lambda} x.t^0} \text{ (lambda-0)} \qquad \frac{t_1^{i+1} \longrightarrow^{i+1} t_2^{i+1}}{\lambda x.t_1^{i+1} \longrightarrow^{i+1} \lambda x.t_2^{i+1}} \text{ (lambda-(i+1))}$$

$$\frac{t_{11}^i \longrightarrow^i t_{12}^i}{t_{11}^i\ t_2^i \longrightarrow^i t_{12}^i\ t_2^i} \text{ (appL-i)} \qquad \frac{t_{21}^i \longrightarrow^i t_{22}^i}{v_1^i\ t_{21}^i \longrightarrow^i v_1^i\ t_{22}^i} \text{ (appR-i)}$$

$$\frac{}{(\underline{\lambda} x.t^0)\ v^0 \longrightarrow^0 t^0[x := v^0]} \text{ (app-0)}$$

$$\frac{t_1^i \longrightarrow^i t_2^i}{!t_1^i \longrightarrow^i !t_2^i} \text{ (run-i)} \qquad \frac{}{!\langle v^1 \rangle \longrightarrow^0 v^1} \text{ (run-0)}$$

$$\frac{t_1^{i+1} \longrightarrow^{i+1} t_2^{i+1}}{\langle t_1^{i+1} \rangle \longrightarrow^i \langle t_2^{i+1} \rangle} \text{ (code-i)}$$

$$\frac{t_1^i \longrightarrow^i t_2^i}{\sim t_1^i \longrightarrow^{i+1} \sim t_2^i} \text{ (splice-(i+1))} \qquad \frac{}{\sim\langle v^1 \rangle \longrightarrow^1 v^1} \text{ (splice-1)}$$

$$\frac{t_{11}^i \longrightarrow^i t_{12}^i}{t_{11}^i + t_2^i \longrightarrow^i t_{12}^i + t_2^i}\ \text{(plusL-i)} \qquad \frac{t_{21}^i \longrightarrow^i t_{22}^i}{v_1^i + t_{21}^i \longrightarrow^i v_1^i + t_{22}^i}\ \text{(plusR-i)}$$

$$\frac{}{n_1 + n_2 \longrightarrow^0 n}\ \text{where } n = n_1 + n_2\ \text{(plus-0)}$$

The (app-0) rule is different from Substitutional MetaML and is analogous to the (app) rule of Explicit ISWIM's Definition 67. The (lambda-0) rule is new, corresponding to the definition of level-0 values. There is no equivalent of the (lambda-(i+1)) rule for underlined lambda abstractions because an underlined lambda abstraction is a value at any level.

The definition of the level-indexed single-step relations is currently incomplete because the percolation of explicit substitutions has not been defined yet.

To show how explicit substitutions percolate, we define new relations $t^i[x := w] \longrightarrow^{\mathrm{x}i} t^i$ for substitution reductions. The new relations ensure that explicit substitutions percolate deterministically.

Definition 138 (Level-indexed Single-step Substitution Reduction Relations)**.** For any $i \in \mathbb{N}$, let the level-indexed single-step substitution reduction relation $\longrightarrow^{\mathrm{x}i}$ be a binary relation between the set of runtime terms at level i and the set of runtime terms at level i.

$$\longrightarrow^{\mathrm{x}i} \subseteq \mathrm{RTERM}^i \times \mathrm{RTERM}^i$$

$$\frac{}{x[x := w] \longrightarrow^{\mathrm{x}i} w}\ \text{(var-eq-subst)} \qquad \frac{}{x_1[x_2 := w] \longrightarrow^{\mathrm{x}i} x_1}\ \text{where } x_1 \not\equiv x_2\ \text{(var-df-subst)}$$

$$\frac{}{n[x := w] \longrightarrow^{\mathrm{x}i} n}\ \text{(num-subst)}$$

$$\frac{}{(t_1^i\ t_2^i)[x := w] \longrightarrow^{\mathrm{x}i} (t_1^i[x := w])\ (t_2^i[x := w])}\ \text{(app-subst)}$$

$$\frac{}{(t_1^i + t_2^i)[x := w] \longrightarrow^{\mathrm{x}i} (t_1^i[x := w]) + (t_2^i[x := w])}\ \text{(plus-subst)}$$

$$\frac{}{(\lambda x_1.t^i)[x_2 := w] \longrightarrow^{\mathrm{x}i} \lambda x_3.t^i[x_1 := x_3][x_2 := w]}\ \text{where } x_3 \notin FV(\lambda x_1.t^i) \cup FV(w) \cup \{x_2\}\ \text{(lam-subst)}$$

$$\frac{}{(\underline{\lambda} x_1.t^0)[x_2 := w] \longrightarrow^{\mathrm{x}i} \underline{\lambda} x_3.t^0[x_1 := x_3][x_2 := w]}\ \text{where } x_3 \notin FV(\underline{\lambda} x_1.t^0) \cup FV(w) \cup \{x_2\}\ \text{(lamu-subst)}$$

$$\frac{}{\langle t^{i+1} \rangle[x := w] \longrightarrow^{\mathrm{x}i} \langle t^{i+1}[x := w] \rangle}\ \text{(code-subst)}$$

$$\frac{}{(!t^i)[x := w] \longrightarrow^{\mathrm{x}i} !t^i[x := w]}\ \text{(run-subst)}$$

$$\frac{}{(\sim t^i)[x := w] \longrightarrow^{\mathrm{x}(i+1)} \sim t^i[x := w]}\ \text{(splice-subst)}$$

$$\frac{t_1^i[x_1 := w_1] \longrightarrow^{\mathrm{x}i} t_2^i}{t_1^i[x_1 := w_1][x_2 := w_2] \longrightarrow^{\mathrm{x}i} t_2^i[x_2 := w_2]}\ \text{(subst-subst)}$$

Most rules describe how explicit substitutions behave when encountering other terms in the language. Every rule except the (lamu-subst) rule and the (subst-subst) rule correspond to an equation of Substitutional MetaML's substitution function (Definition 24). The (subst-subst) rule is analogous to the (subst-subst) rule of Explicit ISWIM's single-step substitution reduction relation, implying that only a single-step of substitution reduction may happen underneath an explicit substitution. The (lamu-subst) rule accommodates the newly invented underlined lambda abstraction and is analogous to the (lam-subst) rule.

Every single-step substitution reduction counts as a single step of computation. Thus we add the following (inj-subst) rule to the definition of the level-indexed single-step relations, Definition 138.

$$\frac{}{t_1^i \longrightarrow^i t_2^i} \text{ where } t_1^i \longrightarrow^{\mathrm{x}i} t_2^i \text{ (inj-subst)}$$

Definition 139 (Level-indexed Multi-step Substitution Reduction Relations)**.** For any $i \in \mathbb{N}$, let the level-indexed multi-step substitution reduction relation $\longrightarrow^{\mathrm{x}i*}$ be the reflexive-transitive closure of the level-indexed single-step substitution reduction relation $\longrightarrow^{\mathrm{x}i}$.

Definition 140 (Level-indexed Multi-step Relations)**.** For any $i \in \mathbb{N}$, let the level-indexed multi-step relation $\longrightarrow^{i*}$ be the reflexive-transitive closureof the level-indexed single-step relation $\longrightarrow^i$.

Example 141. Consider $!\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle$.

By the structural operational semantics of Explicit MetaML, we have:

$$\begin{array}{ll}
 & !\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle \\
\longrightarrow^0 & !\langle \lambda y.(\sim((\underline{\lambda} x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle \\
\longrightarrow^0 & !\langle \lambda y.(\sim((\underline{\lambda} x.\langle x\rangle)\ (\underline{\lambda} x.y))\ 0)\rangle \\
\longrightarrow^0 & !\langle \lambda y.(\sim(\langle x\rangle[x := (\underline{\lambda} x.y)])\ 0)\rangle \\
\longrightarrow^0 & !\langle \lambda y.(\sim(\langle x[x := (\underline{\lambda} x.y)]\rangle)\ 0)\rangle \\
\longrightarrow^0 & !\langle \lambda y.(\sim\langle \underline{\lambda} x.y\rangle\ 0)\rangle \\
\longrightarrow^0 & !\langle \lambda y.((\underline{\lambda} x.y)\ 0)\rangle \\
\longrightarrow^0 & \lambda y.((\underline{\lambda} x.y)\ 0) \\
\longrightarrow^0 & \underline{\lambda} y.((\underline{\lambda} x.y)\ 0)
\end{array}$$

As a comparison, by the substitutional structural operational semantics of MetaML, we have:

$$\begin{array}{ll}
 & !\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle \\
\longrightarrow^0 & !\langle \lambda y.(\sim(\langle x\rangle[(\lambda x.y)/x])\ 0)\rangle \\
= & !\langle \lambda y.(\sim(\langle x[(\lambda x.y)/x]\rangle)\ 0)\rangle \\
= & !\langle \lambda y.(\sim\langle \lambda x.y\rangle\ 0)\rangle \\
\longrightarrow^0 & !\langle \lambda y.((\lambda x.y)\ 0)\rangle \\
\longrightarrow^0 & \lambda y.((\lambda x.y)\ 0)
\end{array}$$

Explicit MetaML takes eight single-steps while Substitutional MetaML completes the execution in three single-steps. Some of these right steps are percolating an explicit substitution through a term and the rest are turning a lambda abstraction to its underlined counterpart.

Example 142. Suppose $x_2 \not\equiv x_3$ and $w \in \text{DENOTABLE}$. Consider $(\lambda x_1.\langle x_1 \rangle)\ ((\lambda x_2.(\lambda x_3.x_2))\ w)$.

By the structural operational semantics of Explicit MetaML, we have:

$$
\begin{array}{lll}
 & (\lambda x_1.\langle x_1 \rangle)\ ((\lambda x_2.(\lambda x_3.x_2))\ w) & \\
\longrightarrow^0 & (\underline{\lambda} x_1.\langle x_1 \rangle)\ ((\lambda x_2.(\lambda x_3.x_2))\ w) & \\
\longrightarrow^0 & (\underline{\lambda} x_1.\langle x_1 \rangle)\ ((\lambda x_3.x_2)[x_2 := w]) & \\
\longrightarrow^0 & (\underline{\lambda} x_1.\langle x_1 \rangle)\ (\lambda x_4.x_2[x_3 := x_4][x_2 := w]) & \text{where } x_4 \notin FV(\lambda x_3.t) \cup FV(w) \cup \{x_3\} \\
\longrightarrow^0 & (\underline{\lambda} x_1.\langle x_1 \rangle)\ (\underline{\lambda} x_4.x_2[x_3 := x_4][x_2 := w]) & \\
\longrightarrow^0 & \langle x_1 \rangle[x_1 := (\underline{\lambda} x_4.x_2[x_3 := x_4][x_2 := w])] & \\
\longrightarrow^0 & \langle x_1[x_1 := (\underline{\lambda} x_4.x_2[x_3 := x_4][x_2 := w])] \rangle & \\
\longrightarrow^0 & \langle \underline{\lambda} x_4.x_2[x_3 := x_4][x_2 := w] \rangle &
\end{array}
$$

Suppose we had not introduced the underlined lambda abstraction to Explicit MetaML, the above example would become:

$$
\begin{array}{lll}
 & (\lambda x_1.\langle x_1 \rangle)\ ((\lambda x_2.(\lambda x_3.x_2))\ w) & \\
\longrightarrow^0 & (\lambda x_1.\langle x_1 \rangle)\ ((\lambda x_3.x_2)[x_2 := w]) & \\
\longrightarrow^0 & (\lambda x_1.\langle x_1 \rangle)\ (\lambda x_4.x_2[x_3 := x_4][x_2 := w]) & \text{where } x_4 \notin FV(\lambda x_3.t) \cup FV(w) \cup \{x_3\} \\
\longrightarrow^0 & \langle x_1 \rangle[x_1 := (\lambda x_4.x_2[x_3 := x_4][x_2 := w])] & \\
\longrightarrow^0 & \langle x_1[x_1 := (\lambda x_4.x_2[x_3 := x_4][x_2 := w])] \rangle & \\
\longrightarrow^0 & \langle \lambda x_4.x_2[x_3 := x_4][x_2 := w] \rangle & \\
\longrightarrow^0 & \langle \lambda x_4.x_2[x_2 := w] \rangle & \\
\longrightarrow^0 & \langle \lambda x_4.w \rangle &
\end{array}
$$

After two single-steps, the lambda abstraction $\lambda x_4.x_2[x_3 := x_4][x_2 := w]$ is a level-0 value. Then it becomes a denotable term in the next step and shall be identified as a denotable term at all times ever since. However, after five single-steps, the lambda abstraction $\lambda x_4.x_2[x_3 := x_4][x_2 := w]$ is put into the context of level 1. We can apply the (lambda-(i+1)) rule to evaluate the body of the lambda abstraction at level 1. This destroys the property that a value at one level must be a value at any higher level, making Explicit MetaML inconsistent with Substitutional MetaML. To avoid this awkward circumstance, we let a level-0 lambda abstraction single-step to its underlined counterpart and make it a value at any level to prevent it from being reduced at higher levels.

In Explicit MetaML, explicit substitutions stop within the top of the body of a level-0 lambda value, even if it sits at higher level. This plays into the development of the analogous conception closures in Environmental MetaML.

Properties. We observe the following properties, which are useful in proving semantics equivalence. (1) The single-step relation preserves alpha equivalence. (2) The multi-step relation preserves the closedness of a runtime term. These are the same properties that Substitutional MetaML holds and are analogous to the ones that Explicit ISWIM holds.

Proposition 143. *If* $t_1^i \sim_\alpha t_2^i$ *and* $t_1^i \longrightarrow^i t_{11}^i$, *then* $t_2^i \longrightarrow^i t_{21}^i$ *and* $t_{11}^i \sim_\alpha t_{21}^i$.

Proposition 144. *If* $FV(t_1^i) = \emptyset$ *and* $t_1^i \longrightarrow^{*i} t_2^i$, *then* $FV(t_2^i) = \emptyset$.

Evaluator. We now define an evaluator in terms of the structural operational semantics of Explicit MetaML. The evaluator is analogous to the one defined in terms of Substitutional MetaML.

Definition 145 (Evaluator based on Structural Operational Semantics of Explicit MetaML)**.** Define the evaluator $eval_{\text{MetaML:ExpSOS}}$ to be a partial function from the set of programs $\text{PRGM}_{\text{MetaML}}$ to the set of answers $\text{ANS}_{\text{MetaML}}$.

$$eval_{\text{MetaML:ExpSOS}} : \text{PRGM}_{\text{MetaML}} \rightharpoonup \text{ANS}_{\text{MetaML}}$$

$$eval_{\text{MetaML:ExpSOS}}(t) = \begin{cases} \texttt{function} & \text{if } t \longrightarrow^{0*} \underline{\lambda} x.t'^0 \\ \texttt{code} & \text{if } t \longrightarrow^{0*} \langle v^1 \rangle \\ n & \text{if } t \longrightarrow^{0*} n \end{cases}$$

This evaluator is defined based on the structural operational semantics of Explicit MetaML. The subscript "$_{\text{MetaML:ExpSOS}}$" in $eval_{\text{MetaML:ExpSOS}}$ denotes the structural operational semantics of Explicit MetaML.

We claim that the evaluators defined based on Substitutional MetaML and Explicit MetaML are equivalent.

Theorem 146 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{MetaML}}$, $eval_{\text{MetaML:SubSOS}}(t)$ *is Kleene equal to* $eval_{\text{MetaML:ExpSOS}}(t)$.

We prove the theorem in the appendices.

5.3 Suspended MetaML

Following the path of refining Explicit ISWIM to Suspended ISWIM as presented in Section 3.3, we refine Explicit MetaML to Suspended MetaML. Suspended MetaML delays explicit substitutions outside of a level-0 lambda value until the lambda abstraction is called in an application. When performing an application at level 0, Suspended MetaML promotes substitution for the lambda bound variable to the front and overwrites any existing explicit substitution for that variable. That is, a level-0 application $(\underline{\lambda} x.t^0)\overline{[x_i := w_i]}\ v^0$ single-steps to $t^0[x := v^0]\overline{[x_i := w_i]}$ in Suspended MetaML. This transformation motivates introducing additional complexity in the ensuing dialects of MetaML. Proving the correctness of this transformation for Suspended MetaML is substantially more complex than for Suspended ISWIM. We introduce a notion of well-boundness judgement to help justify this transformation.

5.3.1 Syntax

We first define the basic syntax of the language: source terms, runtime terms, values and denotable terms. Then we define the free variable function, the substitution function and the alpha equivalence relation.

5.3.1.1 Source Terms, Runtime Terms, Values and Denotable Terms

Definition 147 (Source Terms, Runtime Terms, Values and Denotable Terms)**.** For any $i \in \mathbb{N}$, let STERM^i be the set of level-indexed source terms at level i, RTERM^i be the set of level-indexed runtime terms at level i and VALUE^i be the set of level-indexed values at level i. Let DENOTABLE be the set of denotable terms.

$$x \in \text{VAR},\ i,n \in \mathbb{N},\ t_{\text{s}}^i \in \text{STERM}^i,\ t^i \in \text{RTERM}^i,\ v^i \in \text{VALUE}^i,\ w \in \text{DENOTABLE}$$

$$\begin{array}{lcl}
t_{\text{s}}^0 & := & x \mid t_{\text{s}}^0\ t_{\text{s}}^0 \mid \lambda x.t_{\text{s}}^0 \mid \langle t_{\text{s}}^1 \rangle \mid\ !t_{\text{s}}^0 \mid n \mid t_{\text{s}}^0 + t_{\text{s}}^0 \\
t_{\text{s}}^{i+1} & := & x \mid t_{\text{s}}^{i+1}\ t_{\text{s}}^{i+1} \mid \lambda x.t_{\text{s}}^{i+1} \mid \langle t_{\text{s}}^{i+2} \rangle \mid \sim t_{\text{s}}^i \mid\ !t_{\text{s}}^{i+1} \mid n \mid t_{\text{s}}^{i+1} + t_{\text{s}}^{i+1} \\
 & & \\
t^0 & := & x \mid t^0\ t^0 \mid \lambda x.t^0 \mid \langle t^1 \rangle \mid\ !t^0 \mid n \mid t^0 + t^0 \mid \underline{\lambda} x.t^0 \mid t^0\overline{[x := w]} \\
t^{i+1} & := & x \mid t^{i+1}\ t^{i+1} \mid \lambda x.t^{i+1} \mid \langle t^{i+2} \rangle \mid \sim t^i \mid\ !t^{i+1} \mid n \mid t^{i+1} + t^{i+1} \mid \underline{\lambda} x.t^0 \mid t^i\overline{[x := w]} \mid \boxed{\hat{\lambda} x.t^{i+1}} \\
 & & \\
v^0 & := & \langle v^1 \rangle \mid n \mid \boxed{(\underline{\lambda} x.t^0)\overline{[x := w]}} \\
v^1 & := & x \mid v^1\ v^1 \mid \lambda x.v^1 \mid \langle v^2 \rangle \mid\ !v^1 \mid n \mid v^1 + v^1 \mid \boxed{(\underline{\lambda} x.t^0)\overline{[x := w]}} \\
v^{i+2} & := & x \mid v^{i+2}\ v^{i+2} \mid \lambda x.v^{i+2} \mid \langle v^{i+3} \rangle \mid \sim v^{i+1} \mid\ !v^{i+2} \mid n \mid v^{i+2} + v^{i+2} \mid \boxed{(\underline{\lambda} x.t^0)\overline{[x := w]}} \\
 & & \\
w & := & x \mid v^0
\end{array}$$

In Explicit MetaML, a level-0 lambda value $\underline{\lambda} x.t^0$ is a value at any level. Since Suspended MetaML does not push explicit substitutions into an underlined lambda abstraction, a level-0 lambda value surrounded by explicit substitutions, $(\underline{\lambda} x.t^0)\overline{[x := w]}$, is a value at any level.

Suspended MetaML evaluates the level-0 application $(\underline{\lambda} x.t^0)\overline{[x_i := w_i]}\ v^0$ to $t^0[x := v^0]\overline{[x_i := w_i]}$ rather than $t^0[x := x_N]\overline{[x_i := w_i]}[x_N := v^0]$ (where x_N is a fresh variable). To make it sound with respect to Explicit MetaML, we need to ensure that $FV(v^0) \cap (\bigcup_i \{x_i\}) = \emptyset$. In Suspended MetaML, the only way that v^0 can have free variables is as a result of performing an evaluation under lambdas at levels higher than 0. It is sufficient to make sure that whenever we go under a lambda during an evaluation at a level higher than 0, the lambda bound variable gets renamed to a fresh variable so that it is guaranteed not to clash with other variables. That is, when evaluating a lambda abstraction $\lambda x.t^{i+1}$ where $t^{i+1} \notin \text{VALUE}^{i+1}$, we rename the lambda bound variable x to a globally fresh variable x_N, resulting in $\lambda x_N.t^{i+1}[x := x_N]$, which is observationally equivalent to $\lambda x.t^{i+1}$. To avoid falling in a loop of renaming the lambda bound variable, we replace λ by its hatted counterpart $\hat{\lambda}$ to explicitly indicate that such a renaming has been done. This explains why we need a hatted lambda abstraction in Suspended MetaML. We demonstrate this in detail in Example 157.

5.3.1.2 Free Variable Function

We define the free variable function by extending Explicit MetaML's Definition 134 to accommodate hatted lambda abstractions.

Definition 148 (Free Variable Function)**.** Let the free variable function FV be a total function from the set

of runtime terms to the power set of variables.

$$FV \ : \ \text{RTERM} \longrightarrow \mathscr{P}(\text{VAR})$$

$$\begin{array}{rcll} \vdots & \vdots & \vdots & \vdots \\ FV(\hat{\lambda}x.t) & = & FV(\lambda x.t) & (11) \end{array}$$

Equations (1)-(10) are the same as Definition 134 in Explicit MetaML. Equation (11) tells that the free variables of a hatted lambda abstraction is the same as its unhatted counterpart.

5.3.1.3 Substitution Function

We define the substitution function by extending Explicit MetaML's Definition 135 to accommodate hatted lambda abstractions.

Definition 149 (Substitution Function)**.** Let the substitution function $\cdot[\cdot/\cdot]$ be a partial function from the 3-tuple of the set of runtime terms, the set of denotable terms and the set of variables, to the set of runtime terms.

$$\cdot[\cdot/\cdot] \ : \ (\text{RTERM} \times \text{DENOTABLE} \times \text{VAR}) \rightarrow \text{RTERM}$$

$$\begin{array}{rcll} \vdots & \vdots & \vdots & \vdots \\ (\hat{\lambda}x_1.t_0)[w/x_2] & = & \hat{\lambda}x_3.t_0[x_3/x_1][w/x_2] & \\ & & \text{where } x_3 \notin FV(\hat{\lambda}x_1.t_0) \cup FV(w) \cup \{x_2\} & (12) \end{array}$$

Equations (1)-(11) are the same as Definition 135 in Explicit MetaML. Equation (12) is analogous to Equations (4) and (10).

5.3.1.4 Alpha Equivalence Relation

We define the alpha equivalence relation by extending Explicit MetaML's Definition 136 to accommodate hatted lambda abstractions.

Definition 150 (Alpha Equivalence Relation)**.** Let the alpha equivalence relation $\sim_\alpha$ be a binary relation on runtime terms.

$$\sim_\alpha \subseteq \ \text{RTERM} \times \text{RTERM}$$

$$\vdots$$

$$\frac{t_1[x_3/x_1] \sim_\alpha t_2[x_3/x_2]}{(\hat{\lambda}x_1.t_1) \sim_\alpha (\hat{\lambda}x_2.t_2)} \text{ where } x_3 \notin FV(t_1) \cup FV(t_2) \text{ (lamh)}$$

All rules except (lamh) are the same as Definition 136 in Explicit MetaML. The (lamh) rule is analogous to the (lam) rule and the (lamu) rule.

5.3.2 Structural Operational Semantics

We lay out the structural operational semantics of Suspended MetaML through the level-indexed single-step relations $\longrightarrow^i$, the level-indexed single-step substitution reduction relations $\longrightarrow^{\mathrm{x}i}$, the global single-step relation $\triangleright \longrightarrow$, the level-indexed multi-step substitution reduction relations $\longrightarrow^{\mathrm{x}i*}$, the level-indexed multi-step relations $\longrightarrow^{i*}$ and the global multi-step relation $\triangleright \longrightarrow^*$.

Definition 151 (Level-indexed Single-step Relations)**.** For any $i \in \mathbb{N}$, let the level-indexed single-step relation $\longrightarrow^i$ be a 5-ary relation on the power set of variables, the power set of variables, the power set of variables, the set of runtime terms at level i and the set of runtime terms at level i.

$$\longrightarrow^i \subseteq \mathscr{P}(\mathrm{VAR}) \times \mathscr{P}(\mathrm{VAR}) \times \mathscr{P}(\mathrm{VAR}) \times \mathrm{RTERM}^i \times \mathrm{RTERM}^i$$

$$\boxed{\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash (\lambda x.t^0)\overline{[x_j := w_j]} \longrightarrow^0 (\underline{\lambda} x.t^0)\overline{[x_j := w_j]}}\ \text{(lambda-0)}}$$

$$\boxed{\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash \lambda x.t^{i+1} \longrightarrow^{i+1} \hat{\lambda} x_N.t^{i+1}[x := x_N]}\ \text{where } t^{i+1} \notin \mathrm{VALUE}^{i+1} \text{ and } x_N \notin \mathscr{X}\ \text{(lambda-(i+1)-t)}}$$

$$\boxed{\frac{\mathscr{U} \cup \{x\};\mathscr{V} \cup \{x\};\mathscr{X} \vdash t_1^{i+1} \longrightarrow^{i+1} t_2^{i+1}}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash \hat{\lambda} x.t_1^{i+1} \longrightarrow^{i+1} \hat{\lambda} x.t_2^{i+1}}\ \text{(lambda-(i+1)-r)}}$$

$$\boxed{\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash \hat{\lambda} x.v^{i+1} \longrightarrow^{i+1} \lambda x.v^{i+1}}\ \text{(lambda-(i+1)-v)}}$$

$$\frac{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_{11}^i \longrightarrow^i t_{12}^i}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_{11}^i\ t_2^i \longrightarrow^i t_{12}^i\ t_2^i}\ \text{(appL-i)} \qquad \frac{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_{21}^i \longrightarrow^i t_{22}^i}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash v_1^i\ t_{21}^i \longrightarrow^i v_1^i\ t_{22}^i}\ \text{(appR-i)}$$

$$\boxed{\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash (\underline{\lambda} x.t^0)\overline{[x_i := w_i]}\ v^0 \longrightarrow^0 t^0[x := v^0]\overline{[x_i := w_i]}}\ \text{(app-0)}}$$

$$\frac{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^i \longrightarrow^i t_2^i}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash !t_1^i \longrightarrow^i !t_2^i}\ \text{(run-i)} \qquad \frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash !\langle v^1 \rangle \longrightarrow^0 v^1}\ \text{(run-0)}$$

$$\frac{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^{i+1} \longrightarrow^{i+1} t_2^{i+1}}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash \langle t_1^{i+1} \rangle \longrightarrow^i \langle t_2^{i+1} \rangle}\ \text{(code-i)}$$

$$\frac{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^i \longrightarrow^i t_2^i}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash {\sim} t_1^i \longrightarrow^{i+1} {\sim} t_2^i}\ \text{(splice-(i+1))} \qquad \frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash {\sim}\langle v^1 \rangle \longrightarrow^1 v^1}\ \text{(splice-1)}$$

$$\frac{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_{11}^i \longrightarrow^i t_{12}^i}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_{11}^i + t_2^i \longrightarrow^i t_{12}^i + t_2^i}\ \text{(plusL-i)} \qquad \frac{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_{21}^i \longrightarrow^i t_{22}^i}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash v_1^i + t_{21}^i \longrightarrow^i v_1^i + t_{22}^i}\ \text{(plusR-i)}$$

$$\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash n_1 + n_2 \longrightarrow^0 n}\ \text{where } n = n_1 + n_2\ \text{(plus-0)}$$

$$\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^i \longrightarrow^i t_2^i}\ \text{where } \mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^i \longrightarrow^{\mathrm{x}i} t_2^i\ \text{(inj-subst)}$$

Definition 152 (Level-indexed Single-step Substitution Reduction Relations)**.** For any $i \in \mathbb{N}$, let the level-indexed single-step substitution reduction relation $\longrightarrow^{\mathrm{x}i}$ be a 5-ary relation on the power set of variables, the power set of variables, the power set of variables, the set of runtime terms at level i and the set of runtime terms at level i.

$$\longrightarrow^{\mathrm{x}i} \subseteq \mathscr{P}(\mathrm{VAR}) \times \mathscr{P}(\mathrm{VAR}) \times \mathscr{P}(\mathrm{VAR}) \times \mathrm{RTERM}^i \times \mathrm{RTERM}^i$$

$$\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash x[x := w] \longrightarrow^{\mathrm{x}i} w} \text{ (var-eq-subst)}$$

$$\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash x_1[x_2 := w] \longrightarrow^{\mathrm{x}i} x_1} \text{ where } x_1 \not\equiv x_2 \text{ (var-df-subst)}$$

$$\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash n[x := w] \longrightarrow^{\mathrm{x}i} n} \text{ (num-subst)}$$

$$\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash (t_1^i\ t_2^i)[x := w] \longrightarrow^{\mathrm{x}i} (t_1^i[x := w])\ (t_2^i[x := w])} \text{ (app-subst)}$$

$$\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash (t_1^i + t_2^i)[x := w] \longrightarrow^{\mathrm{x}i} (t_1^i[x := w]) + (t_2^i[x := w])} \text{ (plus-subst)}$$

$$\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash (\lambda x_1.t^{i+1})[x_2 := w] \longrightarrow^{\mathrm{x}(i+1)} \lambda x_N.t^{i+1}[x_1 := x_N][x_2 := w]} \text{ where } x_N \notin \mathscr{X} \text{ (lam-subst)}$$

no (lamu-subst)

$$\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash \langle t^{i+1} \rangle[x := w] \longrightarrow^{\mathrm{x}i} \langle t^{i+1}[x := w] \rangle} \text{ (code-subst)}$$

$$\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash (!t^i)[x := w] \longrightarrow^{\mathrm{x}i} !t^i[x := w]} \text{ (run-subst)}$$

$$\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash (\sim t^i)[x := w] \longrightarrow^{\mathrm{x}(i+1)} \sim t^i[x := w]} \text{ (splice-subst)}$$

$$\frac{\mathscr{U} \cup \{x_2\};\mathscr{V};\mathscr{X} \vdash t_1^i[x_1 := w_1] \longrightarrow^{\mathrm{x}i} t_2^i}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^i[x_1 := w_1][x_2 := w_2] \longrightarrow^{\mathrm{x}i} t_2^i[x_2 := w_2]} \text{ (subst-subst)}$$

The level-indexed single-step relation $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^i \longrightarrow^i t_2^i$ reads as "t_1 single-steps to t_2 at level i bound by $\mathscr{U}$, $\mathscr{V}$ and $\mathscr{X}$".

To make a small-step when evaluating a program, we need to repeatedly apply the structural rules until we find a subterm of the program on which a reduction rule can be applied. Suppose we have found such a subterm t_1^i that is reducible and we have $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^i \longrightarrow^i t_2^i$. As illustrated by the (subst-subst) rule and the (lambda-(i+1)-r) rule, the free variables of the term t_1^i must be bound by any means (i.e., explicit substitutions or hatted lambda bound variables) in the surrounding scope, which are tracked by $\mathscr{U}$. The free variables of the term t_1^i that are bound by hatted lambda bound variables are also tracked by the variable set $\mathscr{V}$ because the well-boundness judgement introduced in Definition 159 needs to pay special attention to these variables. Both of the variable sets $\mathscr{U}$ and $\mathscr{V}$ play important roles in justifying the (app-0) rule. The variable set $\mathscr{X}$ records all variables in the subterm t_1^i and its surrounding scope. We need the variable set $\mathscr{X}$ to determine whether a variable is fresh in the sense that it has not appeared in the term being currently evaluated or in its surrounding scope. The variable sets $\mathscr{U}$, $\mathscr{V}$ and $\mathscr{X}$ help specify the well-boundness

property of terms (Proposition 161).

The (app-0) rule is refined from the following (app-0-naive) rule.

$$\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash (\underline{\lambda} x.t^0)\overline{[x_i := w_i]}\ v^0 \longrightarrow^0 t^0[x := x_0]\overline{[x_i := w_i]}[x_0 := v^0]} \text{ where } \begin{array}{l} x_0 \notin FV(\underline{\lambda} x.t^0) \cup \\ \bigcup_i (FV(w_i) \cup \{x_i\}) \end{array} \quad \text{(app-0-naive)}$$

The (app-0-naive) rule is semantically correct but not ideal. We want to eliminate renamings in any level-0 application, but the (app-0-naive) rule still renames the lambda bound variable.

Observe that if $FV(v^0) \cap (\bigcup_i \{x_i\}) = \emptyset$, then we can promote the substitution $[x_0 := v^0]$ to the front of the explicit substitutions $\overline{[x_i := w_i]}$. Then we can eliminate renaming the lambda bound variable x by combining $[x := x_0]$ and $[x_0 := v^0]$ to $[x := v^0]$. It is provable that when the (app-0) rule is applied to a subterm of an intermediate result of evaluating a program, we have $FV(v^0) \cap (\bigcup_i \{x_i\}) = \emptyset$. We discuss the intuition behind this after presenting the well-boundness judgement whose primary purpose is to help complete this proof.

Since the ultimate goal of the main refinement problem is to develop an environmental operational semantics for MetaML, the (app-0) rule promotes the substitution for the underlined lambda bound variable to the front, superseding any existing explicit substitution for that variable, which is close to the operation of updating an environment in Environmental MetaML that is introduced in the next section.

Explicit MetaML's (lambda-(i+1)) rule is replaced with three (lambda-(i+1)-?) rules in Suspended MetaML. To evaluate a level-$(i+1)$ lambda abstraction, if its body is not a level-$(i+1)$ value, we first apply the (lambda-(i+1)-t) rule to rename the lambda bound variable to a fresh variable that has not occurred in the current term being evaluated and in the surrounding scope and replace λ by $\hat{\lambda}$ to indicate such a renaming is done. It is important to check whether the lambda abstraction is already a value before renaming the lambda bound variable, which ensures no unnecessary renaming can happen. Checking this is a deep syntactic operation in Suspended MetaML but becomes a shallow check in the MEK machine. Then we repeatedly apply the (lambda-(i+1)-r) rule to reduce its body until it is a value. Finally we apply the (lambda-(i+1)-v) rule to change $\hat{\lambda}$ back to λ. Suspended MetaML forces the renaming to make the (app-0) rule sound. Example 157 shows evaluations would behave unexpectedly if Suspended MetaML used Explicit MetaML's (lambda-(i+1)) rule instead.

The (lam-subst) rule no longer concerns level 0 because a level-0 lambda abstraction surrounded by explicit substitutions single-steps to its underlined counterpart which is a level-0 value. There is no (lamu-subst) rule because an underlined lambda abstraction surrounded by explicit substitutions is a value at any level.

To apply the level-indexed single-step relation on a program, the variable sets $\mathscr{U};\mathscr{V};\mathscr{X}$ need to be properly initialised. Moreover, from a user's perspective, the only interface to the single-step relation should be the program to be evaluated. We define the global single-step relation, which initialises the variable sets $\mathscr{U};\mathscr{V};\mathscr{X}$ by itself and only shows what an entire program single-steps to.

Definition 153 (Global Single-step Relation)**.** Let the global single-step relation $\triangleright \longrightarrow$ be a binary relation between the set of level-0 runtime terms and the set of level-0 runtime terms.

$$\triangleright\, t_1^0 \longrightarrow t_2^0 \qquad \text{if and only if} \qquad \emptyset;\emptyset;\textsc{Var}(t_1^0) \vdash t_1^0 \longrightarrow^0 t_2^0$$

The global single-step relation $\triangleright\, t_1^0 \longrightarrow t_2^0$ reads as "t_1 single-steps to t_2".

Definition 154 (Level-indexed Multi-step Substitution Reduction Relations)**.** For any $i \in \mathbb{N}$, let the multi-step substitution reduction relation $\longrightarrow^{\mathrm{x}i*}$ be a 5-ary relation on the power set of variables, the power set of variables, the power set of variables, the set of runtime terms at level i and the set of runtime terms at level i directly based on the level-indexed single-step substitution relation $\longrightarrow^{\mathrm{x}i}$.

$$\longrightarrow^{\mathrm{x}i*} \subseteq \mathscr{P}(\mathrm{VAR}) \times \mathscr{P}(\mathrm{VAR}) \times \mathscr{P}(\mathrm{VAR}) \times \mathrm{RTERM}^i \times \mathrm{RTERM}^i$$

$$\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^i \longrightarrow^{\mathrm{x}i*} t_2^i} \text{ where } \mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^i \longrightarrow^{\mathrm{x}i} t_2^i \text{ (step)} \qquad \frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^i \longrightarrow^{\mathrm{x}i*} t^i} \text{ (refl)}$$

$$\frac{\mathscr{U};\mathscr{V};\mathscr{X} \vdash \ t_1^i \longrightarrow^{\mathrm{x}i*} t_2^i \quad \mathscr{U};\mathscr{V};\mathscr{X} \cup \mathrm{VAR}(t_2^i) \vdash t_2^i \longrightarrow^{\mathrm{x}i*} t_3^i}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^i \longrightarrow^{\mathrm{x}i*} t_3^i} \text{ (trans)}$$

Definition 155 (Level-indexed Multi-step Relations)**.** For any $i \in \mathbb{N}$, let the level-indexed multi-step relation $\longrightarrow^{i*}$ be a 5-ary relation on the power set of variables, the power set of variables, the power set of variables, the set of runtime terms at level i and the set of runtime terms at level i directly based on the level-indexed single-step relation $\longrightarrow^{i}$.

$$\longrightarrow^{i*} \subseteq \mathscr{P}(\mathrm{VAR}) \times \mathscr{P}(\mathrm{VAR}) \times \mathscr{P}(\mathrm{VAR}) \times \mathrm{RTERM}^i \times \mathrm{RTERM}^i$$

$$\frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^i \longrightarrow^{i*} t_2^i} \text{ where } \mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^i \longrightarrow^{i} t_2^i \text{ (step)} \qquad \frac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^i \longrightarrow^{i*} t^i} \text{ (refl)}$$

$$\frac{\mathscr{U};\mathscr{V};\mathscr{X} \vdash \ t_1^i \longrightarrow^{i*} t_2^i \quad \mathscr{U};\mathscr{V};\mathscr{X} \cup \mathrm{VAR}(t_2^i) \vdash t_2^i \longrightarrow^{i*} t_3^i}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^i \longrightarrow^{i*} t_3^i} \text{ (trans)}$$

Definition 156 (Global Multi-step Relation)**.** Let the global multi-step relation $\triangleright \longrightarrow^*$ be the reflexive-transitive closure of the global single-step relation $\triangleright \longrightarrow$.

Example 157. Consider $!\langle \lambda y.{\sim}(((\lambda y.(\lambda x.x))\ n)\ \langle y \rangle)\rangle$.

By the structural operational semantics of Suspended MetaML, we have:

$$
\begin{array}{rll}
 & !\langle \lambda y.\sim(((\lambda y.(\lambda x.x))\ n)\ \langle y\rangle)\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.(\sim(((\lambda y.(\lambda x.x))\ n)\ \langle y\rangle))[y := z]\rangle & \text{where } z \notin \{x, y\} \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.(\sim(((\lambda y.(\lambda x.x))\ n)\ \langle y\rangle)[y := z])\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.\sim(((\lambda y.(\lambda x.x))\ n)[y := z]\ \langle y\rangle[y := z])\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.\sim(((\lambda y.(\lambda x.x))[y := z]\ n[y := z])\ \langle y\rangle[y := z])\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.\sim(((\underline{\lambda} y.(\lambda x.x))[y := z]\ n[y := z])\ \langle y\rangle[y := z])\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.\sim(((\underline{\lambda} y.(\lambda x.x))[y := z]\ n)\ \langle y\rangle[y := z])\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.\sim((\lambda x.x)[y := n][y := z]\ \langle y\rangle[y := z])\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.\sim((\underline{\lambda} x.x)[y := n][y := z]\ \langle y\rangle[y := z])\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.\sim((\underline{\lambda} x.x)[y := n][y := z]\ \langle y[y := z]\rangle)\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.\sim((\underline{\lambda} x.x)[y := n][y := z]\ \langle z\rangle)\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.\sim(x[x := \langle z\rangle][y := n][y := z])\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.\sim\langle z\rangle[y := n][y := z]\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.\sim\langle z[y := n]\rangle[y := z]\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.\sim\langle z[y := n][y := z]\rangle\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.\sim\langle z[y := z]\rangle\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.\sim\langle z\rangle\rangle & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} z.z\rangle & \\
\triangleright \longrightarrow & !\langle \lambda z.z\rangle & \\
\triangleright \longrightarrow & \lambda z.z & \\
\triangleright \longrightarrow & \underline{\lambda} z.z &
\end{array}
$$

Suppose we had not introduced the hatted lambda abstraction to Suspended MetaML, the above evaluation would become

$$
\begin{array}{rl}
 & !\langle \lambda y.\sim(((\lambda y.(\lambda x.x))\ n)\ \langle y\rangle)\rangle \\
\triangleright \longrightarrow & !\langle \lambda y.\sim(((\underline{\lambda} y.(\lambda x.x))\ n)\ \langle y\rangle)\rangle \\
\triangleright \longrightarrow & !\langle \lambda y.\sim((\lambda x.x)[y := n]\ \langle y\rangle)\rangle \\
\triangleright \longrightarrow & !\langle \lambda y.\sim(\boxed{(\underline{\lambda} x.x)[y := n]\ \langle y\rangle})\rangle \\
\triangleright \longrightarrow & !\langle \lambda y.\sim(\boxed{x[x := \langle y\rangle][y := n]})\rangle \\
\triangleright \longrightarrow & !\langle \lambda y.\sim(\langle y\rangle[y := n])\rangle \\
\triangleright \longrightarrow & !\langle \lambda y.\sim\langle y[y := n]\rangle\rangle \\
\triangleright \longrightarrow & !\langle \lambda y.\sim\langle n\rangle\rangle \\
\triangleright \longrightarrow & !\langle \lambda y.n\rangle \\
\triangleright \longrightarrow & \lambda y.n
\end{array}
$$

whose result is incorrect. The boxes highlight a critical mistake of the evaluation.

Since we did not rename the lambda bound variable at the first single-step before we dove into the body of the lambda abstraction, we have $FV(\langle y\rangle) \cap \{y\} \neq \emptyset$ for the boxed application $(\underline{\lambda} x.x)[y := n]\ \langle y\rangle$. Hence

we cannot apply the (app-0) rule to the application. We should apply the (app-0-naive) rule and we get $x[x := x_0][y := n][x_0 := \langle y \rangle]$ where $x_0 \notin FV(\underline{\lambda}x.x) \cup FV(n) \cup \{y\}$.

Example 158. Consider $!\langle \lambda y.(\sim((\lambda x.\langle x \rangle)\ (\lambda z.y))\ 0) \rangle\ 5$.

By the structural operational semantics of Suspended MetaML, we have:

$$
\begin{array}{lll}
 & !\langle \lambda y.(\sim((\lambda x.\langle x \rangle)\ (\lambda z.\langle y \rangle))\ 0) \rangle\ 5 & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} u.(\sim((\lambda x.\langle x \rangle)\ (\lambda z.\langle y \rangle))\ 0)[y := u] \rangle\ 5 & \text{where } u \notin \{x,y,z\} \\
\triangleright \longrightarrow & !\langle \hat{\lambda} u.(\sim((\lambda x.\langle x \rangle)\ (\lambda z.\langle y \rangle))[y := u]\ 0[y := u]) \rangle\ 5 & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} u.(\sim(((\lambda x.\langle x \rangle)\ (\lambda z.\langle y \rangle))[y := u])\ 0[y := u]) \rangle\ 5 & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} u.(\sim((\lambda x.\langle x \rangle)[y := u]\ (\lambda z.\langle y \rangle)[y := u])\ 0[y := u]) \rangle\ 5 & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} u.(\sim((\underline{\lambda} x.\langle x \rangle)[y := u]\ (\lambda z.\langle y \rangle)[y := u])\ 0[y := u]) \rangle\ 5 & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} u.(\sim((\underline{\lambda} x.\langle x \rangle)[y := u]\ (\underline{\lambda} z.\langle y \rangle)[y := u])\ 0[y := u]) \rangle\ 5 & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} u.(\sim\langle x \rangle[x := (\underline{\lambda} z.\langle y \rangle)[y := u]][y := u]\ 0[y := u]) \rangle\ 5 & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} u.(\sim\langle x[x := (\underline{\lambda} z.\langle y \rangle)[y := u]] \rangle[y := u]\ 0[y := u]) \rangle\ 5 & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} u.(\sim\langle x[x := (\underline{\lambda} z.\langle y \rangle)[y := u]][y := u] \rangle\ 0[y := u]) \rangle\ 5 & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} u.(\sim\langle (\underline{\lambda} z.\langle y \rangle)[y := u][y := u] \rangle\ 0[y := u]) \rangle\ 5 & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} u.((\underline{\lambda} z.\langle y \rangle)[y := u][y := u]\ 0[y := u]) \rangle\ 5 & \\
\triangleright \longrightarrow & !\langle \hat{\lambda} u.((\underline{\lambda} z.\langle y \rangle)[y := u][y := u]\ 0) \rangle\ 5 & \\
\triangleright \longrightarrow & !\langle \lambda u.((\underline{\lambda} z.\langle y \rangle)[y := u][y := u]\ 0) \rangle\ 5 & \\
\triangleright \longrightarrow & (\lambda u.((\underline{\lambda} z.\langle y \rangle)[y := u][y := u]\ 0))\ 5 & \\
\triangleright \longrightarrow & (\underline{\lambda} u.((\underline{\lambda} z.\langle y \rangle)[y := u][y := u]\ 0))\ 5 & \\
\triangleright \longrightarrow & ((\underline{\lambda} z.\langle y \rangle)[y := u][y := u]\ 0)[u := 5] & \\
\triangleright \longrightarrow & (\underline{\lambda} z.\langle y \rangle)[y := u][y := u][u := 5]\ 0[u := 5] & \\
\triangleright \longrightarrow & (\underline{\lambda} z.\langle y \rangle)[y := u][y := u][u := 5]\ 0 & \\
\triangleright \longrightarrow & \langle y \rangle[z := 0][y := u][y := u][u := 5] & \\
\triangleright \longrightarrow & \langle y[z := 0] \rangle[y := u][y := u][u := 5] & \\
\triangleright \longrightarrow & \langle y[z := 0][y := u] \rangle[y := u][u := 5] & \\
\triangleright \longrightarrow & \langle y[z := 0][y := u][y := u] \rangle[u := 5] & \\
\triangleright \longrightarrow & \langle \boxed{y[z := 0][y := u][y := u][u := 5]} \rangle & \\
\triangleright \longrightarrow & \langle y[y := u][y := u][u := 5] \rangle & \\
\triangleright \longrightarrow & \langle u[y := u][u := 5] \rangle & \\
\triangleright \longrightarrow & \langle u[u := 5] \rangle & \\
\triangleright \longrightarrow & \langle 5 \rangle &
\end{array}
$$

Consider the boxed term, $y[z := 0][y := u][y := u][u := 5]$. The variable y is bound by the substitution $[y := u]$ in which the denotable term u is bound by the substitution $[u := 5]$. By the (subst-subst) rule, Suspended MetaML resolve the substitutions by performing one substitution at a time. If we represent substitutions using environments, a trivial approach will go wrong. This is discussed in detail at the beginning of the next section.

Properties. Our sole purpose of developing the structural operational semantics is to evaluate programs which are closed level-0 source terms. To evaluate a program t_s^0 at level 0, we expect that the program t_s^0 multi-steps to a value v^0 at level 0. Suppose the closed term t_1^0 is an intermediate result of evaluating the program t_s^0. For any subterm t_{11} of the term t_1^0, we want to ensure that all its free variables are bound in its surrounding scope. In particular, we are interested in the free variables that are bound by hatted lambda bound variables in the surrounding scope. We define the following judgement to specify the well-boundness of a term. We demonstrate how this judgement helps prove the correctness of the (app-0) rule.

Definition 159 (Well-boundness Judgement)**.** Let the well-boundness judgement $\vdash\ wb$ be a ternary relation on the power set of variables, the power set of variables and the set of runtime terms.

$$\vdash\ wb\ \subseteq\ \mathscr{P}(\textsc{Var}) \times \mathscr{P}(\textsc{Var}) \times \textsc{RTerm}$$

$$\frac{}{\mathscr{U};\mathscr{V} \vdash x\ wb}\ \text{where } x \in \mathscr{U}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t_1\ wb \quad \mathscr{U};\mathscr{V} \vdash t_2\ wb}{\mathscr{U};\mathscr{V} \vdash t_1\ t_2\ wb}$$

$$\frac{\mathscr{U} \cup \{x\};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash \lambda x.t\ wb}\ \text{where } x \notin \mathscr{V}$$

$$\frac{\mathscr{U} \cup \{x\};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash \underline{\lambda} x.t\ wb}\ \text{where } x \notin \mathscr{V}$$

$$\frac{\mathscr{U} \cup \{x\};\mathscr{V} \cup \{x\} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash \hat{\lambda} x.t\ wb}\ \text{where } x \notin \mathscr{V}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash \langle t \rangle\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash {\sim} t\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash !t\ wb}$$

$$\frac{}{\mathscr{U};\mathscr{V} \vdash n\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t_1\ wb \quad \mathscr{U};\mathscr{V} \vdash t_2\ wb}{\mathscr{U};\mathscr{V} \vdash t_1 + t_2\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash w\ wb \quad \mathscr{U} \cup \{x\};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash t[x := w]\ wb}\ \text{where } x \notin \mathscr{V}$$

The well-boundness judgement $\mathscr{U};\mathscr{V} \vdash t\ wb$ reads as "t is well bound by $\mathscr{U}$ and $\mathscr{V}$".

We come back to our discussion on the well-boundness of the subterm t_{11} of the term t_1^0. By the above definition, we have $\emptyset;\emptyset \vdash t_1^0\ wb$. Observe that a sub-derivation of $\emptyset;\emptyset \vdash t_1\ wb$ must be the derivation of $\mathscr{U};\mathscr{V} \vdash t_{11}\ wb$ for some variable sets $\mathscr{U}$ and $\mathscr{V}$. The variable set $\mathscr{U}$ tracks all free variables of the subterm

t_{11} that are bound in the surrounding scope. We observe the following property which shows if a term is well bound, we can get an upper bound of the free variables of the term. We can use well-boundness judgement to estimate the free variables of a term.

Proposition 160. *If $\mathscr{U};\mathscr{V} \vdash t$ wb, then $FV(t) \subseteq \mathscr{U}$.*

The variable set $\mathscr{V}$ in $\mathscr{U};\mathscr{V} \vdash t_{11}$ *wb* tracks the free variables of the subterm t_{11} that are bound by hatted lambdas in the surrounding scope. Consider a special case of the subterm t_{11} of the term t_1^0. Suppose t_{11} is an application $(\underline{\lambda}x.t^0)\overline{[x_i := w_i]}\ v^0$ that is reducible by the (app-0) rule. Since the (app-0) rule is not a substitution reduction, it cannot be applied under any explicit substitution. Thus the free variables of v^0 must be bound by hatted lambdas in its surrounding context, which are tracked by the variable set $\mathscr{V}$ of the judgement $\mathscr{U};\mathscr{V} \vdash (\underline{\lambda}x.t^0)\overline{[x_i := w_i]}\ v^0$ *wb*. By the definition of the well-boundness judgement, we have $x_i \notin \mathscr{V}$. Hence we have $FV(v^0) \cap (\bigcup_i \{x_i\}) = \emptyset$, which we call the well-boundness of the application $(\underline{\lambda}x.t^0)\overline{[x_i := w_i]}\ v^0$. This guarantees the correctness of the (app-0) rule.

The well-boundness judgement cooperates well with the multi-step relation. The second property says the former is preserved by the latter.

Proposition 161. *If $\mathscr{U};\mathscr{V} \vdash t_1^i$ wb, $\textsc{Var}(t_1^i) \subseteq \mathscr{X}$, $\mathscr{V} \subseteq \mathscr{U} \subseteq \mathscr{X}$ and $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1^i \longrightarrow^{i*} t_2^i$, then $\mathscr{U};\mathscr{V} \vdash t_2^i$ wb.*

As a corollary of the above properties, the multi-step relation preserves the closedness of runtime terms.

Evaluator. We now define an evaluator in terms of the structural operational semantics of Suspended MetaML. The evaluator is analogous to the one defined for Explicit MetaML.

Definition 162 (Evaluator based on Structural Operational Semantics of Suspended MetaML)**.** Define the evaluator $eval_{\text{MetaML:SusSOS}}$ to be a partial function from the set of programs $\textsc{Prgm}_{\text{MetaML}}$ to the set of answers $\textsc{Ans}_{\text{MetaML}}$.

$$eval_{\text{MetaML:SusSOS}} : \textsc{Prgm}_{\text{MetaML}} \rightharpoonup \textsc{Ans}_{\text{MetaML}}$$

$$eval_{\text{MetaML:SusSOS}}(t) = \begin{cases} \texttt{function} & \text{if } \triangleright\, t \longrightarrow^* (\underline{\lambda}x.t'^0)\overline{[x_i := w_i]} \\ \texttt{code} & \text{if } \triangleright\, t \longrightarrow^* \langle v^1 \rangle \\ n & \text{if } \triangleright\, t \longrightarrow^* n \end{cases}$$

This evaluator is defined based on the structural operational semantics of Suspended MetaML. The subscript "$_{\text{MetaML:SusSOS}}$" in $eval_{\text{MetaML:SusSOS}}$ denotes the structural operational semantics of Suspended MetaML.

We claim that the evaluators defined in terms of Substitutional MetaML and Suspended MetaML are equivalent.

Theorem 163 (Kleene Equality of Evaluators)**.** *For any $t \in \textsc{Prgm}_{\text{MetaML}}$, $eval_{\text{MetaML:SubSOS}}(t)$ is Kleene equal to $eval_{\text{MetaML:SusSOS}}(t)$.*

We prove the above theorem in appendices.

5.4 Environmental MetaML - Structural Operational Semantics

Refining Suspended ISWIM to Environmental ISWIM is discussed in Section 3.4. For a term surrounded by cascaded explicit substitutions, $t\overline{[x_i := w_i]}$, every denotable term w_i is closed in Suspended ISWIM. It is a natural step to replace the explicit substitutions with a corresponding environment.

However, MetaML allows replacing a variable by a open denotable term. In Suspended MetaML, the denotable term of some early explicit substitution in a cascade may have free variables that are bound by some later substitution. Example 158 shows

$$\triangleright \quad !\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda z.y))\ 0)\rangle\ 5 \quad \longrightarrow^* \quad \langle y[z := 0][y := u][y := u][u := 5]\rangle$$

where $u \notin \{x, y, z\}$. The term $y[z := 0][y := u][y := u][u := 5]$ has y bound by u that is bound by 5. If we represent the explicit substitutions that surround a term by one environment, an environment lookup may lead to an unsound result. For example, we represent $y[z := 0][y := u][y := u][u := 5]$ by pairing y with the environment $\rho = \{(z, 0), (y, u), (u, 5)\}$. Looking up y in the environment ρ returns u solely which is not paired with an environment. Then we incorrectly output u as the final result of the evaluation. The correct result is 5 because u is bound to 5.

To refine Suspended MetaML to Environmental MetaML, we propose to replace cascaded explicit substitutions with meta-environments instead of environments. A meta-environment is a finite sequence of environments, among which the free variables of one environment are bound by the next environment in the sequence. To evaluate a variable with a meta-environment, we look up the variable in the first environment of the meta-environment and return the lookup result paired with the remaining environments of the meta-environment. For example, $y[z := 0][y := u][y := u][u := 5]$ may be represented by pairing y with the meta-environment $(\rho_1; \rho_2)$ where $\rho_1 = \{(z, 0), (y, u)\}$ and $\rho_2 = \{(u, 5)\}$. After one small-step of evaluation, we have u paired with the meta-environment ρ_2 because $\rho_1(y) = u$. After another step, we get the final result 5 because $\rho_2(u) = 5$.

5.4.1 Syntax

We first define the basic syntax of the Environmental MetaML: source terms, runtime terms, values, denotable terms, configurations, environments and meta-environments. Then we define the free variable function.

5.4.1.1 Source Terms, Runtime Terms, Values, Denotable Terms, Configurations, Environments and Meta-environments

Definition 164 (Source Terms, Runtime Terms, Values, Denotable Terms and Configurations)**.** For any $i \in \mathbb{N}$, let STERM^i be the set of level-indexed source terms at level i, RTERM^i be the set of level-indexed runtime terms at level i, VALUE^i be the set of level-indexed values at level i, and CONF^i be the set of level-indexed configurations at level i. Let DENOTABLE be the set of denotable terms and ENV be a finite partial function from the set of variables to the set of denotable terms.

$x \in \text{VAR}$, $i,n \in \mathbb{N}$, $t_s^i \in \text{STERM}^i$, $t^i \in \text{RTERM}^i$, $v^i \in \text{VALUE}^i$, $w \in \text{DENOTABLE}$, $c \in \text{CONF}$,
$\rho \in \text{ENV} = \text{VAR} \xrightarrow{\text{fin}} \text{DENOTABLE}$

$$
\begin{array}{lcl}
t_s^0 & := & x \mid t^0\ t^0 \mid \lambda x.t^0 \mid \langle t^1 \rangle \mid\ !t^0 \mid n \mid t^0 + t^0 \\
t_s^{i+1} & := & x \mid t^{i+1}\ t^{i+1} \mid \lambda x.t^{i+1} \mid \langle t^{i+2} \rangle \mid \sim t^i \mid\ !t^{i+1} \mid n \mid t^{i+1} + t^{i+1} \\
 & & \\
t^0 & := & x \mid t^0\ t^0 \mid \triangleleft\ \lambda x.t^0,\ \rho^*\ \triangleright \mid \langle t^1 \rangle \mid\ !t^0 \mid n \mid t^0 + t^0 \\
t^{i+1} & := & x \mid t^{i+1}\ t^{i+1} \mid \lambda x.t^{i+1} \mid \langle t^{i+2} \rangle \mid \sim t^i \mid\ !t^{i+1} \mid n \mid t^{i+1} + t^{i+1} \\
 & & \\
v^0 & := & \triangleleft\ \lambda x.t^0,\ \rho^*\ \triangleright \mid \langle v^1 \rangle \mid n \\
v^1 & := & x \mid v^1\ v^1 \mid \lambda x.v^1 \mid \langle v^2 \rangle \mid\ !v^1 \mid n \mid v^1 + v^1 \\
v^{i+2} & := & x \mid v^{i+2}\ v^{i+2} \mid \lambda x.v^{i+2} \mid \langle v^{i+3} \rangle \mid \sim v^{i+1} \mid\ !v^{i+2} \mid n \mid v^{i+2} + v^{i+2} \\
 & & \\
w & := & x \mid v^0 \\
 & & \\
c^0 & := & v^0 \mid c^0\ c^0 \mid \lambda x.c^0 \mid \blacktriangleleft\ t^0,\ \rho^*\ \blacktriangleright \mid \langle c^1 \rangle \mid !c^0 \mid c^0 + c^0 \\
c^{i+1} & := & v^{i+1} \mid c^{i+1}\ c^{i+1} \mid \lambda x.c^{i+1} \mid \blacktriangleleft\ t^{i+1},\ \rho^*\ \blacktriangleright \mid \langle c^{i+2} \rangle \mid \sim c^i \mid !c^{i+1} \mid c^{i+1} + c^{i+1}
\end{array}
$$

We usually call $\lambda x.c^{i+1}$ a level-(i+1) non-value lambda if $c \notin \text{VALUE}^{i+1}$ and a level-(i+1) lambda value if $c \in \text{VALUE}^{i+1}$.

Definition 165 (Environments)**.** An environment $\rho \in \text{ENV}$ is a finite partial function from the set of variables to the set of denotable terms. Let $\text{dom}(\rho)$ be the domain of the environment ρ and $\text{rng}(\rho)$ be the range of the environment ρ. Let $\rho[x \mapsto w]$ be an environment update and $\rho(x)$ be an environment lookup. We have:

$$\rho[x \mapsto w](y) = \begin{cases} w & \text{if } x \equiv y \\ \rho(y) & \text{if } x \not\equiv y \end{cases}$$

Definition 166 (Initial Environments)**.** Let $\mathscr{X} \subseteq \text{VAR}$. The initial environment $\rho_{\text{init}}^{\mathscr{X}}$ is the identity function whose domain is $\mathscr{X}$.

Definition 167 (Meta-environments)**.** A meta-environment $\rho^* \in \text{ENV}^*$ is a finite sequence of environments.

An empty meta-environment is denoted by ε. A meta-environment containing an empty environment is denoted by $(\emptyset;\varepsilon)$.

A term paired with a meta-environment, i.e., $\blacktriangleleft\ t^i,\ \rho^*\ \blacktriangleright$ or $\triangleleft\ \lambda x.t^0,\ \rho^*\ \triangleright$, is called a *closure*. A level-0 lambda abstraction paired with a meta-environment, i.e., $\triangleleft\ \lambda x.t^0,\ \rho^*\ \triangleright$, is called a *closure value*. A closure that is not a value, $\blacktriangleleft\ t^i,\ \rho^*\ \blacktriangleright$, is called a *non-value closure*. A closure makes its top-level structure immediately evident. For example, it is immediately recognisable that its top-level structure of the closure $\triangleleft\ \lambda x.t^0,\ \rho^*\ \triangleright$ is a level-0 lambda abstraction $\lambda x.t^0$ without having to dive into the meta-environment ρ^*. As a comparison, in Suspended MetaML, to check the top-level structure of $(\underline{\lambda} x.t^0)\overline{[x_i := w_i]}$, we have to

dive down through the cascaded explicit substitutions $\overline{[x_i := w_i]}$ until reaching the level-0 lambda abstraction $\underline{\lambda} x.t^0$.

A closure $\blacklozenge\ t,\ \rho^*\ \blacklozenge$ has the meta-environment ρ^*. The meta-environment of the closure $\blacklozenge\ t,\ (\emptyset;\varepsilon)\ \blacklozenge$ has one environment $\emptyset$. The meta-environment of the closure $\blacklozenge\ t,\ \varepsilon\ \blacklozenge$ has no environments.

Environmental MetaML's closures are different from Environmental ISWIM's closures. In Environmental ISWIM, for an arbitrary closure $\langle t,\ \rho\rangle$, the term t is closed by the environment ρ. In Environmental MetaML, a closure $\blacklozenge\ t,\ \rho^*\ \blacklozenge$ is a pair of a term t and a meta-environment ρ^*. Suppose $\rho^* = \rho_1;\rho_2;...;\rho_m$. The free variables of the term t are bound by the first environment ρ_1. If the first environment ρ_1 does not close the term t, the free variables of the closure $\blacklozenge\ t,\ \rho_1\ \blacklozenge$ must be bound by the second environment ρ_2 and the free variables of the closure $\blacklozenge\ t,\ (\rho_1;\rho_2)\ \blacklozenge$ must be bound by the third environment ρ_3, and so on. The closure $\blacklozenge\ t,\ \rho^*\ \blacklozenge$ may also have free variables, which are bound by its surrounding context.

The design choice of closures and closure values are compatible with the original interpreter of MetaML introduced in [Tah99a]. To ensure that any variable that has been eliminated by some substitution or renaming does not escape from the scope of the substitution or renaming, they used a delayed environment called a cover. A cover works like a normal environment on non-function terms. If a cover encounters a function, the substitutions of the cover are delayed and are only performed on the result of calling the function. Analogously, in Environmental MetaML, environments on a level-0 lambda abstraction are delayed, as modelled by closure values. These environments work like normal environments on the result of applying the level-0 lambda abstraction.

Unlike the previous dialects of MetaML, Environmental MetaML deems the set of configurations rather than the set of (runtime) terms to be the fundamental set on which the operational semantics is defined. Recall that programs are closed level-0 source terms. To evaluate a program t_{s}^0, we first pair it with the initial meta-environment $(\rho_{\mathrm{init}}^{\mathrm{VAR}(t_{\mathrm{s}}^0)};\varepsilon)$, resulting in the initial configuration $c_{\mathrm{s}}^0 = \blacklozenge\ t_{\mathrm{s}}^0,\ (\rho_{\mathrm{init}}^{\mathrm{VAR}(t_{\mathrm{s}}^0)};\varepsilon)\ \blacklozenge$. We then pass the initial configuration to the operational semantics. Someone may wonder why the initial meta-environment is $(\rho_{\mathrm{init}}^{\mathrm{VAR}(t_{\mathrm{s}}^0)};\varepsilon)$ rather than $(\emptyset;\varepsilon)$. In short, choosing $(\rho_{\mathrm{init}}^{\mathrm{VAR}(t_{\mathrm{s}}^0)};\varepsilon)$ makes rules of the semantics consistent, eases proving equivalence of semantics and preserves the correctness of the semantics. We discuss its reasons in detail after introducing the evaluator for the semantics.

5.4.1.2 Free Variable Function

We define the free variable function by extending Substitutional MetaML's Definition 20 to accommodate configurations.

Definition 168 (Free Variable Function)**.** Let the free variable function FV be a total function from the set of configurations to the power set of variables.

$$FV : \text{CONF} \longrightarrow \mathscr{P}(\text{VAR})$$

$$\begin{aligned}
FV(x) &= x && (1)\\
FV(n) &= \emptyset && (2)\\
FV(c_1\ c_2) &= FV(c_1) \cup FV(c_2) && (3)\\
FV(\lambda x.c) &= FV(c) \backslash \{x\} && (4)\\
FV(\langle c \rangle) &= FV(c) && (5)\\
FV(!c) &= FV(c) && (6)\\
FV(\sim c) &= FV(c) && (7)\\
FV(c_1 + c_2) &= FV(c_1) \cup FV(c_2) && (8)\\
FV(\blacktriangleleft\ t,\ \varepsilon\ \blacktriangleright) &= FV(t) && (9)\\
FV(\blacktriangleleft\ t,\ \rho_1^*;\rho_2\ \blacktriangleright) &= \bigcup_i FV(\rho_2(x_i)) \text{ where } x_i \in FV(\blacktriangleleft\ t,\ \rho_1^*\ \blacktriangleright) && (10)\\
FV(\triangleleft\ \lambda x.t,\ \rho^*\ \triangleright) &= FV(\blacktriangleleft\ \lambda x.t,\ \rho^*\ \blacktriangleright) && (11)
\end{aligned}$$

Equations (1)-(2) are the same as Equations (1) and (7) of Definition 20 in Substitutional MetaML. Equations (3)-(8) are analogous to Equations (2)-(6) and (8) of Definition 20. Equations (9)-(11) are based on the definitions of closures and closure values.

5.4.2 Structural Operational Semantics

We lay out the structural operational semantics of Environmental MetaML through the level-indexed single-step relations $\longrightarrow^i$, the global single-step relation $\triangleright \longrightarrow$, the level-indexed multi-step relations $\longrightarrow^{i*}$ and the global multi-step relation $\triangleright \longrightarrow^*$.

Definition 169 (Level-indexed Single-step Relations)**.** For any $i \in \mathbb{N}$, let the level-indexed single-step relation $\longrightarrow^i$ be a 4-ary relation on the power set of variables, the power set of variables, the set of configurations at level i and the set of configurations at level i.

$$\longrightarrow^i \subseteq \mathscr{P}(\mathrm{VAR}) \times \mathscr{P}(\mathrm{VAR}) \times \mathrm{CONF}^i \times \mathrm{CONF}^i$$

$$\frac{\mathscr{V} \cup \{x\}; \mathscr{X} \vdash c_1^{i+1} \longrightarrow^{i+1} c_2^{i+1}}{\mathscr{V}; \mathscr{X} \vdash \lambda x.c_1^{i+1} \longrightarrow^{i+1} \lambda x.c_2^{i+1}} \text{ (lambda-(i+1))}$$

$$\frac{\mathscr{V}; \mathscr{X} \vdash c_{11}^i \longrightarrow^i c_{12}^i}{\mathscr{V}; \mathscr{X} \vdash c_{11}^i\ c_2^i \longrightarrow^i c_{12}^i\ c_2^i} \text{ (appL-i)} \qquad \frac{\mathscr{V}; \mathscr{X} \vdash c_{21}^i \longrightarrow^i c_{22}^i}{\mathscr{V}; \mathscr{X} \vdash v_1^i\ c_{21}^i \longrightarrow^i v_1^i\ c_{22}^i} \text{ (appR-i)}$$

$$\frac{}{\mathscr{V}; \mathscr{X} \vdash \triangleleft\ \lambda x.t^0,\ (\rho;\rho^*)\ \triangleright\ v^0 \longrightarrow^0 \blacktriangleleft\ t^0,\ (\rho[x \mapsto v^0];\rho^*)\ \blacktriangleright} \text{ (app-0)}$$

$$\frac{\mathscr{V}; \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i}{\mathscr{V}; \mathscr{X} \vdash !c_1^i \longrightarrow^i !c_2^i} \text{ (run-i)} \qquad \frac{}{\mathscr{V}; \mathscr{X} \vdash !\langle v^1 \rangle \longrightarrow^0 \blacktriangleleft\ v^1,\ (\rho_{\mathrm{init}}^{\mathscr{X}};\varepsilon)\ \blacktriangleright} \text{ (run-0)}$$

$$\frac{\mathscr{V}; \mathscr{X} \vdash c_1^{i+1} \longrightarrow^{i+1} c_2^{i+1}}{\mathscr{V}; \mathscr{X} \vdash \langle c_1^{i+1} \rangle \longrightarrow^i \langle c_2^{i+1} \rangle} \text{ (code-i)}$$

$$\frac{\mathscr{V}; \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i}{\mathscr{V}; \mathscr{X} \vdash \sim c_1^i \longrightarrow^{i+1} \sim c_2^i} \text{ (splice-(i+1))} \qquad \frac{}{\mathscr{V}; \mathscr{X} \vdash \sim\langle v^1 \rangle \longrightarrow^1 v^1} \text{ (splice-1)}$$

$$\frac{\mathscr{V}; \mathscr{X} \vdash c_{11}^i \longrightarrow^i c_{12}^i}{\mathscr{V}; \mathscr{X} \vdash c_{11}^i + c_2^i \longrightarrow^i c_{12}^i + c_2^i} \text{ (plusL-i)} \qquad \frac{\mathscr{V}; \mathscr{X} \vdash c_{21}^i \longrightarrow^i c_{22}^i}{\mathscr{V}; \mathscr{X} \vdash v_1^i + c_{21}^i \longrightarrow^i v_1^i + c_{22}^i} \text{ (plusR-i)}$$

$$\frac{}{\mathscr{V}; \mathscr{X} \vdash n_1 + n_2 \longrightarrow^0 n} \text{ where } n = n_1 + n_2 \text{ (plus-0)}$$

$$\frac{}{\mathscr{V}; \mathscr{X} \vdash \blacktriangleleft\ \lambda x.t^0,\ \rho^*\ \blacktriangleright \longrightarrow^0 \triangleleft\ \lambda x.t^0,\ \rho^*\ \triangleright} \text{ (lam-0-env)}$$

$$\mathscr{V}; \mathscr{X} \vdash \blacktriangleleft\ \lambda x.t^{i+1},\ (\rho;\rho^*)\ \blacktriangleright \longrightarrow^{i+1} \lambda x_N.\blacktriangleleft\ t^{i+1},\ (\rho[x \mapsto x_N][x_N \mapsto x_N];(\rho[x_N \mapsto x_N])^*)\ \blacktriangleright$$
where $x_N \notin \mathscr{X}$ (lam-(i+1)-env)

$$\frac{}{\mathscr{V}; \mathscr{X} \vdash \blacktriangleleft\ \triangleleft\ \lambda x.t,\ \rho_1^*\ \triangleright,\ \rho_2^*\ \blacktriangleright \longrightarrow^i \triangleleft\ \lambda x.t,\ (\rho_1^*;\rho_2^*)\ \triangleright} \text{ (clov-env)}$$

$$\frac{}{\mathscr{V}; \mathscr{X} \vdash \blacktriangleleft\ w,\ \varepsilon\ \blacktriangleright \longrightarrow^i w} \text{ (den-env)}$$

$$\frac{}{\mathscr{V}; \mathscr{X} \vdash \blacktriangleleft\ x,\ (\rho;\rho^*)\ \blacktriangleright \longrightarrow^i \blacktriangleleft\ \rho(x),\ \rho^*\ \blacktriangleright} \text{ (var-env)}$$

$$\frac{}{\mathscr{V}; \mathscr{X} \vdash \blacktriangleleft\ n,\ \rho^*\ \blacktriangleright \longrightarrow^i n} \text{ (num-env)}$$

$$\frac{}{\mathscr{V}; \mathscr{X} \vdash \blacktriangleleft\ t_1\ t_2,\ \rho^*\ \blacktriangleright \longrightarrow^i \blacktriangleleft\ t_1,\ \rho^*\ \blacktriangleright\blacktriangleleft\ t_2,\ \rho^*\ \blacktriangleright} \text{ (app-env)}$$

$$\frac{}{\mathscr{V}; \mathscr{X} \vdash \blacktriangleleft\ \langle t^{i+1} \rangle,\ \rho^*\ \blacktriangleright \longrightarrow^i \langle \blacktriangleleft\ t^{i+1},\ \rho^*\ \blacktriangleright \rangle} \text{ (code-env)}$$

$$\frac{}{\mathscr{V}; \mathscr{X} \vdash \blacktriangleleft\ !t^i,\ \rho^*\ \blacktriangleright \longrightarrow^i !\blacktriangleleft\ t^i,\ \rho^*\ \blacktriangleright} \text{ (run-env)}$$

$$\frac{}{\mathscr{V}; \mathscr{X} \vdash \blacktriangleleft\ \sim t^i,\ \rho^*\ \blacktriangleright \longrightarrow^{i+1} \sim\blacktriangleleft\ t^i,\ \rho^*\ \blacktriangleright} \text{ (splice-env)}$$

$$\frac{}{\mathscr{V}; \mathscr{X} \vdash \blacktriangleleft\ t_1 + t_2,\ \rho^*\ \blacktriangleright \longrightarrow^i \blacktriangleleft\ t_1,\ \rho^*\ \blacktriangleright + \blacktriangleleft\ t_2,\ \rho^*\ \blacktriangleright} \text{ (plus-env)}$$

The level-indexed single-step relation $\mathscr{V};\mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$ reads as "c_1 single-steps to c_2 at level i bound by $\mathscr{V}$ and $\mathscr{X}$".

To make a small-step when evaluating a program, we need to repeatedly apply the structural rules until we find a configuration of the program on which a reduction rule can be applied. In the meanwhile, we keep track of lambda bound variables when we dive into the body of a level-(i+1) non-value lambda using the (lambda-(i+1)) rule. Suppose we have found such a configuration c_1^i that is reducible and we have $\mathscr{V};\mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$. The free variables of c_1^i must be bound by lambda bound variables of level-(i+1) non-value lambdas in the surrounding scope, which are tracked by the variable set $\mathscr{V}$. The variable set $\mathscr{X}$ records all variables in the configuration c_1^i and its surrounding scope. We need the variable set $\mathscr{X}$ to determine whether a variable is fresh in the sense that it has not appeared in the configuration being currently evaluated or in its surrounding scope. The variable sets $\mathscr{V}$ and $\mathscr{X}$ help specify the well-boundness property of configurations (Proposition 177).

We briefly explain some rules of the single-step relation. (1) The (app-0) rule models an application by updating the environment, which completes the unfinished job of Suspended MetaML's (app-0) rule. (2) The (run-0) rule is about executing a code value at level 0. Someone may want to replace the configuration $\blacklozenge\, v^1,\ (\rho_{\text{init}}^{\mathscr{X}};\varepsilon)\ \blacklozenge$ with the value v^1. This is incorrect. The single-step relation is defined on configurations. The value v^1 must be paired with a meta-environment in order to be evaluated at level 0. (3) The (*-env) rules discuss how to evaluate a closure. (3.1) The (lam-0-env) rule turns a closure into a closure value. (3.2) The (lam-(i+1)-env) rule combines Suspended MetaML's (lambda-(i+1)-t) rule and (lam-subst) rule. To evaluate a lambda abstraction at a level higher than 0, we must rename the lambda bound variable to a fresh variable before diving into the body, where the fresh variable must have not occurred in the current configuration being evaluated or in its surrounding scope. (3.3) The (clov-env) rule concatenates two meta-environments. (3.4) The (den-env) rule is trivial. (3.5) The other (*-env) rules correspond to Suspended MetaML's single-step substitution reduction relations.

Definition 170 (Global Single-step Relation)**.** Let the global single-step relation $\triangleright \longrightarrow$ be a binary relation between the set of level-0 configurations and the set of level-0 configurations.

$$\triangleright\, c_1^0 \longrightarrow c_2^0 \qquad \text{if and only if} \qquad \emptyset;\textsc{Var}(c_1^0) \vdash c_1^0 \longrightarrow^0 c_2^0$$

The global single-step relation $\triangleright\, c_1^0 \longrightarrow c_2^0$ reads as "c_1 single-steps to c_2".

Definition 171 (Level-indexed Multi-step Relations)**.** For any $i \in \mathbb{N}$, let the level-indexed multi-step relation $\longrightarrow^{i*}$ be a 4-ary relation on the power set of variables, the power set of variables, the set of configurations at level i and the set of configurations at level i directly based on the level-indexed single-step relation $\longrightarrow^i$.

$$\longrightarrow^{i*} \subseteq \mathscr{P}(\textsc{Var}) \times \mathscr{P}(\textsc{Var}) \times \textsc{Conf}^i \times \textsc{Conf}^i$$

$$\frac{}{\mathscr{V};\mathscr{X} \vdash c_1^i \longrightarrow^{i*} c_2^i}\ \text{where } \mathscr{V};\mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i\ \text{(step)} \qquad \frac{}{\mathscr{V};\mathscr{X} \vdash c^i \longrightarrow^{i*} c^i}\ \text{(refl)}$$

$$\frac{\mathscr{V};\mathscr{X} \vdash\ c_1^i \longrightarrow^{i*} c_2^i \quad \mathscr{V};\mathscr{X} \cup \textsc{Var}(c_2^i) \vdash c_2^i \longrightarrow^{i*} c_3^i}{\mathscr{V};\mathscr{X} \vdash c_1^i \longrightarrow^{i*} c_3^i}\ \text{(trans)}$$

Definition 172 (Global Multi-step Relation)**.** Let the global multi-step relation $\triangleright \longrightarrow^*$ be the reflexive-transitive closure of the global single-step relation $\triangleright \longrightarrow$.

Example 173. Consider $!\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle$. We first construct a configuration that pairs the above term with an initial meta-environment:

$$\blacklozenge\ !\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle,\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge$$

By the structural operational semantics of Environmental MetaML, we have:

$$
\begin{array}{ll}
 & \blacklozenge\ !\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle,\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge \\
\triangleright \longrightarrow & !\blacklozenge\ \langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle,\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge \\
\triangleright \longrightarrow & !\langle\blacklozenge\ \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0),\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge\rangle \\
\triangleright \longrightarrow & !\langle\lambda z.\blacklozenge\ \sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge\rangle \qquad \text{where } z\notin\{x,y\} \\
\triangleright \longrightarrow & !\langle\lambda z.(\blacklozenge\ \sim((\lambda x.\langle x\rangle)\ (\lambda x.y)),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
\triangleright \longrightarrow & !\langle\lambda z.(\sim\blacklozenge\ ((\lambda x.\langle x\rangle)\ (\lambda x.y)),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
\triangleright \longrightarrow & !\langle\lambda z.(\sim(\blacklozenge\ (\lambda x.\langle x\rangle),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge\blacklozenge\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
\triangleright \longrightarrow & !\langle\lambda z.(\sim(\triangleleft\ (\lambda x.\langle x\rangle),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \triangleright\blacklozenge\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
\triangleright \longrightarrow & !\langle\lambda z.(\sim(\triangleleft\ (\lambda x.\langle x\rangle),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \triangleright\triangleleft\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \triangleright)\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
\triangleright \longrightarrow & !\langle\lambda z.(\sim(\blacklozenge\ \langle x\rangle,\ (\{(x,\triangleleft\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \triangleright),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
\triangleright \longrightarrow & !\langle\lambda z.(\sim\langle\blacklozenge\ x,\ (\{(x,\triangleleft\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \triangleright),(y,z),(z,z)\};\varepsilon)\ \blacklozenge\rangle\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
\triangleright \longrightarrow & !\langle\lambda z.(\sim\langle\blacklozenge\ \triangleleft\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \triangleright,\ (\varepsilon)\ \blacklozenge\rangle\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
\triangleright \longrightarrow & !\langle\lambda z.(\sim\langle\triangleleft\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \triangleright\rangle\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
\triangleright \longrightarrow & !\langle\lambda z.(\triangleleft\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \triangleright\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
\triangleright \longrightarrow & !\langle\lambda z.(\triangleleft\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \triangleright\ 0)\rangle \\
\triangleright \longrightarrow & \blacklozenge\ \lambda z.(\triangleleft\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \triangleright\ 0),\ (\{(x,x),(y,y),(z,z)\};\varepsilon)\ \blacklozenge \\
\triangleright \longrightarrow & \triangleleft\ \lambda z.(\triangleleft\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \triangleright\ 0),\ (\{(x,x),(y,y),(z,z)\};\varepsilon)\ \triangleright
\end{array}
$$

Properties. Our sole purpose of developing the structural operational semantics is to evaluate programs which are closed level-0 source terms. To evaluate a program t_{s}^0 at level 0, we construct the initial configuration $c_{\mathrm{s}}^0 = \blacklozenge\ t_{\mathrm{s}}^0,\ (\rho_{\mathrm{init}}^{\mathrm{VAR}(t_{\mathrm{s}}^0)};\varepsilon)\ \blacklozenge$ and pass it to the structural operational semantics. We expect that the initial configuration multi-steps to a level-0 value v^0. Suppose the closed configuration c_1^0 is an intermediate result of evaluating the initial configuration c_{s}^0. We then repeatedly apply the structural rules on c_1^0 until we find its subconfiguration c_{11} on which a reduction rule can apply. We want to ensure that all free variables of c_{11} are bound in its surrounding scope by lambda bound variables of level-(i+1) non-value lambdas. We define the following judgement to specify the well-boundness of a configuration. We demonstrate how this judgement helps prove the correctness of the (app-0) rule.

Definition 174 (Well-boundness Judgement)**.** Let the well-boundness judgement $\vdash\ wb$ be a ternary relation on the power set of variables, the power set of variables and the set of configurations.

$$\vdash\ wb\ \subseteq\ \mathscr{P}(\textsc{Var}) \times \mathscr{P}(\textsc{Var}) \times \textsc{Conf}$$

$$\frac{}{\mathscr{U};\mathscr{V} \vdash x\ wb}\ \text{where } x \in \mathscr{U}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash c_1\ wb \quad \mathscr{U};\mathscr{V} \vdash c_2\ wb}{\mathscr{U};\mathscr{V} \vdash c_1\ c_2\ wb}$$

$$\frac{\mathscr{U} \cup \{x\};\mathscr{V} \vdash c\ wb}{\mathscr{U};\mathscr{V} \vdash \lambda x.c^0\ wb}\ \text{where } x \notin \mathscr{V}$$

$$\frac{\mathscr{U} \cup \{x\};\mathscr{V} \vdash c\ wb}{\mathscr{U};\mathscr{V} \vdash \lambda x.v^{i+1}\ wb}\ \text{where } x \notin \mathscr{V}$$

$$\frac{\mathscr{U} \cup \{x\};\mathscr{V} \cup \{x\} \vdash c\ wb}{\mathscr{U};\mathscr{V} \vdash \lambda x.c^{i+1}\ wb}\ \text{where } c^{i+1} \notin \textsc{Value}^{i+1} \text{ and } x \notin \mathscr{V}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash \langle t \rangle\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash {\sim} t\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash !t\ wb}$$

$$\frac{}{\mathscr{U};\mathscr{V} \vdash n\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t_1\ wb \quad \mathscr{U};\mathscr{V} \vdash t_2\ wb}{\mathscr{U};\mathscr{V} \vdash t_1 + t_2\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash \blacktriangleleft t,\ \varepsilon \blacktriangleright\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash \rho_m(x_{m_k})\ wb \quad \mathscr{U} \cup \{\overline{x_{m_k}}\};\mathscr{V} \vdash \blacktriangleleft t,\ (\overline{\rho_{i1}^{m-1}};\varepsilon) \blacktriangleright\ wb}{\mathscr{U};\mathscr{V} \vdash \blacktriangleleft t,\ (\overline{\rho_{i1}^{m}};\varepsilon) \blacktriangleright\ wb}$$

where $\textsc{Var}(\blacktriangleleft t,\ \overline{\rho_{i1}^{m}} \blacktriangleright) \subseteq \mathrm{dom}(\rho_i)$ for any ρ_i, $\rho_i(y_j) = y_j$ for any ρ_i and $y_j \in \mathscr{V}$, and $x_{m_k} \in FV(\blacktriangleleft t,\ (\overline{\rho_{i1}^{m-1}};\varepsilon) \blacktriangleright)$.

$$\frac{\mathscr{U};\mathscr{V} \vdash \blacktriangleleft t,\ \overline{\rho_i} \blacktriangleright\ wb}{\mathscr{U};\mathscr{V} \vdash \triangleleft t,\ \overline{\rho_i} \triangleright\ wb}$$

Notation 175. A sequence of environments, $\rho_1;\rho_2;...;\rho_m$, can be abbreviated as $\overline{\rho_{i1}^{m}}$ or $\overline{\rho_i}$.

The well-boundness judgement $\mathscr{U};\mathscr{V} \vdash c\ wb$ reads as "c is well bound by $\mathscr{U}$ and $\mathscr{V}$". The variable set $\mathscr{U}$ tracks all free variables of the configuration c that are bound any means in the surrounding scope. We observe the following property which is analogous to Suspended MetaML's Proposition 160.

Proposition 176. *If $\mathscr{U};\mathscr{V} \vdash c\ wb$, then $FV(c) \subseteq \mathscr{U}$.*

We come back to our discussion of the well-boundness of the subconfiguration c_{11} of the configuration c_1^0. By the definition of well-boundness judgement, we have $\emptyset;\emptyset \vdash c_1^0\ wb$. Observe that a sub-derivation of $\emptyset;\emptyset \vdash c_1^0\ wb$ must be the derivation of $\mathscr{U};\mathscr{V} \vdash c_{11}\ wb$ for some variable sets $\mathscr{U}$ and $\mathscr{V}$. The variable

set $\mathscr{V}$ tracks all free variables of the configuration c_{11} that are bound in the surrounding scope by lambda bound variables of level-$(i+1)$ non-value lambdas. Recall that other than level-$(i+1)$ non-value lambdas, the structural rules do not allow evaluating under lambdas or closures. Thus the variable set $\mathscr{U}$ is the same as $\mathscr{V}$. We have $\mathscr{V};\mathscr{V} \vdash c_{11}\ wb$.

Consider a special case of the configuration c_{11}. Suppose c_{11} is an application $\triangleleft\ \lambda x.t^0,\ (\overline{\rho_{i1}}^m;\varepsilon)\ \triangleright\ v^0$ that is reducible by the (app-0) rule. The free variables of v^0 must be bound by lambda bound variables of level-$(i+1)$ non-value lambdas in its surrounding context, which are tracked by the variable set $\mathscr{V}$ of the judgement $\mathscr{V};\mathscr{V} \vdash \triangleleft\ \lambda x.t^0,\ (\overline{\rho_{i1}}^m;\varepsilon)\ \triangleright\ v^0\ wb$. By definition, $\rho_i(y_j) = y_j$ for any $y_j \in \mathscr{V}$. By Proposition 176, $FV(v^0) \subseteq \mathscr{V}$. Hence, $\rho_i(y_v) = y_v$ for any $y_v \in FV(v^0)$, which we call the well-boundness of the application $\triangleleft\ \lambda x.t^0,\ (\overline{\rho_{i1}}^m;\varepsilon)\ \triangleright\ v^0$.

We briefly explain how the correctness of the (app-0) rule is guaranteed by the well-boundness of the application $\triangleleft\ \lambda x.t^0,\ (\overline{\rho_{i1}}^m;\varepsilon)\ \triangleright\ v^0$. Let's represent each environment as a set of variable-and-denotable-term pairs. We have:

$$\overline{\rho_{i1}}^m = \{\overline{(x_{1i}, w_{1i})}\};\{\overline{(x_{2i}, w_{2i})}\};...;\{\overline{(x_{mi}, w_{mi})}\}$$

Then the application $\triangleleft\ \lambda x.t^0,\ (\overline{\rho_{i1}}^m;\varepsilon)\ \triangleright\ v^0$ denotes

$$(\lambda x.t^0)\{\overline{[w_{1i}/x_{1i}]}\}\{\overline{[w_{2i}/x_{2i}]}\}...\{\overline{[w_{mi}/x_{mi}]}\}\ v^0$$

where each braced set of substitutions works as an environment and substitutions in each braced set are unordered. We first rename the lambda bound variable x to a globally fresh variable x_N and then push all braced sets of substitutions under the new lambda. We get:

$$\lambda x_N.(t^0\{\overline{[w_{1i}/x_{1i}]}[x_N/x]\}\{\overline{[w_{2i}/x_{2i}]}[x_N/x_N]\}...\{\overline{[w_{mi}/x_{mi}]}[x_N/x_N]\})\ v^0$$

Then we perform the application and get:

$$t^0\{\overline{[w_{1i}/x_{1i}]}[x_N/x]\}\{\overline{[w_{2i}/x_{2i}]}[x_N/x_N]\}...\{\overline{[w_{mi}/x_{mi}]}[x_N/x_N]\}\{[v^0/x_N]\}$$

By the well-boundness of the application $\triangleleft\ \lambda x.t^0,\ (\overline{\rho_{i1}}^m;\varepsilon)\ \triangleright\ v^0$, we know for any $y_v \in FV(v^0)$, y_v is substituted by itself in the first m braced sets of substitutions. Then we can splice the last braced substitution into the first braced set and eliminate renaming the lambda bound variable. We get

$$t^0\{\overline{[w_{1i}/x_{1i}]}[v^0/x]\}\{\overline{[w_{2i}/x_{2i}]}\}...\{\overline{[w_{mi}/x_{mi}]}\}$$

which corresponds to

$$\blacktriangleleft\ t^0,\ (\rho_1[x \mapsto v^0];\overline{\rho_2}^m;\varepsilon)\ \blacktriangleright$$

in the (app-0) rule of Environmental MetaML.

The well-boundness judgement cooperates well with the multi-step relation. The following property says the former is preserved by the latter. It is analogous to Suspended MetaML's Proposition 161.

Proposition 177. *If* $\mathscr{V};\mathscr{V} \vdash c_1^i\ wb$, $\textsc{Var}(c_1^i) \subseteq \mathscr{X}$, $\mathscr{V} \subseteq \mathscr{X}$ *and* $\mathscr{V};\mathscr{X} \vdash c_1^i \longrightarrow^{i*} c_2^i$, *then* $\mathscr{V};\mathscr{V} \vdash c_2^i\ wb$.

As a corollary of the above properties, the multi-step relation preserves the closedness of configurations.

Evaluator. We now define an evaluator in terms of the structural operational semantics of Environmental MetaML. Environmental MetaML's multi-step relation is defined on sets of configurations rather than sets of runtime terms. Given a program t, the evaluator applies the multi-step relation on the initial configuration $\lhd\, t,\ (\rho_{\text{init}}^{\text{VAR}(t)};\varepsilon)\ \rhd$ which pairs the program with the initial environment of the program. The evaluator is otherwise analogous to the one defined for Suspended MetaML.

Definition 178 (Evaluator based on Structural Operational Semantics of Environmental MetaML)**.** Let the evaluator $eval_{\text{MetaML:EnvSOS}}$ be a partial function from the set of programs $\text{PRGM}_{\text{MetaML}}$ to the set of answers $\text{ANS}_{\text{MetaML}}$.

$$eval_{\text{MetaML:EnvSOS}} : \text{PRGM}_{\text{MetaML}} \rightharpoonup \text{ANS}_{\text{MetaML}}$$

$$eval_{\text{MetaML:EnvSOS}}(t) = \begin{cases} \texttt{function} & \text{if } \triangleright \lhd t,\ (\rho_{\text{init}}^{\text{VAR}(t)};\varepsilon)\ \rhd \longrightarrow^* \langle\!|\ \lambda x.t'^{0},\ \rho^*\ |\!\rangle \\ \texttt{code} & \text{if } \triangleright \lhd t,\ (\rho_{\text{init}}^{\text{VAR}(t)};\varepsilon)\ \rhd \longrightarrow^* \langle v^1 \rangle \\ n & \text{if } \triangleright \lhd t,\ (\rho_{\text{init}}^{\text{VAR}(t)};\varepsilon)\ \rhd \longrightarrow^* n \end{cases}$$

This evaluator is defined in terms of the structural operational semantics of Environmental MetaML. The subscript "$_{\text{MetaML:EnvSOS}}$" in $eval_{\text{MetaML:EnvSOS}}$ denotes the structural operational semantics of Environmental MetaML.

Someone may wonder why the initial configuration has the meta-environment $(\rho_{\text{init}}^{\text{VAR}(t_s^0)};\varepsilon)$ rather than $(\emptyset;\varepsilon)$. First of all, for any program t, the initial configuration $\lhd\, t,\ \rho^*\ \rhd$ is closed regardless of whether $\rho^* = (\rho_{\text{init}}^{\text{VAR}(t_s^0)};\varepsilon)$ or $\rho^* = (\emptyset;\varepsilon)$. Choosing $(\rho_{\text{init}}^{\text{VAR}(t_s^0)};\varepsilon)$ still preserves the closedness of the initial configuration. Secondly, as implied by the theorem at the end of this section, choosing $(\rho_{\text{init}}^{\text{VAR}(t_s^0)};\varepsilon)$ over $(\emptyset;\varepsilon)$ does not destroy the soundness or completeness of the semantics. Thirdly, having the meta-environment $(\rho_{\text{init}}^{\text{VAR}(t_s^0)};\varepsilon)$ in the initial configuration is consistent with the (run-0) rule where the newly constructed configuration has the meta-environment $(\rho_{\text{init}}^{\mathscr{X}};\varepsilon)$. Fourthly, by the well-boundness judgement, given a well bound closure $\lhd\, t,\ \rho^*\ \rhd$ where $\rho^* = \overline{\rho_i}$, we must have $\text{VAR}(\lhd\, t,\ \overline{\rho_i}\ \rhd) \subseteq \text{dom}(\rho_i)$ for any ρ_i. If we do not choose $(\rho_{\text{init}}^{\text{VAR}(t_s^0)};\varepsilon)$ over $(\emptyset;\varepsilon)$, we need a more complicated well-boundness judgement that may obscure the fundamental concepts, making proving semantics equivalence more complicated. In Section 7.2, we propose a novel way of modelling environments, which no long requires the ad-hoc step of collecting all the variables that exist in the program for the initial configuration as $(\rho_{\text{init}}^{\text{VAR}(t_s^0)};\varepsilon)$.

We claim that the evaluators defined in terms of Substitutional MetaML and Environmental MetaML are equivalent.

Theorem 179 (Kleene Equality of Evaluators)**.** *For any $t \in \text{PRGM}_{\text{MetaML}}$, $eval_{\text{MetaML:SubSOS}}(t)$ is Kleene equal to $eval_{\text{MetaML:EnvSOS}}(t)$.*

We prove the theorem in appendices.

5.5 Environmental MetaML - Reduction Semantics

Following the path of refining a structural operational semantics to its corresponding reduction semantics as presented in Sections 3.5 and 4.1, we refine Environmental MetaML's structural operational semantics to a reduction semantics.

5.5.1 Syntax

The definitions of source terms, runtime terms, values, denotable terms, configurations and meta-environments are the same as Section 5.4.

5.5.1.1 Evaluation Contexts

Section 4.1 defined evaluation contexts for Substitutional MetaML. Unlike Substitutional MetaML, Environmental MetaML defines its semantics on configurations rather than on terms. Taking this difference into consideration, we define evaluation contexts for Environmental MetaML, which is analogous to Substitutional MetaML's Definition 114.

Definition 180 (Evaluation Contexts: Inside-out)**.** Let $\textsc{Ecxt}^{i-\circ j}$ be the set of evaluation contexts with inner level i and outer level j.

$$E^{i-\circ j} \in \textsc{Ecxt}^{i-\circ j},\ c^i \in \textsc{Conf}^i,\ v^i \in \textsc{Value}^i$$

$$\frac{}{\Box \in \textsc{Exct}^{i-\circ i}}\ \text{(ept-i)}$$

$$\frac{E \in \textsc{Exct}^{i-\circ j}}{E[\Box\ c_2^i] \in \textsc{Exct}^{i-\circ j}}\ \text{(appL-i)} \qquad \frac{E \in \textsc{Exct}^{i-\circ j}}{E[v_1^i\ \Box] \in \textsc{Exct}^{i-\circ j}}\ \text{(appR-i)}$$

$$\frac{E \in \textsc{Exct}^{(i+1)-\circ j}}{E[\lambda x.\Box] \in \textsc{Exct}^{(i+1)-\circ j}}\ \text{(lambda-(i+1))}$$

$$\frac{E \in \textsc{Exct}^{i-\circ j}}{E[\langle\Box\rangle] \in \textsc{Exct}^{(i+1)-\circ j}}\ \text{(code-i)} \qquad \frac{E \in \textsc{Exct}^{(i+1)-\circ j}}{E[\sim\Box] \in \textsc{Exct}^{i-\circ j}}\ \text{(splice-(i+1))} \qquad \frac{E \in \textsc{Exct}^{i-\circ j}}{E[!\Box] \in \textsc{Exct}^{i-\circ j}}\ \text{(run-i)}$$

$$\frac{E \in \textsc{Exct}^{i-\circ j}}{E[\Box + c_2^i] \in \textsc{Exct}^{i-\circ j}}\ \text{(plusL-i)} \qquad \frac{E \in \textsc{Exct}^{i-\circ j}}{E[v_1^i + \Box] \in \textsc{Exct}^{i-\circ j}}\ \text{(plusR-i)}$$

5.5.2 Reduction Semantics

We lay out the reduction semantics through the level-indexed notions of reduction $\mathscr{R}^i$, the level-indexed single-reduction relations $\longmapsto^i$, the global single-reduction relation $\triangleright \longmapsto$, the level-indexed multi-reduction relations $\longmapsto^{i*}$ and the global multi-reduction relation $\triangleright \longmapsto^*$.

Definition 181 (Level-indexed Notions of Reduction)**.** For any $i \in \mathbb{N}$, let the notions of reduction $\mathscr{R}^i$ be a binary relation between the set of configurations at level i and the set of configurations at level i.

$$\mathscr{R}^i \subseteq \mathscr{P}(\textsc{Var}) \times \textsc{Conf}^i \times \textsc{Conf}^i$$

$\mathscr{X} \vdash$	$\langle\!\vert\, \lambda x.t^0,\ (\rho;\rho^*) \,\vert\!\rangle\ v^0$	$\mathscr{R}^0$	$\lfloor\!\vert\, t^0,\ (\rho[x \mapsto v^0];\rho^*) \,\vert\!\rfloor$	(app-0)
$\mathscr{X} \vdash$	$!\langle v^1 \rangle$	$\mathscr{R}^0$	$\lfloor\!\vert\, v^1,\ (\rho^{\mathscr{X}}_{\text{init}};\varepsilon) \,\vert\!\rfloor$	(run-0)
$\mathscr{X} \vdash$	$\sim\langle v^1 \rangle$	$\mathscr{R}^1$	v^1	(splice-1)
$\mathscr{X} \vdash$	$n_1 + n_2$	$\mathscr{R}^0$	n where $n = n_1 + n_2$	(plus-0)
$\mathscr{X} \vdash$	$\lfloor\!\vert\, \lambda x.t^0,\ \rho^* \,\vert\!\rfloor$	$\mathscr{R}^0$	$\langle\!\vert\, \lambda x.t^0,\ \rho^* \,\vert\!\rangle$	(conf-lam-0)
$\mathscr{X} \vdash$	$\lfloor\!\vert\, \lambda x.t^{i+1},\ (\rho;\rho^*) \,\vert\!\rfloor$	$\mathscr{R}^{i+1}$	$\lambda x_N.\lfloor\!\vert\, t^{i+1},\ (\rho[x \mapsto x_N];(\rho[x_N \mapsto x_N])^* \,\vert\!\rfloor$ where $x_N \notin \mathscr{X}$	(conf-lam-(i+1))
$\mathscr{X} \vdash$	$\lfloor\!\vert\, \langle\!\vert\, \lambda x.t,\ \rho_1^*,\ \rho_2^* \,\vert\!\rangle \,\vert\!\rfloor$	$\mathscr{R}^i$	$\langle\!\vert\, \lambda x.t,\ (\rho_1^*;\rho_2^*) \,\vert\!\rangle$	(conf-clov-i)
$\mathscr{X} \vdash$	$\lfloor\!\vert\, \omega,\ \varepsilon \,\vert\!\rfloor$	$\mathscr{R}^i$	w	(conf-den-i)
$\mathscr{X} \vdash$	$\lfloor\!\vert\, x,\ (\rho;\rho^*) \,\vert\!\rfloor$	$\mathscr{R}^i$	$\lfloor\!\vert\, \rho(x),\ \rho^* \,\vert\!\rfloor$	(conf-var-i)
$\mathscr{X} \vdash$	$\lfloor\!\vert\, n,\ \rho^* \,\vert\!\rfloor$	$\mathscr{R}^i$	n	(conf-num-i)
$\mathscr{X} \vdash$	$\lfloor\!\vert\, t_1\ t_2,\ \rho^* \,\vert\!\rfloor$	$\mathscr{R}^i$	$\lfloor\!\vert\, t_1,\ \rho^* \,\vert\!\rfloor \lfloor\!\vert\, t_2,\ \rho^* \,\vert\!\rfloor$	(conf-app-i)
$\mathscr{X} \vdash$	$\lfloor\!\vert\, \langle t^{i+1} \rangle,\ \rho^* \,\vert\!\rfloor$	$\mathscr{R}^i$	$\langle \lfloor\!\vert\, t^{i+1},\ \rho^* \,\vert\!\rfloor \rangle$	(conf-code-i)
$\mathscr{X} \vdash$	$\lfloor\!\vert\, !t^i,\ \rho^* \,\vert\!\rfloor$	$\mathscr{R}^i$	$!\lfloor\!\vert\, t^i,\ \rho^* \,\vert\!\rfloor$	(conf-run-i)
$\mathscr{X} \vdash$	$\lfloor\!\vert\, \sim t^i,\ \rho^* \,\vert\!\rfloor$	$\mathscr{R}^{i+1}$	$\sim\lfloor\!\vert\, t^i,\ \rho^* \,\vert\!\rfloor$	(conf-splice-i)
$\mathscr{X} \vdash$	$\lfloor\!\vert\, t_1 + t_2,\ \rho^* \,\vert\!\rfloor$	$\mathscr{R}^i$	$\lfloor\!\vert\, t_1,\ \rho^* \,\vert\!\rfloor + \lfloor\!\vert\, t_2,\ \rho^* \,\vert\!\rfloor$	(conf-plus-i)

The notion of reduction $\mathscr{X} \vdash c_1^i\ \mathscr{R}^i\ c_2^i$ reads as "c_1 reduces to c_2 at level i bound by $\mathscr{X}$". Each notion corresponds to one reduction rule of the single-step relations presented in Definition 169. The variable set $\mathscr{X}$ records all variables in the configuration c_1^i and in its surrounding context, which is used in the (conf-lam-(i+1)) rule to determine whether a variable is globally fresh.

Definition 182 (Level-indexed Single-reduction Relation)**.** For any $i \in \mathbb{N}$, let the level-indexed single-reduction relation $\longmapsto^i$ be a binary relation between the set of configurations at level i and the set of configurations at level i directly based on the notions of reduction $\mathscr{R}^j$.

$$\longmapsto^i \subseteq \textsc{Conf}^i \times \textsc{Conf}^i$$

$$\frac{\mathscr{X} \vdash t_1^j\ \mathscr{R}^j\ t_2^j}{\mathscr{X} \vdash E^{j-\circ i}[t_1^j] \longmapsto^i E^{j-\circ i}[t_2^j]}$$

The level-indexed single-reduction relation $\mathscr{X} \vdash c_1^i \longmapsto^i c_2^i$ reads as "c_1 single-reduces to c_2 at level i bound by $\mathscr{X}$". The above definition states that the single-reduction relation respects performing any notion of reduction in an evaluation context.

Definition 183 (Global Single-reduction Relation)**.** Let the global single-reduction relation $\triangleright \longmapsto$ be a binary relation between the set of level-0 configurations and the set of level-0 configurations.

$$\triangleright\, c_1^0 \longmapsto c_2^0 \qquad \text{if and only if} \qquad \textsc{Var}(c_1^0) \vdash c_1^0 \longmapsto^0 c_2^0$$

The global single-reduction relation $\triangleright\, c_1^0 \longrightarrow c_2^0$ reads as "c_1 single-reduces to c_2". It is defined based on the single-reduction relation with a particular initialisation of the variable set $\mathscr{X}$.

Definition 184 (Level-indexed Multi-reduction Relations)**.** For any $i \in \mathbb{N}$, let the level-indexed multi-reduction relation $\longmapsto^{i*}$ be the a binary relation between the set of configurations at level i and the set of configurations at level i directly based on the level-indexed single-reduction relation $\longmapsto^{i}$.

$$\longmapsto^{i*} \subseteq \mathscr{P}(\text{VAR}) \times \text{CONF}^i \times \text{CONF}^i$$

$$\frac{}{\mathscr{X} \vdash c_1^i \longmapsto^{i*} c_2^i} \text{ where } \mathscr{X} \vdash c_1^i \longmapsto^{i} c_2^i \text{ (step)} \qquad \frac{}{\mathscr{X} \vdash c^i \longmapsto^{i*} c^i} \text{ (refl)}$$

$$\frac{\mathscr{X} \vdash \ c_1^i \longmapsto^{i*} c_2^i \quad \mathscr{X} \cup \text{VAR}(c_2^i) \vdash c_2^i \longmapsto^{i*} c_3^i}{\mathscr{X} \vdash c_1^i \longmapsto^{i*} c_3^i} \text{ (trans)}$$

Definition 185 (Global Multi-reduction Relation)**.** Let the global multi-reduction relation $\triangleright \longrightarrow^*$ be the reflexive-transitive closure of the global single-reduction relation $\triangleright \longrightarrow$.

Example 186. Consider $!\langle \lambda y.(\sim((\lambda x.\langle x \rangle)\ (\lambda x.y))\ 0) \rangle$.

We first construct a configuration that pairs the above term with an initial meta-environment:

$$\blacklozenge\ !\langle \lambda y.(\sim((\lambda x.\langle x \rangle)\ (\lambda x.y))\ 0) \rangle,\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge$$

By the reduction semantics of Environmental MetaML, we have:

$$\begin{array}{rl} \triangleright & \blacklozenge\ !\langle \lambda y.(\sim((\lambda x.\langle x \rangle)\ (\lambda x.y))\ 0) \rangle,\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge \\ \longmapsto^* & \triangleleft\ \lambda z.(\triangleleft\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \triangleright\ 0),\ (\{(x,x),(y,y),(z,z)\};\varepsilon)\ \triangleright \end{array}$$

as demonstrated in Figure 5.1.

A comparison of the Figure 5.1 with Figures 3.1 and 4.1 tells that Environmental MetaML's reduction semantics follows the exact same three-step break-apply-plug pattern of evaluating a program as Environmental ISWIM's reduction semantics and Substitutional MetaML's reduction semantics.

Property. The global multi-reduction relation preserves the closedness of a configuration. This is the same property that Environmental MetaML's structural operational semantics holds.

Proposition 187 (Closedness of Configurations)**.** *If* $\triangleright\, c_1^0 \longmapsto^* c_2^0$ *and* $FV(c_1^0) = \emptyset$*, then* $FV(c_2^0) = \emptyset$*.*

Evaluator. We now define an evaluator based on the reduction semantics of Environmental MetaML. The evaluator is analogous to the evaluators defined based on the structural operational semantics of Environmental MetaML.

Definition 188 (Evaluator based on Reduction Semantics of Environmental MetaML)**.** Let the evaluator $eval_{\text{MetaML:EnvRed}}$ be a partial function from the set of programs $\text{PRGM}_{\text{MetaML}}$ to the set of answers $\text{ANS}_{\text{MetaML}}$.

$$
\begin{array}{ll}
 & \blacklozenge\ !\langle\lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle,\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge \\
= & \Box^{0\multimap 0}[\blacklozenge\ !\langle\lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle,\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge] \\
\triangleright\!\!\mapsto & \Box^{0\multimap 0}[!\blacklozenge\ \langle\lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle,\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge] \\
= & !\blacklozenge\ \langle\lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle,\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge \\
= & (\Box[!\Box])^{0\multimap 0}[\blacklozenge\ \langle\lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle,\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge] \\
\triangleright\!\!\mapsto & (\Box[!\Box])^{0\multimap 0}[\langle\blacklozenge\ \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0),\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge\rangle] \\
= & !\langle\blacklozenge\ \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0),\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge\rangle \\
= & (\Box[!\Box][\langle\Box\rangle])^{1\multimap 0}[\blacklozenge\ \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0),\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge] \\
\triangleright\!\!\mapsto & (\Box[!\Box][\langle\Box\rangle])^{1\multimap 0}[\lambda z.\blacklozenge\ \sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge] \qquad \text{where } z\notin\{x,y\} \\
= & !\langle\lambda z.\blacklozenge\ \sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge\rangle \\
= & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box])^{1\multimap 0}[\blacklozenge\ \sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge] \\
\triangleright\!\!\mapsto & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box])^{1\multimap 0}[\blacklozenge\ \sim((\lambda x.\langle x\rangle)\ (\lambda x.y)),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge] \\
= & !\langle\lambda z.(\blacklozenge\ \sim((\lambda x.\langle x\rangle)\ (\lambda x.y)),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
= & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge])^{1\multimap 0}[\blacklozenge\ \sim((\lambda x.\langle x\rangle)\ (\lambda x.y)),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge] \\
\triangleright\!\!\mapsto & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge])^{1\multimap 0}[\sim\blacklozenge\ ((\lambda x.\langle x\rangle)\ (\lambda x.y)),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge] \\
= & !\langle\lambda z.(\sim\blacklozenge\ ((\lambda x.\langle x\rangle)\ (\lambda x.y)),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
= & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge][\sim\Box])^{0\multimap 0}[\blacklozenge\ ((\lambda x.\langle x\rangle)\ (\lambda x.y)),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge] \\
\triangleright\!\!\mapsto & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge][\sim\Box])^{0\multimap 0}[\blacklozenge\ (\lambda x.\langle x\rangle),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge\blacklozenge\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge] \\
= & !\langle\lambda z.(\sim\blacklozenge\ (\lambda x.\langle x\rangle),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge\blacklozenge\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
= & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge][\sim\Box][\Box\blacklozenge\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge])^{0\multimap 0}[\blacklozenge\ (\lambda x.\langle x\rangle),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge] \\
\triangleright\!\!\mapsto & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge][\sim\Box][\Box\blacklozenge\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge])^{0\multimap 0}[\mathbb{d}\ (\lambda x.\langle x\rangle),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}] \\
= & !\langle\lambda z.(\sim(\mathbb{d}\ (\lambda x.\langle x\rangle),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\blacklozenge\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
= & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge][\sim\Box][\mathbb{d}\ (\lambda x.\langle x\rangle),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\Box])^{0\multimap 0}[\blacklozenge\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge] \\
\triangleright\!\!\mapsto & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge][\sim\Box][\mathbb{d}\ (\lambda x.\langle x\rangle),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\Box])^{0\multimap 0}[\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}] \\
= & !\langle\lambda z.(\sim(\mathbb{d}\ (\lambda x.\langle x\rangle),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\ \mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D})\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
= & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge][\sim\Box])^{0\multimap 0}[\mathbb{d}\ (\lambda x.\langle x\rangle),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\ \mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}] \\
\triangleright\!\!\mapsto & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge][\sim\Box])^{0\multimap 0}[\blacklozenge\ \langle x\rangle,\ (\{(x,\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}),(y,z),(z,z)\};\varepsilon)\ \blacklozenge] \\
= & !\langle\lambda z.(\sim(\blacklozenge\ \langle x\rangle,\ (\{(x,\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
= & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge][\sim\Box])^{0\multimap 0}[\blacklozenge\ \langle x\rangle,\ (\{(x,\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}),(y,z),(z,z)\};\varepsilon)\ \blacklozenge] \\
\triangleright\!\!\mapsto & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge][\sim\Box])^{0\multimap 0}[\langle\blacklozenge\ x,\ (\{(x,\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}),(y,z),(z,z)\};\varepsilon)\ \blacklozenge\rangle] \\
= & !\langle\lambda z.(\sim\langle\blacklozenge\ x,\ (\{(x,\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}),(y,z),(z,z)\};\varepsilon)\ \blacklozenge\rangle\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
= & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge][\sim\Box][\langle\Box\rangle])^{1\multimap 0}[\blacklozenge\ x,\ (\{(x,\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}),(y,z),(z,z)\};\varepsilon)\ \blacklozenge] \\
\triangleright\!\!\mapsto & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge][\sim\Box][\langle\Box\rangle])^{1\multimap 0}[\blacklozenge\ \mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D},\ (\varepsilon)\ \blacklozenge] \\
= & !\langle\lambda z.(\sim\langle\blacklozenge\ \mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D},\ (\varepsilon)\ \blacklozenge\rangle\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
= & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge][\sim\Box][\langle\Box\rangle])^{1\multimap 0}[\blacklozenge\ \mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D},\ (\varepsilon)\ \blacklozenge] \\
\triangleright\!\!\mapsto & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge][\sim\Box][\langle\Box\rangle])^{1\multimap 0}[\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}] \\
= & !\langle\lambda z.(\sim\langle\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\rangle\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
= & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge])^{1\multimap 0}[\sim\langle\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\rangle] \\
\triangleright\!\!\mapsto & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\Box\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge])^{1\multimap 0}[\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}] \\
= & !\langle\lambda z.(\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge)\rangle \\
= & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\Box])^{1\multimap 0}[\blacklozenge\ 0,\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \blacklozenge] \\
\triangleright\!\!\mapsto & (\Box[!\Box][\langle\Box\rangle][\lambda z.\Box][\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\Box])^{1\multimap 0}[0] \\
= & !\langle\lambda z.(\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\ 0)\rangle \\
= & \Box^{0\multimap 0}[!\langle\lambda z.(\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\ 0)\rangle] \\
\triangleright\!\!\mapsto & \Box^{0\multimap 0}[\blacklozenge\ \lambda z.(\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\ 0),\ (\{(x,x),(y,y),(z,z)\};\varepsilon)\ \blacklozenge] \\
= & \blacklozenge\ \lambda z.(\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\ 0),\ (\{(x,x),(y,y),(z,z)\};\varepsilon)\ \blacklozenge \\
= & \Box^{0\multimap 0}[\blacklozenge\ \lambda z.(\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\ 0),\ (\{(x,x),(y,y),(z,z)\};\varepsilon)\ \blacklozenge] \\
\triangleright\!\!\mapsto & \Box^{0\multimap 0}[\mathbb{d}\ \lambda z.(\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\ 0),\ (\{(x,x),(y,y),(z,z)\};\varepsilon)\ \mathbb{D}] \\
= & \mathbb{d}\ \lambda z.(\mathbb{d}\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \mathbb{D}\ 0),\ (\{(x,x),(y,y),(z,z)\};\varepsilon)\ \mathbb{D}
\end{array}
$$

Figure 5.1: Evaluation of $!\langle\lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle$ in Reduction Semantics of Environmental MetaML.

$$eval_{\text{MetaML:EnvRed}} : \text{PRGM}_{\text{MetaML}} \rightharpoonup \text{ANS}_{\text{MetaML}}$$

$$eval_{\text{MetaML:EnvRed}}(t) = \begin{cases} \texttt{function} & \text{if } \triangleright \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon) \blacklozenge \longmapsto^* \langle\!| \lambda x.t'^{0},\, \rho^* |\!\rangle \\ \texttt{code} & \text{if } \triangleright \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon) \blacklozenge \longmapsto^* \langle v^1 \rangle \\ n & \text{if } \triangleright \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon) \blacklozenge \longmapsto^* n \end{cases}$$

This evaluator is defined based on the reduction semantics of Environmental MetaML. The subscript "$_{\text{MetaML:EnvRed}}$" in $eval_{\text{MetaML:EnvRed}}$ denotes the reduction semantics of Environmental MetaML.

We claim that the evaluators defined in terms of the structural operational semantics and the reduction semantics of Environmental MetaML are equivalent.

Theorem 189 (Kleene Equality of Evaluators)**.** *For any $t \in \text{PRGM}_{\text{MetaML}}$, $eval_{\text{MetaML:EnvSOS}}(t)$ is Kleene equal to $eval_{\text{MetaML:EnvRed}}(t)$.*

We prove the theorem in appendices.

5.6 Environmental MetaML - Abstract Machine (MEK Machine)

Following the path of refining a reduction semantics to a corresponding abstract machine as presented in Sections 3.6 and 4.2, we refine Environmental MetaML's reduction semantics to an abstract machine. We call the abstract machine the MEK machine.

5.6.1 Syntax

The definitions of source terms, runtime terms, values, denotable terms, configurations and meta-environments are the same as Sections 5.4 and 5.5.

5.6.1.1 Evaluation Contexts

Analogous to Substitutional MetaML's Definition 121, we provide an alternative definition for evaluation contexts.

Definition 190 (Evaluation Contexts: Outside-in)**.** Let $i, j \in \mathbb{N}$. Define $\text{ECXT}^{i-\circ j}$ to be the set of evaluation contexts with inner level i and outer level j.

$$E^{i\multimap j} \in \textsc{Ecxt}^{i\multimap j},\ c^i \in \textsc{Conf}^i,\ v^i \in \textsc{Value}^i$$

$$\frac{}{\Box \in \textsc{Exct}^{j\multimap j}}\ \text{(ept-j)}$$

$$\frac{E \in \textsc{Exct}^{i\multimap j}}{(E\ c_2^j) \in \textsc{Exct}^{i\multimap j}}\ \text{(appL-j)} \qquad \frac{E \in \textsc{Exct}^{i\multimap j}}{(v_1^j\ E) \in \textsc{Exct}^{i\multimap j}}\ \text{(appR-j)}$$

$$\frac{E \in \textsc{Exct}^{i\multimap (j+1)}}{\lambda x.E \in \textsc{Exct}^{i\multimap (j+1)}}\ \text{(lambda-(j+1))}$$

$$\frac{E \in \textsc{Exct}^{i\multimap (j+1)}}{\langle E\rangle \in \textsc{Exct}^{i\multimap j}}\ \text{(code-j)} \qquad \frac{E \in \textsc{Exct}^{i\multimap j}}{{\sim}E \in \textsc{Exct}^{i\multimap (j+1)}}\ \text{(splice-(j+1))} \qquad \frac{E \in \textsc{Exct}^{i\multimap j}}{!E \in \textsc{Exct}^{i\multimap j}}\ \text{(run-j)}$$

$$\frac{E \in \textsc{Exct}^{i\multimap j}}{(E + c_2^j) \in \textsc{Exct}^{i\multimap j}}\ \text{(plusL-j)} \qquad \frac{E \in \textsc{Exct}^{i\multimap j}}{(v_1^j + E) \in \textsc{Exct}^{i\multimap j}}\ \text{(plusR-j)}$$

5.6.1.2 Machine Configurations

Section 4.2 defines the states of the MK machine through four modes of machine configurations. We revise the four-mode definition to accommodate an environmental semantics.

Definition 191 (Machine Configurations)**.** Define CFG to be the set of machine configurations.

$$i, j \in \mathbb{N},\ C \in \textsc{Cfg},\ c^i \in \textsc{Conf}^i,\ v^i \in \textsc{Value}^i,\ E^{i\multimap j} \in \textsc{Ectx}^{i\multimap j}$$

$$\begin{aligned} C \ ::=\ & v^0 \\ & \mid \langle i,\ E^{i\multimap 0},\ c^i\rangle_{\mathrm{r}} \\ & \mid \langle i,\ E^{i\multimap 0},\ c^i\rangle_{\mathrm{f}} \\ & \mid \langle i,\ E^{i\multimap 0},\ v^i\rangle_{\mathrm{b}} \end{aligned}$$

The machine operates in four modes: the *value* mode v^0, the *reduce* mode $\langle i,\ E^{i\multimap 0},\ c^i\rangle_{\mathrm{r}}$, the *focus* mode $\langle i,\ E^{i\multimap 0},\ c^i\rangle_{\mathrm{f}}$, the *build* mode $\langle i,\ E^{i\multimap 0},\ v^i\rangle_{\mathrm{b}}$.

A machine configuration $\langle i,\ E^{i\multimap 0},\ c^i\rangle_?$ where $? \in \{\mathrm{r},\mathrm{f},\mathrm{b}\}$ unloads to the configuration $E^{i\multimap 0}[c^i]$. Precisely, the configuration c^i in a machine configuration at reduce mode $\langle i,\ E^{i\multimap 0},\ c^i\rangle_{\mathrm{r}}$ needs to be a redex.

5.6.2 Abstract Machine (MEK Machine)

We lay out the abstract machine of Environmental MetaML, i.e., the MEK machine, through the reduction relation $\longmapsto_{\mathrm{mek}}$ and the multi-reduction relation $\longmapsto^*_{\mathrm{mek}}$.

Definition 192 (Reduction Relation)**.** Let the reduction relation $\longmapsto_{\mathrm{mek}}$ be a binary relation between the set of machine configurations and the set of machine configurations.

$$\longmapsto_{\mathrm{mek}} \subseteq \textsc{Cfg} \times \textsc{Cfg}$$

Reduce rules: $\langle i,\, E^{i-\circ 0},\, c^i \rangle_{\mathrm{r}}$

$$\begin{array}{rcll}
\langle 0,\, E,\, \triangleleft\ \lambda x.t^0,\, (\rho;\rho^*)\ \triangleright\ v^0 \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle 0,\, E,\, \blacktriangleleft\ t^0,\, (\rho[x \mapsto v^0];\rho^*)\ \blacktriangleright \rangle_{\mathrm{f}} & \text{(r-app-0)} \\
\langle 0,\, E,\, !\langle v^1 \rangle \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle 0,\, E,\, \blacktriangleleft\ v^1,\, (\rho_{\mathrm{init}}^{\mathrm{VAR}\langle E[!\langle v^1 \rangle]\rangle};\varepsilon)\ \blacktriangleright \rangle_{\mathrm{f}} & \text{(r-run-0)} \\
\langle 1,\, E,\, \sim\langle v^1 \rangle \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle 1,\, E,\, v^1 \rangle_{\mathrm{f}} & \text{(r-splice-1)} \\
\langle 0,\, E,\, n_1 + n_2 \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle 0,\, E,\, n \rangle_{\mathrm{f}} \text{ where } n = n_1 + n_2 & \text{(r-plus-0)} \\
\langle 0,\, E,\, \blacktriangleleft\ \lambda x.t^0,\, \rho^*\ \blacktriangleright \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle 0,\, E,\, \triangleleft\ \lambda x.t^0,\, \rho^*\ \triangleright \rangle_{\mathrm{f}} & \text{(r-conf-lam-0)} \\
\langle i+1,\, E,\, \blacktriangleleft\ \lambda x.t^{i+1},\, (\rho;\rho^*)\ \blacktriangleright \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle i+1,\, E,\, \lambda x_N.\blacktriangleleft\ t^{i+1},\, (\rho[x \mapsto x_N];\rho[x_N \mapsto x_N]^*)\ \blacktriangleright \rangle_{\mathrm{f}} & \\
 & & \qquad \text{where } x_N \notin \mathrm{VAR}(E[\blacktriangleleft\ \lambda x.t^{i+1},\, (\rho;\rho^*)\ \blacktriangleright]) & \text{(r-conf-lam-(i+1))} \\
\langle i,\, E,\, \blacktriangleleft\ \triangleleft\ \lambda x.t,\, \rho_1^*\ \triangleright,\, \rho_2^*\ \blacktriangleright \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E,\, \triangleleft\ \lambda x.t,\, (\rho_1^*;\rho_2^*)\ \triangleright \rangle_{\mathrm{f}} & \text{(r-conf-clov-i)} \\
\langle i,\, E,\, \blacktriangleleft\ w,\, \varepsilon\ \blacktriangleright \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E,\, w \rangle_{\mathrm{f}} & \text{(r-conf-den-i)} \\
\langle i,\, E,\, \blacktriangleleft\ x,\, (\rho;\rho^*)\ \blacktriangleright \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E,\, \blacktriangleleft\ \rho(x),\, \rho^*\ \blacktriangleright \rangle_{\mathrm{f}} & \text{(r-conf-var-i)} \\
\langle i,\, E,\, \blacktriangleleft\ n,\, \rho^*\ \blacktriangleright \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E,\, n \rangle_{\mathrm{f}} & \text{(r-conf-num-i)} \\
\langle i,\, E,\, \blacktriangleleft\ t_1\ t_2,\, \rho^*\ \blacktriangleright \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E,\, \blacktriangleleft\ t_1,\, \rho^*\ \blacktriangleright\ \blacktriangleleft\ t_2,\, \rho^*\ \blacktriangleright \rangle_{\mathrm{f}} & \text{(r-conf-app-i)} \\
\langle i,\, E,\, \blacktriangleleft\ \langle t^{i+1} \rangle,\, \rho^*\ \blacktriangleright \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E,\, \langle \blacktriangleleft\ t^{i+1},\, \rho^*\ \blacktriangleright \rangle \rangle_{\mathrm{f}} & \text{(r-conf-code-i)} \\
\langle i,\, E,\, \blacktriangleleft\ !t^i,\, \rho^*\ \blacktriangleright \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E,\, !\blacktriangleleft\ t^i,\, \rho^*\ \blacktriangleright \rangle_{\mathrm{f}} & \text{(r-conf-run-i)} \\
\langle i,\, E,\, \blacktriangleleft\ \sim t^i,\, \rho^*\ \blacktriangleright \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E,\, \sim\blacktriangleleft\ t^i,\, \rho^*\ \blacktriangleright \rangle_{\mathrm{f}} & \text{(r-conf-splice-(i+1))} \\
\langle i,\, E,\, \blacktriangleleft\ t_1 + t_2,\, \rho^*\ \blacktriangleright \rangle_{\mathrm{r}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E,\, \blacktriangleleft\ t_1,\, \rho^*\ \blacktriangleright + \blacktriangleleft\ t_2,\, \rho^*\ \blacktriangleright \rangle_{\mathrm{f}} & \text{(r-conf-plus-i)}
\end{array}$$

Focus rules: $\langle i,\, E^{i-\circ 0},\, c^i \rangle_{\mathrm{f}}$

$$\begin{array}{rcll}
\langle i,\, E,\, \blacktriangleleft\ t,\, \rho^*\ \blacktriangleright \rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E,\, \blacktriangleleft\ t,\, \rho^*\ \blacktriangleright \rangle_{\mathrm{r}} & \text{(f-conf-i)} \\
\langle i+1,\, E,\, x \rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mek}} & \langle i+1,\, E,\, x \rangle_{\mathrm{b}} & \text{(f-var-(i+1))} \\
\langle i,\, E,\, c_1\ c_2 \rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E[\Box\ c_2],\, c_1 \rangle_{\mathrm{f}} & \text{(f-appL-i)} \\
\langle 0,\, E,\, \triangleleft\ \lambda x.t,\, \rho^*\ \triangleright \rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mek}} & \langle 0,\, E,\, \triangleleft\ \lambda x.t,\, \rho^*\ \triangleright \rangle_{\mathrm{b}} & \text{(f-lambda-0)} \\
\langle i+1,\, E,\, \lambda x.c \rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mek}} & \langle i+1,\, E[\lambda x.\Box],\, c \rangle_{\mathrm{f}} & \text{(f-lambda-(i+1))} \\
\langle i,\, E,\, \langle c \rangle \rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mek}} & \langle i+1,\, E[\langle \Box \rangle],\, c \rangle_{\mathrm{f}} & \text{(f-code-i)} \\
\langle i+1,\, E,\, \sim c \rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E[\sim\Box],\, c \rangle_{\mathrm{f}} & \text{(f-splice-(i+1))} \\
\langle i,\, E,\, !c \rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E[!\Box],\, c \rangle_{\mathrm{f}} & \text{(f-run-i)} \\
\langle i,\, E,\, n \rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E,\, n \rangle_{\mathrm{b}} & \text{(f-num-i)} \\
\langle i,\, E,\, c_1 + c_2 \rangle_{\mathrm{f}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E[\Box + c_2],\, c_1 \rangle_{\mathrm{f}} & \text{(f-plusL-i)}
\end{array}$$

Build rules: $\langle i,\, E^{i-\circ 0},\, v^i \rangle_{\mathrm{b}}$

$$\begin{array}{rcll}
\langle 0,\, \Box,\, v \rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mek}} & v & \text{(b-value-0)} \\
\langle i,\, E[\Box\ c_2],\, v_1 \rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E[v_1\ \Box],\, c_2 \rangle_{\mathrm{f}} & \text{(b-appL-i)} \\
\langle 0,\, E[v_1\ \Box],\, v_2 \rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mek}} & \langle 0,\, E,\, v_1\ v_2 \rangle_{\mathrm{r}} & \text{(b-appR-0)} \\
\langle i+1,\, E[v_1\ \Box],\, v_2 \rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mek}} & \langle i+1,\, E,\, v_1\ v_2 \rangle_{\mathrm{b}} & \text{(b-appR-(i+1))} \\
\langle i+1,\, E[\lambda x.\Box],\, v \rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mek}} & \langle i+1,\, E,\, \lambda x.v \rangle_{\mathrm{b}} & \text{(b-lambda-(i+1))} \\
\langle i+1,\, E[\langle \Box \rangle],\, v \rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E,\, \langle v \rangle \rangle_{\mathrm{b}} & \text{(b-code-(i+1))} \\
\langle 0,\, E[\sim\Box],\, v \rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mek}} & \langle 1,\, E,\, \sim v \rangle_{\mathrm{r}} & \text{(b-splice-0)} \\
\langle i+1,\, E[\sim\Box],\, v \rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mek}} & \langle i+2,\, E,\, \sim v \rangle_{\mathrm{b}} & \text{(b-splice-(i+1))} \\
\langle 0,\, E[!\Box],\, v \rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mek}} & \langle 0,\, E,\, !v \rangle_{\mathrm{r}} & \text{(b-run-0)} \\
\langle i+1,\, E[!\Box],\, v \rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mek}} & \langle i+1,\, E,\, !v \rangle_{\mathrm{b}} & \text{(b-run-(i+1))} \\
\langle i,\, E[\Box + c_2],\, v_1 \rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mek}} & \langle i,\, E[v_1 + \Box],\, c_2 \rangle_{\mathrm{f}} & \text{(b-plusL-i)} \\
\langle 0,\, E[v_1 + \Box],\, v_2 \rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mek}} & \langle 0,\, E,\, v_1 + v_2 \rangle_{\mathrm{r}} & \text{(b-plusR-0)} \\
\langle i+1,\, E[v_1 + \Box],\, v_2 \rangle_{\mathrm{b}} & \longmapsto_{\mathrm{mek}} & \langle i+1,\, E,\, v_1 + v_2 \rangle_{\mathrm{b}} & \text{(b-plusR-(i+1))}
\end{array}$$

The reduction relation $C_1 \longmapsto_{\text{mek}} C_2$ reads as "C_1 reduces to C_2" or "C_1 single-reduces to C_2".

The intuition behind the above relation is analogous to that of CEK machine's reduction relation. See comments below Definition 108.

Definition 193 (Multi-reduction Relation)**.** Let the multi-reduction relation $\longmapsto^*_{\text{mek}}$ be the reflexive-transitive closure of the reduction relation $\longmapsto_{\text{mek}}$.

The abstract machine defined above is also known as the MEK machine. M stands for multi-stage, E stands for environment, and K stands for continuation, i.e., the evaluation context.

Example 194. Consider $!\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle$.

We first construct a closure that pairs the above term with an initial meta-environment:

$$\blacklozenge\ !\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle,\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge$$

Then we construct the initial machine configuration at focus mode:

$$\langle 0,\ \square,\ \blacklozenge\ !\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle,\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge\rangle_{\text{f}}$$

By the MEK machine, we have:

$$\begin{array}{ll} & \langle 0,\ \square,\ \blacklozenge\ !\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle,\ (\{(x,x),(y,y)\};\varepsilon)\ \blacklozenge\rangle_{\text{f}} \\ \longmapsto^*_{\text{mek}} & \triangleleft\ \lambda z.(\triangleleft\ (\lambda x.y),\ (\{(x,x),(y,z),(z,z)\};\varepsilon)\ \triangleright\ 0),\ (\{(x,x),(y,y),(z,z)\};\varepsilon)\ \triangleright \end{array}$$

as demonstrated in Figure 5.2.

Evaluator. We now define an evaluator in terms of the MEK machine. The MEK machine's multi-reduction relation is defined on machine configurations. Given a program t, the evaluator applies the multi-reduction relation on the machine configuration $\langle 0,\ \square,\ \blacklozenge\ t,\ (\rho_{\text{init}}^{\text{VAR}(t)};\varepsilon)\ \blacklozenge\rangle_{\text{f}}$ in which the program is associated with an initial meta-environment and an empty evaluation context. The evaluator is otherwise analogous to the one defined in terms of the reduction semantics of Environmental MetaML.

Definition 195 (Evaluator based on MEK Machine)**.** Let the evaluator $eval_{\text{MetaML:MEK}}$ be a partial function from the set of programs $\text{PRGM}_{\text{MetaML}}$ to the set of answers $\text{ANS}_{\text{MetaML}}$.

$$eval_{\text{MetaML:MEK}} : \text{PRGM}_{\text{MetaML}} \rightharpoonup \text{ANS}_{\text{MetaML}}$$

$$eval_{\text{MetaML:MEK}}(t) = \begin{cases} \texttt{function} & \text{if } \langle 0,\ \square,\ \blacklozenge\ t,\ (\rho_{\text{init}}^{\text{VAR}(t)};\varepsilon)\ \blacklozenge\rangle_{\text{f}} \longmapsto^*_{\text{mek}} \triangleleft\ \lambda x.t'^{0},\ \rho^*\ \triangleright \\ \texttt{code} & \text{if } \langle 0,\ \square,\ \blacklozenge\ t,\ (\rho_{\text{init}}^{\text{VAR}(t)};\varepsilon)\ \blacklozenge\rangle_{\text{f}} \longmapsto^*_{\text{mek}} \langle v^1\rangle \\ n & \text{if } \langle 0,\ \square,\ \blacklozenge\ t,\ (\rho_{\text{init}}^{\text{VAR}(t)};\varepsilon)\ \blacklozenge\rangle_{\text{f}} \longmapsto^*_{\text{mek}} n \end{cases}$$

This evaluator is defined based on the MEK machine. The subscript "$_{\text{MetaML:MEK}}$" in $eval_{\text{MetaML:MEK}}$ denotes the MEK machine of Environmental MetaML.

	⟨0,	□,	◖ !⟨λy.(∼((λx.⟨x⟩) (λx.y)) 0)⟩, ({(x,x),(y,y)};ε) ◗	⟩_f
⟼_mek	⟨0,	□,	◖ !⟨λy.(∼((λx.⟨x⟩) (λx.y)) 0)⟩, ({(x,x),(y,y)};ε) ◗	⟩_r
⟼_mek	⟨0,	□,	!◖ ⟨λy.(∼((λx.⟨x⟩) (λx.y)) 0)⟩, ({(x,x),(y,y)};ε) ◗	⟩_f
⟼_mek	⟨0,	□[!□],	◖ ⟨λy.(∼((λx.⟨x⟩) (λx.y)) 0)⟩, ({(x,x),(y,y)};ε) ◗	⟩_f
⟼_mek	⟨0,	□[!□],	◖ ⟨λy.(∼((λx.⟨x⟩) (λx.y)) 0)⟩, ({(x,x),(y,y)};ε) ◗	⟩_r
⟼_mek	⟨0,	□[!□],	⟨◖ λy.(∼((λx.⟨x⟩) (λx.y)) 0), ({(x,x),(y,y)};ε) ◗⟩	⟩_f
⟼_mek	⟨1,	□[!□][⟨□⟩],	◖ λy.(∼((λx.⟨x⟩) (λx.y)) 0), ({(x,x),(y,y)};ε) ◗	⟩_f
⟼_mek	⟨1,	□[!□][⟨□⟩],	◖ λy.(∼((λx.⟨x⟩) (λx.y)) 0), ({(x,x),(y,y)};ε) ◗	⟩_r
⟼_mek	⟨1,	□[!□][⟨□⟩],	λz.◖ ∼((λx.⟨x⟩) (λx.y)) 0, ({(x,x),(y,z),(z,z)};ε) ◗	⟩_f
			where z ∉ {x,y}	
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□],	◖ ∼((λx.⟨x⟩) (λx.y)) 0, ({(x,x),(y,z),(z,z)};ε) ◗	⟩_f
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□],	◖ ∼((λx.⟨x⟩) (λx.y)) 0, ({(x,x),(y,z),(z,z)};ε) ◗	⟩_r
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□],	◖ ∼((λx.⟨x⟩) (λx.y)), ({(x,x),(y,z),(z,z)};ε) ◗ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗	⟩_f
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗],	◖ ∼((λx.⟨x⟩) (λx.y)), ({(x,x),(y,z),(z,z)};ε) ◗	⟩_f
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗],	◖ ∼((λx.⟨x⟩) (λx.y)), ({(x,x),(y,z),(z,z)};ε) ◗	⟩_r
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗],	∼◖ (λx.⟨x⟩) (λx.y), ({(x,x),(y,z),(z,z)};ε) ◗	⟩_f
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□],	◖ (λx.⟨x⟩) (λx.y), ({(x,x),(y,z),(z,z)};ε) ◗	⟩_f
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□],	◖ (λx.⟨x⟩) (λx.y), ({(x,x),(y,z),(z,z)};ε) ◗	⟩_r
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□],	◖ (λx.⟨x⟩), ({(x,x),(y,z),(z,z)};ε) ◗ ◖ (λx.y), ({(x,x),(y,z),(z,z)};ε) ◗	⟩_f
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□][□ ◖ (λx.y), ({(x,x),(y,z),(z,z)};ε) ◗],	◖ (λx.⟨x⟩), ({(x,x),(y,z),(z,z)};ε) ◗	⟩_f
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□][□ ◖ (λx.y), ({(x,x),(y,z),(z,z)};ε) ◗],	◖ (λx.⟨x⟩), ({(x,x),(y,z),(z,z)};ε) ◗	⟩_r
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□][□ ◖ (λx.y), ({(x,x),(y,z),(z,z)};ε) ◗],	◁ (λx.⟨x⟩), ({(x,x),(y,z),(z,z)};ε) ▷	⟩_f
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□][□ ◖ (λx.y), ({(x,x),(y,z),(z,z)};ε) ◗],	◁ (λx.⟨x⟩), ({(x,x),(y,z),(z,z)};ε) ▷	⟩_b
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□][◁ (λx.⟨x⟩), ({(x,x),(y,z),(z,z)};ε) ▷ □],	◖ (λx.y), ({(x,x),(y,z),(z,z)};ε) ◗	⟩_f
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□][◁ (λx.⟨x⟩), ({(x,x),(y,z),(z,z)};ε) ▷ □],	◖ (λx.y), ({(x,x),(y,z),(z,z)};ε) ◗	⟩_r
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□][◁ (λx.⟨x⟩), ({(x,x),(y,z),(z,z)};ε) ▷ □],	◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷	⟩_f
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□][◁ (λx.⟨x⟩), ({(x,x),(y,z),(z,z)};ε) ▷ □],	◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷	⟩_b
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□],	◁ (λx.⟨x⟩), ({(x,x),(y,z),(z,z)};ε) ▷ ◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷	⟩_r
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□],	◖ ⟨x⟩, ({(x,◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷),(y,z),(z,z)};ε) ◗	⟩_f
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□],	◖ ⟨x⟩, ({(x,◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷),(y,z),(z,z)};ε) ◗	⟩_r
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□],	⟨◖ x, ({(x,◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷),(y,z),(z,z)};ε) ◗⟩	⟩_f
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□][⟨□⟩],	◖ x, ({(x,◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷),(y,z),(z,z)};ε) ◗	⟩_f
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□][⟨□⟩],	◖ x, ({(x,◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷),(y,z),(z,z)};ε) ◗	⟩_r
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□][⟨□⟩],	◖ ◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷, (ε) ◗	⟩_f
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□][⟨□⟩],	◖ ◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷, (ε) ◗	⟩_r
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□][⟨□⟩],	◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷	⟩_f
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□][⟨□⟩],	◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷	⟩_b
⟼_mek	⟨0,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗][∼□],	⟨◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷⟩	⟩_b
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗],	∼⟨◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷⟩	⟩_r
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗],	◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷	⟩_f
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][□ ◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗],	◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷	⟩_b
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷ □],	◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗	⟩_f
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷ □],	◖ 0, ({(x,x),(y,z),(z,z)};ε) ◗	⟩_r
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷ □],	0	⟩_f
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□][◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷ □],	0	⟩_b
⟼_mek	⟨1,	□[!□][⟨□⟩][λz.□],	◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷ 0	⟩_b
⟼_mek	⟨1,	□[!□][⟨□⟩],	λz.(◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷ 0)	⟩_b
⟼_mek	⟨0,	□[!□],	⟨λz.(◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷ 0)⟩	⟩_b
⟼_mek	⟨0,	□,	!⟨λz.(◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷ 0)⟩	⟩_r
⟼_mek	⟨0,	□,	◖ λz.(◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷ 0), ({(x,x),(y,y),(z,z)};ε) ◗	⟩_f
⟼_mek	⟨0,	□,	◖ λz.(◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷ 0), ({(x,x),(y,y),(z,z)};ε) ◗	⟩_r
⟼_mek	⟨0,	□,	◁ λz.(◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷ 0), ({(x,x),(y,y),(z,z)};ε) ▷	⟩_f
⟼_mek	⟨0,	□,	◁ λz.(◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷ 0), ({(x,x),(y,y),(z,z)};ε) ▷	⟩_b
⟼_mek			◁ λz.(◁ (λx.y), ({(x,x),(y,z),(z,z)};ε) ▷ 0), ({(x,x),(y,y),(z,z)};ε) ▷	

Figure 5.2: Evaluation of $!\langle \lambda y.(\sim((\lambda x.\langle x\rangle)\ (\lambda x.y))\ 0)\rangle$ in the MEK Machine.

We claim that the evaluators defined in terms of Environmental MetaML's reduction semantics and the MEK machine are equivalent.

Theorem 196 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{MetaML}}$*,* $eval_{\text{MetaML:EnvRed}}(t)$ *is Kleene equal to* $eval_{\text{MetaML:MEK}}(t)$*.*

We prove the above theorem in appendices.

As a corollary, the evaluators defined in terms of Substitutional MetaML and the MEK machine are equivalent.

Corollary 197 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{MetaML}}$*,* $eval_{\text{MetaML:SubSOS}}(t)$ *is Kleene equal to* $eval_{\text{MetaML:MEK}}(t)$*.*

Proof. It immediately follows from Theorems 179, 189 and 196 by the transitivity of Kleene equality. □

5.7 Chapter Summary

Utilising the experience of refining semantics along two dimensions, this chapter eventually solved the following semantics refinement problem.

> Can we refine the substitutional structural operational semantics of MetaML to a corresponding environmental abstract machine, which we call the MEK machine, and demonstrate their equivalence?

We accomplished the development progressively in several manageable steps, each of which led to an intermediate semantics. We first reviewed the substitutional structural operational semantics of MetaML that we developed in Section 2.2. Then we successively developed the structural operational semantics of Explicit MetaML, the structural operational semantics of Suspended MetaML, the structural operational semantics of Environmental MetaML, the reduction semantics of Environmental MetaML, and finally derived the abstract machine of Environmental MetaML. We call the abstract machine of Environmental MetaML the MEK machine.

We defined an evaluator based on each semantics. By proving the equivalence of every two adjacent semantics, we finally showed that the MEK machine is equivalent to the substitutional structural operational semantics of MetaML.

By this chapter together with Chapter 2, we have successfully solved the main semantics refinement problem.

Chapter 6

Proof Methodology and Related Work

We first summarise three proof techniques that were adopted throughout the thesis the prove semantics equivalences. Then we compare our thesis with the related work.

6.1 Proof Methodology

Most proofs of the thesis can be categorised as proving (1) the equivalence of two structural operational semantics, (2) the equivalence of a structural operational semantics and a reduction semantics, or (3) the equivalence of a reduction semantics and an abstract machine.

Proving Equivalence of Two Structural Operational Semantics. To prove the equivalence of structural operational semantics of language A (defined by the single-step relation $\longrightarrow_{\mathrm{A}}$ and the multi-step relation $\longrightarrow_{\mathrm{A}}^{*}$) and structural operational semantics of language B (defined by the single-step relation $\longrightarrow_{\mathrm{B}}$ and the multi-step relation $\longrightarrow_{\mathrm{B}}^{*}$), we first define a bisimulation relation between their terms, i.e., $\simeq \subseteq \textsc{Term}_{\mathrm{A}} \times \textsc{Term}_{\mathrm{B}}$. Then the key is to demonstrate that the bisimulation relation respects the following properties.

1. $\forall p \in \textsc{Prgm},\ \mathrm{inj}_{\mathrm{A}}(p) \simeq \mathrm{inj}_{\mathrm{B}}(p)$.

   That is, for any program, its injected initial terms in languages A and B shall be related.

2. If $v_{\mathrm{A}} \simeq v_{\mathrm{B}}$, then $\mathrm{obs}_{\mathrm{A}}(v_{\mathrm{A}}) = \mathrm{obs}_{\mathrm{B}}(v_{\mathrm{B}})$.

   That is, two related values shall have the same observable results.

3. Canonisation:

   (a) If $v_{\mathrm{A}} \simeq t_{\mathrm{B}}$, then $t_{\mathrm{B}} \longrightarrow_{\mathrm{B}}^{*} v_{\mathrm{B}}$ and $v_{\mathrm{A}} \simeq v_{\mathrm{B}}$.

   (b) If $t_{\mathrm{A}} \simeq v_{\mathrm{B}}$, then $t_{\mathrm{A}} \longrightarrow_{\mathrm{A}}^{*} v_{\mathrm{A}}$ and $v_{\mathrm{A}} \simeq v_{\mathrm{B}}$.

4. Weak Bisimulation:

   (a) If $t_{\mathrm{A}_1} \simeq t_{\mathrm{B}_1}$ and $t_{\mathrm{A}_1} \longrightarrow_{\mathrm{A}} t_{\mathrm{A}_2}$, then $t_{\mathrm{B}_1} \longrightarrow_{\mathrm{B}}^{*} t_{\mathrm{B}_2}$ and $t_{\mathrm{A}_2} \simeq t_{\mathrm{B}_2}$.

   (b) If $t_{\mathrm{A}_1} \simeq t_{\mathrm{B}_1}$ and $t_{\mathrm{B}_1} \longrightarrow_{\mathrm{B}} t_{\mathrm{B}_2}$, then $t_{\mathrm{A}_1} \longrightarrow_{\mathrm{A}}^{*} t_{\mathrm{A}_2}$ and $t_{\mathrm{A}_2} \simeq t_{\mathrm{B}_2}$.

The above-mentioned framework of proving the equivalence of structural operational semantics was motivated by the bisimulation proof method [San11, PS12].

Proving Equivalence of a Structural Operational Semantics and a Reduction Semantics. To prove the equivalence of a structural operational semantics (defined by the single-step relation $\longrightarrow$ and the multi-step relation $\longrightarrow^*$) and a reduction semantics (defined by the reduction relation $\longmapsto$ and the multi-reduction relation $\longmapsto^*$) of the same language, the key is to prove the following two lemmas.

1. If $t_1 \longrightarrow t_2$, then $t_1 \longmapsto t_2$.

   We may need to prove: If $t_1 \longrightarrow t_2$ and $E \in \text{ECXT}$, then $E[t_1] \longmapsto E[t_2]$.

2. If $t_1 \longmapsto t_2$, then $t_1 \longrightarrow t_2$.

   We may need to prove: If $t_1 \longrightarrow t_2$ and $E \in \text{ECXT}$, then $E[t_1] \longrightarrow E[t_2]$.

Proving Equivalence of a Reduction Semantics and an Abstract Machine. To prove the equivalence of a reduction semantics (defined by the single-reduction relation $\longmapsto$ and the multi-reduction relation $\longmapsto^*$) and an abstract machine (defined by the reduction relation $\longmapsto_{\text{abs}}$ and the multi-reduction relation $\longmapsto^*_{\text{abs}}$) of the same language, we first define a translator $\mathscr{T}$ to translate any machine configuration to its corresponding term. Then the key is to prove the following two lemmas.

1. If $E_0[t_0] = E_1[t_1]$ and $E_1[t_1] \longmapsto E_1[t_2]$ where $t_1 \mathscr{R} t_2$, then $\langle E_0, t_0\rangle_{\text{f}} \longmapsto^*_{\text{abs}} \langle E_0, t_0\rangle_{\text{f}}$.

   We may need to prove: If $t = E_1[t_1]$ and $t_1 \mathscr{R} t_2$, then $\langle E, t\rangle_{\text{f}} \longmapsto^*_{\text{abs}} \langle EE_1, t_1\rangle_{\text{f}}$.

2. If $C_1 \longmapsto_{\text{abs}} C_2$, then $\mathscr{T}(C_1) \longmapsto^* \mathscr{T}(C_2)$.

The above-mentioned framework of proving the equivalence of a reduction semantics and an abstract machine was motivated by the proof of the equivalence of the CC machine and the substitutional reduction semantics of ISWIM in [FFF09].

The above-mentioned three proof frameworks are adaptable to more complex languages. For example, the language of our interest may define its single-step relation (or the single-reduction relation) on configurations rather than on terms. As a result, we need to replace all t's by c's in the frameworks. As another example, in a multi-stage language, a machine configuration may have a level component. As a result, we need to add one more component of levels to any machine configuration in the last framework.

6.2 Related Work

Multi-stage Programming Languages. Several multi-stage programming languages and language extensions have been developed. For example, there are MetaML [TS97, She98, Tah99a, Tah99b] extends ML, MetaOCaml [Tah04] extends OCaml, MetaHaskell [Mai12] extends Haskell, Mint [WRI+09] extends Java and Metaphor [NR04] that extends C#. We intensively studied MetaML in the thesis.

Operational Semantics of MetaML. Taha [Tah99a] modelled a minimal subset of MetaML through two formulations that extend the lambda calculus, i.e., the λ-M language and the λ-U language. They presented a call-by-value substitutional natural semantics and a call-by-name substitutional natural semantics for the

λ-M language. They developed a call-by-name substitutional reduction semantics for the λ-U language and demonstrated its equivalence with respect to the call-by-name substitutional natural semantics of the λ-M language.

Our thesis took the call-by-value substitutional natural semantics of λ-M language defined in [Tah99a] as the reference semantics of MetaML. The substitutional structural operational semantics of MetaML developed in Chapter 2 can be deemed as a call-by-value substitutional structural operational semantics for the λ-M language.

Refining Semantics for ISWIM. Felleisen et al. [FFF09] presented how to develop the CEK machine from the substitutional reduction semantics of ISWIM and demonstrated their equivalence. Given the substitutional reduction semantics of ISWIM, they first derived a substitutional abstract machine called the CC machine in which a machine state is composed by a control string and an evaluation context. They then simplified the CC machine to the SCC machine to eliminate unnecessary state transition rules and side-conditions. Next they introduced a date structure called a continuation to make the evaluation context around the current control string the most evident and they refined the SCC machine to the CK machine in which a machine state is composed by a control string and a continuation. Finally they refined the CK machine to an environmental abstract machine, the CEK machine, by introducing environments to represent substitutions. Each CEK machine state has three components: a control string, an environment and a continuation. Furthermore, they built an evaluator for each above-mentioned semantics and they demonstrated the equivalence of the evaluators.

Chapter 3 of our thesis developed the CEK machine from the substitutional structural operational semantics of ISWIM. Instead of deriving a series of abstract machines, we developed a series of structural operational semantics. We introduced explicit substitutions in the structural operational semantics of Explicit ISWIM, suspended explicit substitutions in the structural operational semantics of Suspended ISWIM and environments in the structural operational semantics of Environmental ISWIM. We developed a reduction semantics for Environmental ISWIM, based on which the CEK machine was formulated.

Our thesis represents continuations by evaluation contexts. [FFF09] maintained a unique data structure to represent continuations but we did not.

Explicit Substitutions. The idea of explicit substitutions was introduced in [Cur85, ACCL91]. The development of Explicit ISWIM was partially inspired and motivated by the definitions of λx-terms and the definitions of λxgc-reduction in the λxgc-calculus [Ros96].

Proof Methodology. As mentioned in the previous section, the framework of proving the equivalence of structural operational semantics was motivated by the bisimulation proof method [San11, PS12]. The framework of proving the equivalence of a reduction semantics and an abstract machine was motivated by the proof of the equivalence of the CC machine and the substitutional reduction semantics of ISWIM in [FFF09].

Chapter 7

Conclusion

We conclude this thesis, summarise the limitations and list several directions for future work.

7.1 Conclusion

This thesis studied the problem of refining operational semantics for MetaML. We took the pre-existing substitutional natural semantics presented in [Tah99a] as the reference semantics of MetaML. The main research problem of our thesis, which was called the main semantics refinement problem, was stated as:

> Can we refine the pre-existing substitutional natural semantics of MetaML to a corresponding environmental abstract machine and demonstrate their equivalence?

As an environmental abstract machine is a small-step operational semantics, its development is more natural and convenient to start from a structural operational semantics than a natural semantics. In Chapter 2, we developed a substitutional structural operational semantics for MetaML and demonstrated its equivalence with respect to the substitutional natural semantics.

We then simplified the main semantics refinement problem along two dimensions—each dimension leads to a less complicated semantics refinement problem.

Following the first dimension, Chapter 3 studied how to develop an environmental abstract machine for a single-stage language, ISWIM, rather than the multi-stage language MetaML. We refined the substitutional structural operational semantics of ISWIM to its corresponding environmental abstract machine known as the CEK machine.

Following the second dimension, Chapter 4 studied how to develop a substitutional abstract machine rather than an environmental abstract machine for the multi-stage language MetaML. We refined the substitutional structural operational semantics of MetaML to its corresponding substitutional abstract machine, which we called the MK machine.

Utilising the experience of refining semantics along two dimensions, Chapter 5 finally studied the main semantics refinement problem, i.e., how to develop an environmental abstract machine for MetaML. We refined the substitutional structural operational semantics of MetaML to its corresponding environmental abstract machine, which we called the MEK machine.

Furthermore, three proof techniques were adopted throughout the thesis to prove the equivalence of two structural operational semantics, the equivalence of a structural operational semantics and a reduction semantics, and the equivalence of a reduction semantics and an abstract machine.

7.2 Limitations and Future Work

We briefly summarise the limitations of our thesis and point out several research ideas to be further explored in future work.

Simplifying Abstract Machines. We developed the CEK machine in Chapter 3, the MK machine in Chapter 4 and the MEK machine in Chapter 5. These machines have several redundant transformations of machine configurations. Felleisen et al. [FFF09] simplified abstract machines by (1) letting the machine exploit information from both the control strings and the evaluation contexts, and (2) combining definite transformations. Adopting the same approach, we can simplify our CEK machine, CK machine and MEK machine analogously. For example, for the MEK machine, the (b-appR-0) rule

$$\langle 0,\, E[v_1\ \square],\, v_2\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mek}} \langle 0,\, E,\, v_1\ v_2\rangle_{\mathrm{r}}$$

and the (r-app-0) rule

$$\langle 0,\, E,\, \triangleleft\ \lambda x.t^0,\, (\rho;\rho^*)\ \triangleright\ v^0\rangle_{\mathrm{r}} \longmapsto_{\mathrm{mek}} \langle 0,\, E,\, \blacklozenge\ t^0,\, (\rho[x \mapsto v^0];\rho^*)\ \blacklozenge\rangle_{\mathrm{f}}$$

can be merged into one rule

$$\langle 0,\, E[v_1\ \square],\, v_2\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mek}} \langle 0,\, E,\, \blacklozenge\ t^0,\, (\rho[x \mapsto v^0];\rho^*)\ \blacklozenge\rangle_{\mathrm{f}}.$$

Modelling Fresh Variables. The (lambda-(i+1)-t) rule of the single-step relation of Suspended MetaML, the (lam-(i+1)-env) rule of the single-step relation of Environmental MetaML, the (conf-lam-(i+1)) rule of the notions of reduction of Environmental MetaML and the (r-conf-lam-(i+1)) rule of the single-transformation relation of the MEK machine require that the variable x_N is globally fresh in the sense that it has not appeared in the current term/configuration being evaluated or in its surrounding context. Being globally fresh is a very strict restriction on x_N.

To loosen the restriction, we may maintain a set of variables $\mathscr{W}$ to keep track of the variables that have lost their freshness due to acting as a fresh variable before in the above-mentioned rules. Then we may interpret "x_N is fresh" as that the variable x_N is locally fresh and does not belong to $\mathscr{W}$. We need formal proofs to support our conjecture.

Modelling Environments by Finitary Functions. In Environmental MetaML, to evaluate a program t_{s}^0, we first construct the initial configuration $\blacklozenge\ t_{\mathrm{s}}^0,\ (\rho_{\mathrm{init}}^{\mathrm{VAR}(t_{\mathrm{s}}^0)};\varepsilon)\ \blacklozenge$ and pass it to the semantics. Moreover, Environmental MetaML reduces $!\langle v^1\rangle$ to the initial configuration $\blacklozenge\ t_{\mathrm{s}}^0,\ (\rho_{\mathrm{init}}^{\mathscr{X}};\varepsilon)\ \blacklozenge$ where the variable set $\mathscr{X}$ contains the variables in the current configuration being evaluated and in its surrounding context. Environmental MetaML models environments as partial functions from variables to denotable terms and needs collecting all variables that exist in the program for the initial configurations.

We propose to model environments as total functions from variables to denotable terms with the restriction that only a finite number of variables do not map to themselves, which we call *finitary functions*.

There are several reasons that finitary functions are suitable for modelling environments in Environmental MetaML. First of all, this model eliminates the extra ad-hoc step of collecting all variables that exist in the program for the initial configurations. The initial meta-environment in an initial configuration is simply the singleton list containing the identity environment. Secondly, this model captures the possibility of computing with open terms while preserving the finitary character of any environment that may arise during real computation. Thirdly, this model still allows us to reason by induction on non-identical mappings of environments. Fourthly, this model allows separating the computer representation (i.e., a finite list of mappings) from the mathematical model (i.e., a finitary function). We conjecture that changing how environments are modelled in Environmental MetaML will not cause fundamental problems in developing the MEK machine.

Machine-checked Proofs. All proofs of the thesis are handwritten, which we believe are error-prone. We may utilise the proof assistants [BC04, BDN09] to check our proofs mechanically.

Abstract Interpretation of MEK Machine. The reason that we developed the MEK machine is to apply a general-purpose framework of developing static analysis [VHM12] to it. With the help of the framework, it is expected that we are able to get a sound and decidable control flow analysis for MetaML.

Bibliography

[ACCL91] Martin Abadi, Luca Cardelli, P-L Curien, and J-J Lévy, *Explicit substitutions*, Journal of functional programming **1** (1991), no. 04, 375–416.

[BC04] Yves Bertot and Pierre Casteran, *Interactive theorem proving and program development*, SpringerVerlag, 2004.

[BDN09] Ana Bove, Peter Dybjer, and Ulf Norell, *A brief overview of agda–a functional language with dependent types*, International Conference on Theorem Proving in Higher Order Logics, Springer, 2009, pp. 73–78.

[Cur85] P. L. Curien, *Categorical combinatory logic*, pp. 130–139, Springer Berlin Heidelberg, Berlin, Heidelberg, 1985.

[FF86] Matthias Felleisen and Daniel P Friedman, *Control operators, the secd-machine, and the λ-calculus*, Indiana University, Computer Science Department, 1986.

[FFF09] Matthias Felleisen, Robert Bruce Findler, and Matthew Flatt, *Semantics engineering with plt redex*, The MIT Press, 2009.

[FH92] Matthias Felleisen and Robert Hieb, *The revised report on the syntactic theories of sequential control and state*, Theoretical computer science **103** (1992), no. 2, 235–271.

[JGS93] Neil D Jones, Carsten K Gomard, and Peter Sestoft, *Partial evaluation and automatic program generation*, Peter Sestoft, 1993.

[Kah87] G. Kahn, *Natural semantics*, pp. 22–39, Springer Berlin Heidelberg, Berlin, Heidelberg, 1987.

[Kle52] S.C. Kleene, *Introduction to metamathematics*, Bibliotheca Mathematica, Wolters-Noordhoff, 1952.

[Lan64] Peter J Landin, *The mechanical evaluation of expressions*, The Computer Journal **6** (1964), no. 4, 308–320.

[Lan66] _____, *The next 700 programming languages*, Communications of the ACM **9** (1966), no. 3, 157–166.

[Mai12] Geoffrey Mainland, *Explicitly heterogeneous metaprogramming with metahaskell*, ACM SIGPLAN Notices, vol. 47, ACM, 2012, pp. 311–322.

[NR04] Gregory Neverov and Paul Roe, *Metaphor: A multi-stage, object-oriented programming language*, pp. 168–185, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[Plo81] Gordon D Plotkin, *A structural approach to operational semantics*.

[PS12] Damien Pous and Davide Sangiorgi, *Enhancements of the bisimulation proof method*, Advanced Topics in Bisimulation and Coinduction (2012).

[Ros96] Kristoffer Høgsbro Rose, *Explicit substitution: tutorial & survey*, Computer Science Department, 1996.

[San11] Davide Sangiorgi, *Introduction to bisimulation and coinduction*, Cambridge University Press, New York, NY, USA, 2011.

[She98] Tim Sheard, *Using metaml: A staged programming language*, Advanced Functional Programming, Springer, 1998, pp. 207–239.

[She01] ———, *Accomplishments and research challenges in meta-programming*, pp. 2–44, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[Tah99a] Walid Taha, *Multi-stage programming: Its theory and applications*, Ph.D. thesis, Oregon Graduate Institute of Science and Technology, 1999.

[Tah99b] ———, *A sound reduction semantics for untyped cbn mutli-stage computation. or, the theory of metaml is non-trival*, ACM SIGPLAN Notices **34** (1999), no. 11, 34–43.

[Tah04] ———, *A gentle introduction to multi-stage programming*, Domain-Specific Program Generation, Springer, 2004, pp. 30–50.

[TS97] Walid Taha and Tim Sheard, *Multi-stage programming with explicit annotations*, ACM SIGPLAN Notices, vol. 32, ACM, 1997, pp. 203–217.

[VHM12] David Van Horn and Matthew Might, *Systematic abstraction of abstract machines*, Journal of Functional Programming **22** (2012), no. 4-5, 705–746.

[WRI+09] Edwin Westbrook, Mathias Ricken, Jun Inoue, Yilong Yao, Tamer Abdelatif, and Walid Taha, *Multi-stage programming for mainstream languages*, Tech. report, Technical Report TR09-02, Rice University, 2009.

Appendix A

Proofs of Chapter 2

A.1 Equivalence of Substitutional Natural Semantics and Substitutional Structural Operational Semantics of MetaML

We demonstrate the equivalence of the substitutional natural semantics and the substitutional structural operational semantics of MetaML.

Lemma 198.

1. *If* $t_1^{i+1} \longrightarrow^{*(i+1)} t_2^{i+1}$, *then* $\lambda x.t_1^{i+1} \longrightarrow^{*(i+1)} \lambda x.t_2^{i+1}$.
2. *If* $t_{11}^{i} \longrightarrow^{*i} t_{12}^{i}$, *then* $t_{11}^{i}\ t_2^{i} \longrightarrow^{*i} t_{12}^{i}\ t_2^{i}$.
3. *If* $t_{21}^{i} \longrightarrow^{*i} t_{22}^{i}$, *then* $v_1^{i}\ t_{21}^{i} \longrightarrow^{*i} v_1^{i}\ t_{22}^{i}$.
4. *If* $t_1^{i} \longrightarrow^{*i} t_2^{i}$, *then* $!t_1^{i} \longrightarrow^{*i} !t_2^{i}$.
5. *If* $t_1^{i+1} \longrightarrow^{*(i+1)} t_2^{i+1}$, *then* $\langle t_1^{i+1} \rangle \longrightarrow^{*i} \langle t_2^{i+1} \rangle$.
6. *If* $t_1^{i} \longrightarrow^{*i} t_2^{i}$, *then* $\sim t_1^{i} \longrightarrow^{*(i+1)} \sim t_2^{i}$.
7. *If* $t_{11}^{i} \longrightarrow^{*i} t_{12}^{i}$, *then* $t_{11}^{i} + t_2^{i} \longrightarrow^{*i} t_{12}^{i} + t_2^{i}$.
8. *If* $t_{21}^{i} \longrightarrow^{*i} t_{22}^{i}$, *then* $v_1^{i} + t_{21}^{i} \longrightarrow^{*i} v_1^{i} + t_{22}^{i}$.

Proof. We proceed by cases.

1. Suppose the length of $t_1^{i+1} \longrightarrow^{*(i+1)} t_2^{i+1}$ is $j \in \mathbb{N}$. By induction on j.
   (a) ($j = 0$). Then $t_1^{i+1} = t_2^{i+1}$. By (refl), $\lambda x.t_1^{i+1} \longrightarrow^{*(i+1)} \lambda x.t_2^{i+1}$.
   (b) Suppose $t_1^{i+1} \longrightarrow^{(j)(i+1)} t_{11}^{i+1} \longrightarrow^{(1)(i+1)} t_2^{i+1}$. By the induction hypothesis, $\lambda x.t_1^{i+1} \longrightarrow^{*(i+1)} \lambda x.t_{11}^{i+1}$. Given $t_{11}^{i+1} \longrightarrow^{i+1} t_2^{i+1}$, by (lambda-(i+1)), $\lambda x.t_{11}^{i+1} \longrightarrow^{i+1} \lambda x.t_2^{i+1}$. By (step) and (trans), $\lambda x.t_1^{i+1} \longrightarrow^{*(i+1)} \lambda x.t_2^{i+1}$.
2. Suppose the length of $t_{11}^{i} \longrightarrow^{*i} t_{12}^{i}$ is $j \in \mathbb{N}$. By induction on j.
   (a) ($j = 0$). Then $t_{11}^{i} = t_{12}^{i}$. By (refl), $t_{11}^{i}\ t_2^{i} \longrightarrow^{*i} t_{12}^{i}\ t_2^{i}$.
   (b) Suppose $t_{11}^{i} \longrightarrow^{(j)(i)} t_{111}^{i} \longrightarrow^{(1)(i)} t_{12}^{i}$. By the induction hypothesis, $t_{11}^{i}\ t_2^{i} \longrightarrow^{*i} t_{111}^{i}\ t_2^{i}$. Given $t_{111}^{i} \longrightarrow^{i} t_{12}^{i}$, by (appL-i), $t_{111}^{i}\ t_2^{i} \longrightarrow^{i} t_{12}^{i}\ t_2^{i}$. By (step) and (trans), $t_{11}^{i}\ t_2^{i} \longrightarrow^{*i} t_{12}^{i}\ t_2^{i}$.

3. Analogous to Case 2.
4. Suppose the length of $t_1^i \longrightarrow^{*i} t_2^i$ is $j \in \mathbb{N}$. By induction on j.
   (a) ($j = 0$). Then $t_1^i = t_2^i$. By (refl), $!t_1^i \longrightarrow^{*i} !t_2^i$.
   (b) Suppose $t_1^i \longrightarrow^{(j)(i)} t_{11}^i \longrightarrow^{(1)(i)} t_2^i$. By the induction hypothesis, $!t_1^i \longrightarrow^{*i} !t_{11}^i$. Given $t_{11}^i \longrightarrow^{i} t_2^i$, by (run-i), $!t_{11}^i \longrightarrow^{i} !t_2^i$. By (step) and (trans), $!t_1^i \longrightarrow^{*i} !t_2^i$.
5. Analogous to Case 4.
6. Analogous to Case 4.
7. Analogous to Case 2.
8. Analogous to Case 3.

□

Theorem 199. *$\longrightarrow^{*i}$ admits every rule from $\Downarrow^i$.*

1. $\lambda x.t^0 \longrightarrow^{*0} \lambda x.t^0$.
2. *If* $t_1^{i+1} \longrightarrow^{*(i+1)} v_2^{i+1}$, *then* $\lambda x.t_1^{i+1} \longrightarrow^{*(i+1)} \lambda x.v_2^{i+1}$.
3. *If* $t_1^0 \longrightarrow^{*0} \lambda x.t_{11}^0$, $t_2^0 \longrightarrow^{*0} v_2^0$, *and* $t_{11}^0[v_2^0/x] \longrightarrow^{*0} v^0$, *then* $t_1^0\ t_2^0 \longrightarrow^{*0} v^0$.
4. *If* $t_1^i \longrightarrow^{*i} v_1^i$ *and* $t_2^i \longrightarrow^{*i} v_2^i$, *then* $t_1^i\ t_2^i \longrightarrow^{*i} v_1^i\ v_2^i$.
5. *If* $t_1^0 \longrightarrow^{*0} \langle v_1^1 \rangle$ *and* $v_1^1 \longrightarrow^{*0} v_2^0$, *then* $!t_1^0 \longrightarrow^{*0} v_2^0$.
6. *If* $t^i \longrightarrow^{*i} v^i$, *then* $!t^i \longrightarrow^{*i} !v^i$.
7. *If* $t^{i+1} \longrightarrow^{*(i+1)} v^{i+1}$, *then* $\langle t^{i+1} \rangle \longrightarrow^{*i} \langle v^{i+1} \rangle$.
8. *If* $t^0 \longrightarrow^{*0} \langle v^1 \rangle$, *then* $\sim t^0 \longrightarrow^{*1} v^1$.
9. *If* $t^i \longrightarrow^{*i} v^i$, *then* $\sim t^i \longrightarrow^{*(i+1)} \sim v^i$.
10. $x \longrightarrow^{*(i+1)} x$.
11. $n \longrightarrow^{*i} n$.
12. *If* $t_1^0 \longrightarrow^{*0} n_1$, $t_2^0 \longrightarrow^{*0} n_2$, *and* $n = n_1 + n_2$, *then* $t_1^0 + t_2^0 \longrightarrow^{*0} n$.
13. *If* $t_1^i \longrightarrow^{*i} v_1^i$ *and* $t_2^i \longrightarrow^{*i} v_2^i$, *then* $t_1^i + t_2^i \longrightarrow^{*i} v_1^i + v_2^i$.

Proof. We proceed by cases.

1. By (refl), $\lambda x.t^0 \longrightarrow^{*0} \lambda x.t^0$.

2. By Lemma 198 Case 1.

3. By Case 4, $t_1^0\ t_2^0 \longrightarrow^{*0} (\lambda x.t_{11}^0)\ v_2^0$. By (app-0), $(\lambda x.t_{11}^0)\ v_2^0 \longrightarrow^0 t_{11}^0[v_2^0/x]$. By (step), $(\lambda x.t_{11}^0)\ v_2^0 \longrightarrow^{*0} t_{11}^0[v_2^0/x]$. Together with $t_{11}^0[v_2^0/x] \longrightarrow^{*0} v^0$, by (trans), $t_1^0\ t_2^0 \longrightarrow^{*0} v^0$.

4. By Lemma 198 Case 2, $t_1^i\ t_2^i \longrightarrow^{*i} v_1^i\ t_2^i$. By Lemma 198 Case 3, $v_1^i\ t_2^i \longrightarrow^{*i} v_1^i\ v_2^i$. By (trans), $t_1^i\ t_2^i \longrightarrow^{*i} v_1^i\ v_2^i$.

5. By Case 6, $!t_1^0 \longrightarrow^{*0} !\langle v_1^1 \rangle$. By (run-0), $!\langle v_1^1 \rangle \longrightarrow^0 v_1^1$. By (step), $!\langle v_1^1 \rangle \longrightarrow^{*0} v_1^1$. Together with $v_1^1 \longrightarrow^{*0} v_2^0$, by (trans), $!t_1^0 \longrightarrow^{*0} v_2^0$.

6. By Lemma 198 Case 4.

7. By Lemma 198 Case 5.

8. By Case 9, $\sim t^0 \longrightarrow^{*1} \sim\langle v^1 \rangle$. By (splice-1), $\sim\langle v^1 \rangle \longrightarrow^1 v^1$. By (step) and (trans), $\sim t^0 \longrightarrow^{*1} v^1$.

9. By Lemma 198 Case 6.

10. By (refl), $x \longrightarrow^{*(i+1)} x$.

11. By (refl), $n \longrightarrow^{*i} n$.

12. By Case 13, $t_1^0 + t_2^0 \longrightarrow^{*0} n_1 + n_2$. By (plus-0), $n_1 + n_2 \longrightarrow^0 n$. By (step) and (trans), $t_1^0 + t_2^0 \longrightarrow^{*0} n$.

13. By Lemma 198 Case 7, $t_1^i + t_2^i \longrightarrow^{*i} v_1^i + t_2^i$. By Lemma 198 Case 8, $v_1^i + t_2^i \longrightarrow^{*i} v_1^i + v_2^i$. By (trans), $t_1^i + t_2^i \longrightarrow^{*i} v_1^i + v_2^i$.

□

We demonstrate the soundness of the substitutional structural operational semantics with respect to the substitutional natural semantics of MetaML.

Corollary 200 (Soundness of Substitutional Structural Operational Semantics w.r.t Substitutional Natural Semantics)**.** *If $t^i \Downarrow^i v^i$ then $t^i \longrightarrow^{*i} v^i$.*

Proof. We proceed by induction on the derivation of $t^i \Downarrow^i v^i$.

Case 1. (lambda-0). By Theorem 199 Case 1.

Case 2. (lambda-(i+1)). By Theorem 199 Case 2.

Case 3. (app-0). By Theorem 199 Case 3.

Case 4. (app-(i+1)). By Theorem 199 Case 4.

Case 5. (run-0). By Theorem 199 Case 5.

Case 6. (run-(i+1)). By Theorem 199 Case 6.

Case 7. (code-i). By Theorem 199 Case 7.

Case 8. (splice-1). By Theorem 199 Case 8.

Case 9. (splice-(i+2)). By Theorem 199 Case 9.

Case 10. (ref-(i+1)). By Theorem 199 Case 10.

Case 11. (num-i). By Theorem 199 Case 11.

Case 12. (plus-0). By Theorem 199 Case 12.

Case 13. (plus-(i+1)). By Theorem 199 Case 13.

□

Theorem 201.

1. *If* $\lambda x.t_1 \longrightarrow^{*i} v_2$, *then either*
   (a) $i = 0$ *and* $v_2 = \lambda x.t_1$ *; or*
   (b) $i = (j+1) > 0$, $t_1 \longrightarrow^{*(j+1)} v_{21}^{j+1}$, *and* $v_2 = \lambda x.v_{21}^{j+1}$.
2. *If* $t_1\ t_2 \longrightarrow^{*i} v$, *then* $t_1 \longrightarrow^{*i} v_1^i$, $t_2 \longrightarrow^{*i} v_2^i$, *and either*
   (a) $i = 0$, $v_1^0 = \lambda x.t_{11}^0$, *and* $t_{11}^0[v_2^0/x] \longrightarrow^{*0} v$; *or*
   (b) $i = (j+1) > 0$, $v = v_1^{j+1}\ v_2^{j+1}$.
3. *If* $!t_1 \longrightarrow^{*i} v_2$, *then* $t_1 \longrightarrow^{*i} v_1^i$, *and either*
   (a) $i = 0$, $v_1^0 = \langle v_{11}^1 \rangle$, *and* $v_{11}^1 \longrightarrow^{*0} v_2$; *or*
   (b) $i = (j+1) > 0$, *and* $v_2 = !v_1^{j+1}$.
4. *If* $\langle t_1 \rangle \longrightarrow^{*i} v_2$, *then* $t_1 \longrightarrow^{*(i+1)} v_1^{i+1}$, *and* $v_2 = \langle v_1^{i+1} \rangle$.
5. *If* $\sim t_1 \longrightarrow^{*i} v_2$, *then* $i = (j+1) > 0$, $t_1 \longrightarrow^{*j} v_1^j$, *and either*
   (a) $j = 0$, $v_1^0 = \langle v_{11}^1 \rangle$, *and* $v_2 = v_{11}^1$; *or*
   (b) $j = (k+1) > 0$, *and* $v_2 = \sim v_1^{k+1}$.
6. *If* $x \longrightarrow^{*i} v$, *then* $i = (j+1) > 0$, *and* $v = x$.
7. *If* $n \longrightarrow^{*i} v$, *then* $v = n$.
8. *If* $t_1 + t_2 \longrightarrow^{*i} v$, *then* $t_1 \longrightarrow^{*i} v_1^i$, $t_2 \longrightarrow^{*i} v_2^i$, *and either*
   (a) $i = 0$, $v_1^0 = n_1$, $v_2^0 = n_2$, *and* $v = n_1 + n_2$; *or*

(b) $i = (j+1) > 0$, $v = v_1^{j+1} + v_2^{j+1}$.

Proof. We proceed by cases.

1. We proceed by cases on i.
   (a) $(i = 0)$. Then, $\lambda x.t_1 \in \text{VALUE}^0$, and $\lambda x.t_1 \not\longrightarrow^0$. By (refl), $v_2 = \lambda x.t_1^0$.
   (b) $(i = (j+1) > 0)$. Suppose the length of $\lambda x.t_1 \longrightarrow^{*(j+1)} v_2$ is $k \in \mathbb{N}$. By induction on k.
      i. $(k = 0)$. Then, $v_2 = \lambda x.t_1$, and $t_1 \in \text{VALUE}^{i+1}$. By (refl), $t_1 \longrightarrow^{*(j+1)} t_1$.
      ii. Suppose $\lambda x.t_1 \longrightarrow^{(1)(j+1)} t_2 \longrightarrow^{(k)(j+1)} v_2$. Proceed by cases on $\lambda x.t_1 \longrightarrow^{j+1} t_2$. The only case is (lambda-(i+1)). Then, $t_1 \longrightarrow^{j+1} t_{21}$, and $t_2 = \lambda x.t_{21}$. Given $\lambda x.t_{21} \longrightarrow^{(k)(j+1)} v_2$, by the induction hypothesis, $t_{21} \longrightarrow^{*(j+1)} v_{21}^{j+1}$, and $v_2 = \lambda x.v_{21}^{j+1}$. By (step) and (trans), $t_1 \longrightarrow^{*(j+1)} v_{21}^{j+1}$.

2. Suppose t_1 does not multi-step to a value. Then, (appL-i) is the only rule that is applicable, and it keeps applying, in which case $t_1\ t_2$ can never multi-step to v. Hence, t_1 has to multi-step to a value, i.e., $t_1 \longrightarrow^{*i} v_1^i$. Analogously, t_2 has to multi-step to a value, i.e., $t_2 \longrightarrow^{*i} v_2^i$. We have $t_1\ t_2 \longrightarrow^{*i} v_1^i\ v_2^i$. We proceed by cases on i.
   (a) $(i = 0)$. Assume $v_1^0 \neq \lambda x.t_{11}^0$. Then, $v_1^0\ v_2^0 \not\longrightarrow^0$ and $v_1^0\ v_2^0 \notin \text{VALUE}^0$. We get $t_1\ t_2 \longrightarrow^{*0} v_1^0\ v_2^0 \not\longrightarrow^{*0} v$, which contradicts with $t_1\ t_2 \longrightarrow^{*0} v$. Hence, $v_1^0 = \lambda x.t_{11}^0$.
   Given $v_1^0\ v_2^0 = (\lambda x.t_{11}^0)\ v_2^0$, (app-0) is the only rule that is applicable. By (app-0), $(\lambda x.t_{11}^0)\ v_2^0 \longrightarrow^0 t_{11}^0[v_2^0/x]$. Given $(\lambda x.t_{11}^0)\ v_2^0 \longrightarrow^{*0} v$, by the determinism of the language, $t_{11}^0[v_2^0/x] \longrightarrow^{*0} v$.
   (b) $(i = (j+1) > 0)$. Then $v_1^{j+1}\ v_2^{j+1} \in \text{VALUE}^{j+1}$. $v = v_1^{j+1}\ v_2^{j+1}$.

3. Suppose t_1 does not multi-step to a value. Then, (run-i) is the only rule that is applicable, and it keeps applying, in which case $!t_1$ can never multi-step to v_2. Hence, t_1 has to multi-step to a value, i.e., $t_1 \longrightarrow^{*i} v_1^i$. We proceed by cases on i.
   (a) $(i = 0)$. Assume $v_1^0 \neq \langle v_{11}^1 \rangle$. Then, $!v_1^0 \not\longrightarrow^0$ and $!v_1^0 \notin \text{VALUE}^0$. We get $!t_1 \longrightarrow^{*0} !v_1^0 \not\longrightarrow^{*0} v_2$, which contradicts with $!t_1 \longrightarrow^{*i} v_2$. Hence, $v_1^0 = \langle v_{11}^1 \rangle$.
   Given $!v_1^0 = !\langle v_{11}^1 \rangle$, (run-0) is the only rule that is applicable. By (run-0), $!\langle v_{11}^1 \rangle \longrightarrow^0 v_{11}^1$. Given $!\langle v_{11}^1 \rangle \longrightarrow^{*0} v_2$, by the determinism of the language, $v_{11}^1 \longrightarrow^{*0} v_2$.
   (b) $(i = (j+1) > 0)$. Then, $!v_1^{j+1} \in \text{VALUE}^{j+1}$. $v_2 = !v_1^{j+1}$.

4. Suppose t_1 does not multi-step to a value. Then, (code-i) is the only rule that is applicable, and it keeps applying, in which case $\langle t_1 \rangle$ can never multi-step to v_2. Hence, t_1 has to multi-step to a value, i.e., $t_1 \longrightarrow^{*(i+1)} v_1^{i+1}$. Then, $\langle v_1^{i+1} \rangle \in \text{VALUE}^i$. $v_2 = \langle v_1^{i+1} \rangle$.

5. Analogous to Case 3.

6. By (refl), $x \longrightarrow^{*(j+1)} x$. $x \in \text{VALUE}^{j+1}$. $v = x$.

7. By (refl), $n \longrightarrow^{*i} n$. $n \in \text{VALUE}^i$. $v = n$.

8. Analogous to Case 2.

□

We demonstrate the completeness of the substitutional structural operational semantics with respect to the substitutional natural semantics of MetaML.

Corollary 202 (Completeness of Substitutional Structural Operational Semantics w.r.t Substitutional Natural Semantics)**.** *If* $t^i \longrightarrow^{*i} v^i$ *then* $t^i \Downarrow^i v^i$.

Proof. We proceed by the structure of $t^i \in \text{TERM}^i$ and by induction on the size of derivation of $t^i \longrightarrow^{*i} v^i$.

Case 1. ($t^i = x$). By Theorem 201, $i = (j+1) > 0$, and $v = x$. By (ref-(i+1)), $x \Downarrow^{j+1} x$.

Case 2. ($t^i = t_1^i\ t_2^i$). By Theorem 201, $t_1^i \longrightarrow^{*i} v_1^i$, $t_2^i \longrightarrow^{*i} v_2^i$, and either

- $i = 0$, $v_1^0 = \lambda x.t_{11}^0$, and $t_{11}^0[v_2^0/x] \longrightarrow^{*0} v^0$; or
- $i = (j+1) > 0$, $v^{j+1} = v_1^{j+1}\ v_2^{j+1}$.

Given $t_1^i \longrightarrow^{*i} v_1^i$ and $t_2^i \longrightarrow^{*i} v_2^i$, by the induction hypothesis, $t_1^i \Downarrow^i v_1^i$ and $t_2^i \Downarrow^i v_2^i$. Proceed by cases on i.

Case i. ($i = 0$). Obviously the derivation of $t_{11}^0[v_2^0/x] \longrightarrow^{*0} v^0$ is smaller than that of $t_1^0\ t_2^0 \longrightarrow^{*0} v^0$. By the induction hypothesis, $t_{11}^0[v_2^0/x] \Downarrow^{*0} v^0$. Together with $t_1^0 \Downarrow^0 \lambda x.t_{11}^0$ and $t_2^0 \Downarrow^0 v_2^0$, by (app-0), $t_1^0\ t_2^0 \Downarrow^0 v^0$.

Case ii. ($i = (j+1) > 0$). Given $t_1^{j+1} \Downarrow^{j+1} v_1^{j+1}$ and $t_2^{j+1} \Downarrow^{j+1} v_2^{j+1}$, by (app-(i+1)), $t_1^{j+1}\ t_2^{j+1} \Downarrow^{j+1} v_1^{j+1}\ v_2^{j+1}$.

Case 3. ($t^i = \lambda x.t_1^i$). By Theorem 201, either

- $i = 0$ and $v^0 = \lambda x.t_1^0$; or
- $i = (j+1) > 0$, $t_1^{j+1} \longrightarrow^{*(j+1)} v_2^{j+1}$, and $v^{j+1} = \lambda x.v_2^{j+1}$.

Proceed by cases on i.

Case i. ($i = 0$). By (lambda-0), $\lambda x.t_1^0 \Downarrow^0 \lambda x.t_1^0$.

Case ii. ($i = (j+1) > 0$). By the induction hypothesis, $t_1^{j+1} \Downarrow^{j+1} v_2^{j+1}$. By (lambda-(i+1)), $\lambda x.t_1^{j+1} \Downarrow^{j+1} \lambda x.v_2^{j+1}$.

Case 4. ($t^i = \langle t_1^{i+1} \rangle$). By Theorem 201, $t_1^{i+1} \longrightarrow^{*(i+1)} v_1^{i+1}$, and $v^i = \langle v_1^{i+1} \rangle$. By the induction hypothesis, $t_1^{i+1} \Downarrow^{i+1} v_1^{i+1}$. By (code-i), $\langle t_1^{i+1} \rangle \Downarrow^{i+1} \langle v_1^{i+1} \rangle$.

Case 5. ($t^{i+1} = {\sim}t_1^i$). By Theorem 201, $t_1^i \longrightarrow^{*i} v_1^i$, and either

1. $i = 0$, $v_1^0 = \langle v_{11}^1 \rangle$, and $v^1 = v_{11}^1$; or
2. $i = (j+1) > 0$, and $v^{j+2} = {\sim}v_1^{j+1}$.

Given $t_1^i \longrightarrow^{*i} v_1^i$, by the induction hypothesis, $t_1^i \Downarrow^i v_1^i$. Proceed by cases on i.

Case i. $(i = 0)$. Given $t_1^0 \Downarrow^i \langle v_{11}^1 \rangle$, by (splice-1), ${\sim}t_1^0 \Downarrow^1 v_{11}^1$.

Case ii. $(i = (j+1) > 0)$. By (splice-(i+2)), ${\sim}t_1^{j+1} \Downarrow^{j+2} {\sim}v_1^{j+1}$.

Case 6. $(t^i = !t_1^i)$. Analogous to Case 5.

Case 7. $(t^i = n)$. By Theorem 201, $v^i = n$. By (num-i), $n \Downarrow^i n$.

Case 8. $(t^i = t_1^i + t_2^i)$. Analogous to Case 2.

□

We demonstrate the soundness and completeness of the substitutional structural operational semantics with respect to the substitutional natural semantics of MetaML.

Theorem 203 (Soundness and Completeness of Substitutional Structural Operational Semantics w.r.t Substitutional Natural Semantics)**.** *For any $i \in \mathbb{N}$, $t^i \Downarrow^i v^i$ if and only if $t^i \longrightarrow^{*i} v^i$.*

Proof. It directly follows Corollaries 200 and 202. □

We prove the Kleene equality of evaluators $eval_{\text{MetaML:SubNat}}(t)$ and $eval_{\text{MetaML:SubSOS}}(t)$.

Theorem 204 (Kleene Equality of Evaluators)**.** *For any $t \in \text{PRGM}_{\text{MetaML}}$, $eval_{\text{MetaML:SubNat}}(t)$ is Kleene equal to $eval_{\text{MetaML:SubSOS}}(t)$.*

Proof. For any $t \in \text{PRGM}_{\text{MetaML}}$, by Theorem 203, $t^0 \Downarrow^0 v^0$ if and only if $t^0 \longrightarrow^{*0} v^0$.

We first show if $eval_{\text{MetaML:SubNat}}(t) = a$ where $a \in \text{ANS}_{\text{MetaML}}$, then $eval_{\text{MetaML:SubSOS}}(t) = a$.

Case 1. If $eval_{\text{MetaML:SubNat}}(t) = \texttt{function}$, then $t \Downarrow^0 \lambda x.t'^0$. Then $t \longrightarrow^{0*} \lambda x.t'^0$.
We have $eval_{\text{MetaML:SubSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:SubNat}}(t) = \texttt{code}$, then $t \Downarrow^0 \langle v^1 \rangle$. Then $t \longrightarrow^{0*} \langle v^1 \rangle$.
We have $eval_{\text{MetaML:SubSOS}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:SubNat}}(t) = n$, then $t \Downarrow^0 n$. Then $t \longrightarrow^{0*} n$.
We have $eval_{\text{MetaML:SubSOS}}(t) = n$.

We then show if $eval_{\text{MetaML:SubSOS}}(t) = a$ where $a \in \text{ANS}_{\text{MetaML}}$, then $eval_{\text{MetaML:SubNat}}(t) = a$.

Case 1. If $eval_{\text{MetaML:SubSOS}}(t) = \texttt{function}$, then $t \longrightarrow^{0*} \lambda x.t'^0$. Then $t \Downarrow^0 \lambda x.t'^0$.
We have $eval_{\text{MetaML:SubNat}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:SubSOS}}(t) = \texttt{code}$, then $t \longrightarrow^{0*} \langle v^1 \rangle$. Then $t \Downarrow^0 \langle v^1 \rangle$.
We have $eval_{\text{MetaML:SubNat}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:SubSOS}}(t) = n$, then $t \longrightarrow^{0*} n$. Then $t \Downarrow^0 n$.
We have $eval_{\text{MetaML:SubNat}}(t) = n$.

We observe that $eval_{\text{MetaML:SubNat}}(t)$ is undefined if and only if $eval_{\text{MetaML:SubSOS}}(t)$ is undefined. Therefore, $eval_{\text{MetaML:SubNat}}(t)$ is Kleene equal to $eval_{\text{MetaML:SubSOS}}(t)$. □

Appendix B

Proofs of Chapter 3

B.1 Equivalence of ISWIM and Explicit ISWIM

We demonstrate the equivalence of the substitutional structural operational semantics of ISWIM and the structural operational semantics of Explicit ISWIM. We use subscripts "$_{\text{sub}}$" and "$_{\text{exp}}$" to differentiate the syntax of (Substitutional) ISWIM from the syntax of Explicit ISWIM.

B.1.1 Bisimulation Relation

We first introduce a bisimulation relation that relates (Substitutional) ISWIM terms to Explicit ISWIM terms.

Definition 205 (Bisimulation Relation)**.** Define the bisimulation relation $\simeq$ to be a binary relation up to alpha equivalence between the set of terms in (Substitutional) ISWIM and the set of terms in Explicit ISWIM.

$$\simeq \subseteq \text{TERM}_{\text{sub}} \times \text{TERM}_{\text{exp}}$$

$$\frac{}{x \simeq x}\ \text{(var-sim)} \qquad \frac{t_{a_1} \simeq t_{b_1} \quad t_{a_2} \simeq t_{b_2}}{t_{a_1}\, t_{a_2} \simeq t_{b_1}\, t_{b_2}}\ \text{(app-sim)} \qquad \frac{t_a \simeq t_b}{(\lambda x.t_a) \simeq (\lambda x.t_b)}\ \text{(lam-sim)}$$

$$\frac{}{n \simeq n}\ \text{(num-sim)} \qquad \frac{t_{a_1} \simeq t_{b_1} \quad t_{a_2} \simeq t_{b_2}}{t_{a_1} + t_{a_2} \simeq t_{b_1} + t_{b_2}}\ \text{(plus-sim)} \qquad \frac{t_a \simeq t_b \quad w_a \simeq w_b}{t_a[w_a/x] \simeq t_b[x := w_b]}\ \text{(subst-sim)}$$

Remark 206. We explain each rule of the relation as follows.

(var-sim) A variable x from (Substitutional) ISWIM relates to the same variable x from Explicit ISWIM.

(app-sim) An application $t_{a_1}\, t_{a_2}$ from (Substitutional) ISWIM and an application $t_{b_1}\, t_{b_2}$ from Explicit ISWIM are related, if their operators t_{a_1} and t_{b_1} are related, and their operands t_{a_2} and t_{b_2} are related.

(lam-sim) A lambda abstraction $\lambda x.t_a$ from (Substitutional) ISWIM and a lambda abstraction $\lambda x.t_b$ from Explicit ISWIM with the same bound variable are related, if their bodies t_a and t_b are related.

(num-sim) A natural number n from (Substitutional) ISWIM relates to the same natural number n from Explicit ISWIM.

(plus-sim) An addition of terms $t_{a_1} + t_{a_2}$ from (Substitutional) ISWIM and an addition of terms $t_{b_1} + t_{b_2}$ from Explicit ISWIM are related, if their first operands t_{a_1} and t_{b_1} are related, and their second operands t_{a_2} and t_{b_2} are related.

(subst-sim) A term surrounded by a substitution $t_a[w_a/x]$ from (Substitutional) ISWIM and a term surrounded by an explicit substitution $t_b[x := w_b]$ from Explicit ISWIM are related, if the terms t_a and t_b are related, and the denotable terms w_a and w_b are related.

Remark 207. The bisimulation relation $\sim$ is up to alpha equivalence. We immediately have: (1) if $t_{a_1} \simeq t_b$ and $t_{a_1} \sim_\alpha t_{a_2}$ then $t_{a_2} \simeq t_b$, and (2) if $t_a \simeq t_{b_1}$ and $t_{b_1} \sim_\alpha t_{b_2}$ then $t_a \simeq t_{b_2}$.

B.1.2 Unload Function

We define $U(t)$ to unload an Explicit ISWIM term t to (Substitutional) ISWIM.

Definition 208 (Unload Function)**.** Define the unloading function U to be a total function from the set of terms in Explicit ISWIM to the set of terms in (Substitutional) ISWIM.

$$U \;:\; \text{TERM}_{\text{exp}} \longrightarrow \text{TERM}_{\text{sub}}$$

$$\begin{array}{rcl} U(x) & = & x \\ U(t_1\, t_2) & = & U(t_1)\, U(t_2) \\ U(\lambda x.t) & = & \lambda x.U(t) \\ U(n) & = & n \\ U(t_1 + t_2) & = & U(t_1) + U(t_2) \\ U(t[x := w]) & = & U(t)[U(w)/x] \end{array}$$

Lemma 209 (Equality of Related Terms w.r.t. Unload Function)**.** *If $t_a \simeq t_b$, then $t_a = U(t_b)$.*

Proof. We proceed by structural induction on $t_a \simeq t_b$.

Case 1. (var-sim). Then, $t_a = t_b = x$. We immediately get $x = U(x)$.

Case 2. (app-sim). Then, $t_a = t_{a_1}\, t_{a_2}$ and $t_b = t_{b_1}\, t_{b_2}$ where $t_{a_1} \simeq t_{b_1}$ and $t_{a_2} \simeq t_{b_2}$. By the induction hypothesis, $t_{a_1} = U(t_{b_1})$ and $t_{a_2} = U(t_{b_2})$. Hence $U(t_{b_1}\, t_{b_2}) = U(t_{b_1})\, U(t_{b_2}) = t_{a_1}\, t_{a_2}$.

Case 3. (lam-sim). Then, $t_a = \lambda x.t_{a_1}$ and $t_b = \lambda x.t_{b_1}$ where $t_{a_1} \simeq t_{b_1}$. By the induction hypothesis, $t_{a_1} = U(t_{b_1})$. Hence $U(\lambda x.t_{b_1}) = \lambda x.U(t_{b_1}) = \lambda x.t_{b_1}$.

Case 4. (num-sim). Then, $t_a = t_b = n$. We immediately get $U(n) = n$.

Case 5. (plus-sim). Then, $t_a = t_{a_1} + t_{a_2}$ and $t_b = t_{b_1} + t_{b_2}$ where $t_{a_1} \simeq t_{b_1}$ and $t_{a_2} \simeq t_{b_2}$. By the induction hypothesis, $t_{a_1} = U(t_{b_1})$ and $t_{a_2} = U(t_{b_2})$. Hence $U(t_{b_1} + t_{b_2}) = U(t_{b_1}) + U(t_{b_2}) = t_{a_1} + t_{a_2}$.

Case 6. (subst-sim). Then, $t_a = t_{a_1}[w_{a_1}/x]$ and $t_b = t_{b_1}[x := w_{b_1}]$ where $t_{a_1} \simeq t_{b_1}$ and $w_{a_1} \simeq w_{b_1}$. By the induction hypothesis, $t_{a_1} = U(t_{b_1})$ and $w_{a_1} = U(w_{b_1})$. Hence $U(t_{b_1}[x := w_{b_1}]) = U(t_{b_1})[U(w_{b_1})/x] = t_{a_1}[w_{a_1}/x]$

□

B.1.3 Substitution Normal Form

In Explicit ISWIM, the terms that cannot perform single-step substitution reduction are in substitution normal form.

Definition 210 (Substitution Normal Form)**.** A term $t \in \text{TERM}_{\text{exp}}$ is in *substitution normal form* if and only if $t \not\longrightarrow^{\text{x}}$.

Remark 211. We use s with or without any subscript or superscript as a metavariable to range over the set of terms of Explicit ISWIM in substitution normal form.

Remark 212. An Explicit ISWIM term in substitution normal form is not necessarily in the normal form with respect to the single-step relation $\longrightarrow$. For example, $(\lambda x.t)\ v$ is in substitution normal form but is not in the normal form with respect to the single-step relation $\longrightarrow$.

Lemma 213. *If* $t_a \simeq t_{b_1}[x := w_{b_1}]$, *then* $t_{b_1} \longrightarrow^{\text{x}*} s_{b_1}$, $s_{b_1}[x := w_{b_1}] \longrightarrow^{\text{x}*} s_{b_2}$, *and* $t_a \simeq s_{b_2}$.

Proof. We proceed by structural induction on $t_a \simeq t_{b_1}[x := w_{b_1}]$. Only (subst-sim) applies. Let $t_a = t_{a_1}[w_{a_1}/x]$ and we have

$$\frac{t_{a_1} \simeq t_{b_1} \quad w_{a_1} \simeq w_{b_1}}{t_{a_1}[w_{a_1}/x] \simeq t_{b_1}[x := w_{b_1}],}$$

We proceed by cases on $t_{a_1} \in \text{TERM}_{\text{sub}}$.

Case 1. ($t_{a_1} = x$) We have

$$\frac{x \simeq t_{b_1} \quad w_{a_1} \simeq w_{b_1}}{x[w_{a_1}/x] \simeq t_{b_1}[x := w_{b_1}].}$$

Then, $x[w_{a_1}/x] = w_{a_1}$. We proceed by cases on $x \simeq t_{b_1}$.

Case i. (var-sim). Let $t_{b_1} = x$. Then, $x \longrightarrow^{\text{x}*} x$, $x[x := w_{b_1}] \longrightarrow^{\text{x}} w_{b_1}$, and $w_{a_1} \simeq w_{b_1}$.

Case ii. (subst-sim). Let $t_{b_1} = t_{b_{11}}[x_1 := w_{b_{11}}]$. Given $x \simeq t_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t_{b_{11}} \longrightarrow^{\text{x}*} s_{b_{11}}$, $s_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{\text{x}*} s_{b_{12}}$, and $x \simeq s_{b_{12}}$. We proceed by cases on $x \simeq s_{b_{12}}$. The only case is (var-sim), so let $s_{b_{12}} = x$. Then, $t_{b_1} \longrightarrow^{\text{x}*} x$, $x[x := w_{b_1}] \longrightarrow^{\text{x}} w_{b_1}$, and $w_{a_1} \simeq w_{b_1}$.

Case 2. ($t_{a_1} = x_0$ and $x_0 \not\equiv x$). We have

$$\frac{x_0 \simeq t_{b_1} \quad w_{a_1} \simeq w_{b_1}}{x_0[w_{a_1}/x] \simeq t_{b_1}[x := w_{b_1}].}$$

Then, $x_0[w_{a_1}/x] = x_0$. We proceed by cases on $x_0 \simeq t_{b_1}$.

Case i. (var-sim). Let $t_{b_1} = x_0$. Then, $x_0 \longrightarrow^{\text{x}*} x_0$, $x_0[x := w_{b_1}] \longrightarrow^{\text{x}} x_0$, and $x_0 \simeq x_0$.

Case ii. (subst-sim). Let $t_{b_1} = t_{b_{11}}[x_1 := w_{b_{11}}]$. Given $x_0 \simeq t_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t_{b_{11}} \longrightarrow^{x*} s_{b_{11}}$, $s_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{x*} s_{b_{12}}$, and $x_0 \simeq s_{b_{12}}$. We proceed by cases on $x_0 \simeq s_{b_{12}}$. The only case is (var-sim), so let $s_{b_{12}} = x_0$. Then, $t_{b_1} \longrightarrow^{x*} x_0$, $x_0[x := w_{b_1}] \longrightarrow^{x*} x_0$, and by (var-sim) $x_0 \simeq x_0$.

Case 3. $(t_{a_1} = (t_{a_{11}}\ t_{a_{12}}))$. We have

$$\frac{(t_{a_{11}}\ t_{a_{12}}) \simeq t_{b_1} \quad w_{a_1} \simeq w_{b_1}}{(t_{a_{11}}\ t_{a_{12}})[w_{a_1}/x] \simeq t_{b_1}[x := w_{b_1}]}.$$

Then, $(t_{a_{11}}\ t_{a_{12}})[w_{a_1}/x] = (t_{a_{11}}[w_{a_1}/x])\ (t_{a_{12}}[w_{a_1}/x])$. We proceed by cases on $(t_{a_{11}}\ t_{a_{12}}) \simeq t_{b_1}$.

Case i. (app-sim). Let $t_{b_1} = (t_{b_{11}}\ t_{b_{12}})$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $t_{a_{12}} \simeq t_{b_{12}}$. We have $(t_{b_{11}}\ t_{b_{12}}) \longrightarrow^{x*} (t_{b_{11}}\ t_{b_{12}})$ and $(t_{b_{11}}\ t_{b_{12}})[x := w_{b_1}] \longrightarrow^{x} (t_{b_{11}}[x := w_{b_1}])\ (t_{b_{12}}[x := w_{b_1}])$. By (subst-sim) and (app-sim), we get $(t_{a_{11}}[w_{a_1}/x])\ (t_{a_{12}}[w_{a_1}/x]) \simeq (t_{b_{11}}[x := w_{b_1}])\ (t_{b_{12}}[x := w_{b_1}])$.

Case ii. (subst-sim). Let $t_{b_1} = t_{b_{11}}[x_1 := w_{b_{11}}]$. Given $(t_{a_{11}}\ t_{a_{12}}) \simeq t_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t_{b_{11}} \longrightarrow^{x*} s_{b_{11}}$, $s_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{x*} s_{b_{12}}$, and $(t_{a_{11}}\ t_{a_{12}}) \simeq s_{b_{12}}$. We proceed by cases on $(t_{a_{11}}\ t_{a_{12}}) \simeq s_{b_{12}}$. The only case is (app-sim), so let $s_{b_{12}} = (t_{b_{121}}\ t_{b_{122}})$ where $t_{a_{11}} \simeq t_{b_{121}}$ and $t_{a_{12}} \simeq t_{b_{122}}$. Then, $t_{b_1} \longrightarrow^{x*} (t_{b_{121}}\ t_{b_{122}})$ and $(t_{b_{121}}\ t_{b_{122}})[x := w_{b_1}] \longrightarrow^{x} (t_{b_{121}}[x := w_{b_1}])\ (t_{b_{122}}[x := w_{b_1}])$. By (subst-sim) and (app-sim), we get $(t_{a_{11}}[w_{a_1}/x])\ (t_{a_{12}}[w_{a_1}/x]) \simeq (t_{b_{121}}[x := w_{b_1}])\ (t_{b_{122}}[x := w_{b_1}])$.

Case 4. $(t_{a_1} = \lambda x_0.t_{a_{11}})$. We have

$$\frac{(\lambda x_0.t_{a_{11}}) \simeq t_{b_1} \quad w_{a_1} \simeq w_{b_1}}{(\lambda x_0.t_{a_{11}})[w_{a_1}/x] \simeq t_{b_1}[x := w_{b_1}]}.$$

Then, $(\lambda x_0.t_{a_{11}})[w_{a_1}/x] = \lambda x_1.t_{a_{11}}[x_1/x_0][w_{a_1}/x]$ where $x_1 \notin FV(\lambda x_0.t_{a_{11}}) \cup FV(w_{a_1}) \cup \{x\}$. We proceed by cases on $(\lambda x_0.t_{a_{11}}) \simeq t_{b_1}$.

Case i. (lam-sim). Let $t_{b_1} = \lambda x_0.t_{b_{11}}$ where $t_{a_{11}} \simeq t_{b_{11}}$. We have $\lambda x_0.t_{b_{11}} \longrightarrow^{x*} \lambda x_0.t_{b_{11}}$ and $(\lambda x_0.t_{b_{11}})[x := w_{b_1}] \longrightarrow^{x} \lambda x_2.t_{b_{11}}[x_0 := x_2][x := w_{b_1}]$ where $x_2 \notin FV(\lambda x_0.t_{b_{11}}) \cup FV(w_{b_1}) \cup \{x\}$.

Let $x_3 \notin FV(\lambda x_0.t_{a_{11}}) \cup FV(w_{a_1}) \cup FV(\lambda x_0.t_{b_{11}}) \cup FV(w_{b_1}) \cup \{x\}$, then by the definition of α-equivalence, we get $\lambda x_1.t_{a_{11}}[x_1/x_0][w_{a_1}/x] \sim_\alpha \lambda x_3.t_{a_{11}}[x_3/x_0][w_{a_1}/x]$ and $\lambda x_2.t_{b_{11}}[x_0 := x_2][x := w_{b_1}] \sim_\alpha \lambda x_3.t_{b_{11}}[x_0 := x_3][x := w_{b_1}]$.

By (lam-sim) and (subst-sim), we get $\lambda x_3.t_{a_{11}}[x_3/x_0][w_{a_1}/x] \simeq \lambda x_3.t_{b_{11}}[x_0 := x_3][x := w_{b_1}]$. Hence we have $\lambda x_1.t_{a_{11}}[x_1/x_0][w_{a_1}/x] \simeq \lambda x_2.t_{b_{11}}[x_0 := x_2][x := w_{b_1}]$.

Case ii. (subst-sim). Let $t_{b_1} = t_{b_{11}}[x_1 := w_{b_{11}}]$. Given $\lambda x_0.t_{a_{11}} \simeq t_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t_{b_{11}} \longrightarrow^{x*} s_{b_{11}}$, $s_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{x*} s_{b_{12}}$, and $\lambda x_0.t_{a_{11}} \simeq s_{b_{12}}$. We proceed by cases on $(\lambda x_0.t_{a11}) \simeq s_{b12}$. The only case is (lam-sim), so let $s_{b_{12}} =$

$\lambda x_0.t_{b_{121}}$ where $t_{a_{11}} \simeq t_{b_{121}}$. We have $t_{b_1} \longrightarrow^{x*} \lambda x_0.t_{b_{121}}$ and $(\lambda x_0.t_{b_{121}})[x := w_{b_1}] \longrightarrow^{x} \lambda x_2.t_{b_{121}}[x_0 := x_2][x := w_{b_1}]$ where $x_2 \notin FV(\lambda x_0.t_{b_{121}}) \cup FV(w_{b_1}) \cup \{x\}$.

Let $x_3 \notin FV(\lambda x_0.t_{a_{11}}) \cup FV(w_{a_1}) \cup FV(\lambda x_0.t_{b_{121}}) \cup FV(w_{b_1}) \cup \{x\}$, then by the definition of α-equivalence, we get $\lambda x_1.t_{a_{11}}[x_1/x_0][w_{a_1}/x] \sim_\alpha \lambda x_3.t_{a_{11}}[x_3/x_0][w_{a_1}/x]$ and $\lambda x_2.t_{b_{121}}[x_0 := x_2][x := w_{b_1}] \sim_\alpha \lambda x_3.t_{b_{121}}[x_0 := x_3][x := w_{b_1}]$.

By (lam-sim) and (subst-sim), we get $\lambda x_3.t_{a_{11}}[x_3/x_0][w_{a_1}/x] \simeq \lambda x_3.t_{b_{121}}[x_0 := x_3][x := w_{b_1}]$. Hence we have $\lambda x_1.t_{a_{11}}[x_1/x_0][w_{a_1}/x] \simeq \lambda x_2.t_{b_{121}}[x_0 := x_2][x := w_{b_1}]$.

Case 5. ($t_{a_1} = n$). We have

$$\frac{n \simeq t_{b_1} \quad w_{a_1} \simeq w_{b_1}}{n[w_{a_1}/x] \simeq t_{b_1}[x := w_{b_1}].}$$

Then, $n[w_{a_1}/x] = n$. We proceed by cases on $n \simeq t_{b_1}$.

Case i. (num-sim). Let $t_{b_1} = n$. We have $n \longrightarrow^{x*} n$, $n[x := w_{b1}] \longrightarrow^{x} n$ and $n \simeq n$.

Case ii. (subst-sim). Let $t_{b_1} = t_{b_{11}}[x_1 := w_{b_{11}}]$. Given $n \simeq t_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t_{b_{11}} \longrightarrow^{x*} s_{b_{11}}$, $s_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{x*} s_{b_{12}}$, and $n \simeq s_{b_{12}}$. We proceed by cases on $n \simeq s_{b_{12}}$. The only case is (num-sim), so let $s_{b_{12}} = n$. We have $t_{b_1} \longrightarrow^{x*} n$, $n[x := w_{b_1}] \longrightarrow^{x*} n$ and by (num-sim) $n \simeq n$.

Case 6. ($t_{a_1} = t_{a_{11}} + t_{a_{12}}$). We have

$$\frac{t_{a_{11}} + t_{a_{12}} \simeq t_{b_1} \quad w_{a_1} \simeq w_{b_1}}{(t_{a_{11}} + t_{a_{12}})[w_{a_1}/x] \simeq t_{b_1}[x := w_{b_1}].}$$

Then, $(t_{a_{11}} + t_{a_{12}})[w_{a_1}/x] = t_{a_{11}}[w_{a_1}/x] + t_{a_{12}}[w_{a_1}/x]$. We proceed by cases on $t_{a_{11}} + t_{a_{12}} \simeq t_{b_1}$.

Case i. (plus-sim). Let $t_{b_1} = t_{b_{11}} + t_{b_{12}}$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $t_{a_{12}} \simeq t_{b_{12}}$. We have $t_{b_{11}} + t_{b_{12}} \longrightarrow^{x*} t_{b_{11}}\ t_{b_{12}}$ and $(t_{b_{11}} + t_{b_{12}})[x := w_{b_1}] \longrightarrow^{x} t_{b_{11}}[x := w_{b_1}] + t_{b_{12}}[x := w_{b_1}]$. By (subst-sim) and (plus-sim), we get $t_{a_{11}}[w_{a_1}/x] + t_{a_{12}}[w_{a_1}/x] \simeq t_{b_{11}}[x := w_{b_1}] + t_{b_{12}}[x := w_{b_1}]$.

Case ii. (subst-sim). Let $t_{b_1} = t_{b_{11}}[x_1 := w_{b_{11}}]$. Given $t_{a_{11}} + t_{a_{12}} \simeq t_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t_{b_{11}} \longrightarrow^{x*} s_{b_{11}}$, $s_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{x*} s_{b_{12}}$, and $t_{a_{11}} + t_{a_{12}} \simeq s_{b_{12}}$. We proceed by cases on $t_{a_{11}} + t_{a_{12}} \simeq s_{b_{12}}$. The only case is (plus-sim), so let $s_{b_{12}} = t_{b_{121}} + t_{b_{122}}$ where $t_{a_{11}} \simeq t_{b_{121}}$ and $t_{a_{12}} \simeq t_{b_{122}}$. Then, $t_{b_1} \longrightarrow^{x*} t_{b_{121}} + t_{b_{122}}$ and $(t_{b_{121}} + t_{b_{122}})[x := w_{b_1}] \longrightarrow^{x} t_{b_{121}}[x := w_{b_1}] + t_{b_{122}}[x := w_{b_1}]$. By (subst-sim) and (plus-sim), we get $t_{a_{11}}[w_{a_1}/x] + t_{a_{12}}[w_{a_1}/x] \simeq t_{b_{121}}[x := w_{b_1}] + t_{b_{122}}[x := w_{b_1}]$.

□

B.1.4 Canonisation

Given two related terms, if one term is a value, then the other term is a value or multi-steps to a value. We have the following two lemmas.

Lemma 214 (Canonisation of (Substitutional) ISWIM)**.** *If* $t_{a_1} \simeq v_{b_1}$*, then* $t_{a_1} \in \text{VALUE}_{\text{sub}}$.

Proof. We proceed by structural induction on $t_{a_1} \simeq v_{b_1}$.

Case 1. (var-sim). This case is vacuous.

Case 2. (app-sim). This case is vacuous.

Case 3. (lam-sim). Let $t_{a_1} = \lambda x.t_{a_{11}}$ and $v_{b_1} = \lambda x.t_{b_{11}}$ where $t_{a_{11}} \simeq t_{b_{11}}$. We have $\lambda x.t_{a_{11}} \in \text{VALUE}_{\text{sub}}$.

Case 4. (num-sim). Let $t_{a_1} = v_{b_1} = n$. We have $n \in \text{VALUE}_{\text{sub}}$.

Case 5. (plus-sim). This case is vacuous.

Case 6. (subst-sim). This case is vacuous.

□

Lemma 215 (Canonisation of Explicit ISWIM)**.** *If* $v_{a_1} \simeq t_{b_1}$*, then* $t_{b_1} \longrightarrow^* v_{b_2}$ *and* $v_{a_1} \simeq v_{b_2}$.

Proof. We proceed by structural induction on $v_{a_1} \simeq t_{b_1}$.

Case 1. (var-sim). This case is vacuous.

Case 2. (app-sim). This case is vacuous.

Case 3. (lam-sim). Let $v_{a_1} = \lambda x.t_{a_{11}}$ and $t_{b_1} = \lambda x.t_{b_{11}}$ where $t_{a_{11}} \simeq t_{b_{11}}$. We have $\lambda x.t_{b_{11}} \longrightarrow^* \lambda x.t_{b_{11}}$, $\lambda x.t_{b_{11}} \in \text{VALUE}_{\text{exp}}$ and $v_{a_1} \simeq \lambda x.t_{b_{11}}$.

Case 4. (num-sim). Let $v_{a_1} = t_{b_1} = n$. We have $n \longrightarrow^* n$, $n \in \text{VALUE}_{\text{exp}}$, and $v_{a_1} \simeq n$.

Case 5. (plus-sim). This case is vacuous.

Case 6. (subst-sim). Let $v_{a_1} = t_{a_{11}}[w_{a_1}/x]$ and $t_{b_1} = t_{b_{11}}[x := w_{b_1}]$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $w_{a_1} \simeq w_{b_1}$. By Lemma 213, we get $t_{b_{11}}[x := w_{b_1}] \longrightarrow^* s_{b_{12}}$ and $t_{a_{11}}[w_{a_1}/x] \simeq s_{b_{12}}$. We proceed by cases on $t_{a_{11}}[w_{a_1}/x] \simeq s_{b_{12}}$.

Case i. (var-sim). This case is vacuous.

Case ii. (app-sim). This case is vacuous.

Case iii. (lam-sim). Let $t_{a_{11}}[w_{a_1}/x] = \lambda x_1.t_{a_{111}}$ and $s_{b_{12}} = \lambda x_1.t_{b_{121}}$ where $t_{a_{111}} \simeq t_{b_{121}}$. We have $t_{b_{11}}[x := w_{b_1}] \longrightarrow^* \lambda x_1.t_{b_{121}}$, $\lambda x_1.t_{b121} \in \text{VALUE}_{\text{exp}}$ and $v_{a_1} \simeq \lambda x_1.t_{b_{121}}$.

Case iv. (num-sim). Let $t_{a_{11}}[w_{a_1}/x] = s_{b_{12}} = n$. We have $t_{b_{11}}[x := w_{b_1}] \longrightarrow^* n$, $n \in \text{VALUE}_{\text{exp}}$ and $v_{a_1} \simeq n$.

Case v. (plus-sim). This case is vacuous.

Case vi. (subst-sim). This case is vacuous.

□

B.1.5 Explicit Substitution Descendant Relation

We define the explicit substitution descendant relation and show its well-foundedness. As a result, we can do induction on explicit substitution descendants.

Definition 216 (Explicit Substitution Descendant Relation)**.** For any $t_1, t_2 \in \text{TERM}_{\text{exp}}$, $t_1 \prec^{\text{x}} t_2$ if and only if $t_2 \longrightarrow^{\text{x}} t_1$. We call $\prec^{\text{x}}$ the explicit substitution descendant relation.

Definition 217 (Weight Function)**.** For any $t \in \text{TERM}_{\text{exp}}$, its weight is $W(t)$ where W is a function defined as follows.

$$W \ : \ \text{TERM}_{\text{exp}} \longrightarrow \mathbb{Z}^+$$

$$\begin{array}{rcl} W(x) & = & 1 \\ W(t_1\ t_2) & = & W(t_1) + W(t_2) + 1 \\ W(\lambda x.t) & = & 1 \\ W(n) & = & 1 \\ W(t_1 + t_2) & = & W(t_1) + W(t_2) + 1 \\ W(t[x := w]) & = & W(t) \cdot (W(w) + 1) \end{array}$$

Lemma 218 (Substitution reduction decreases weight.)**.** *For any $t_1, t_2 \in \text{TERM}_{\text{exp}}$, if $t_1 \longrightarrow^{\text{x}} t_2$, then $W(t_2) < W(t_1)$.*

Proof. We proceed by structural induction on $t_1 \longrightarrow^{\text{x}} t_2$.

Case 1. (var-eq-subst). Let $t_1 = x[x := w]$ and $t_2 = w$. Then, $W(x[x := w]) = W(x) \cdot (W(w) + 1) = W(w) + 1$. We have $W(w) < W(w) + 1 = W(x[x := w])$.

Case 2. (var-dif-subst). Let $t_1 = x_1[x_2 := w]$ and $t_2 = x_1$ where $x_1 \not\equiv x_2$. Then, $W(x_1[x_2 := w]) = W(x_1) \cdot (W(w) + 1) = W(w) + 1$. We have $W(x_1) = 1 < W(w) + 1 = W(x_1[x_2 := w])$.

Case 3. (num-subst). Let $t_1 = n[x := w]$ and $t_2 = n$. Then, $W(n[x := w]) = W(n) \cdot (W(w) + 1) = W(w) + 1$. We have $W(n) = 1 < W(w) + 1 = W(n[x := w])$.

Case 4. (app-subst). Let $t_1 = (t_{11}\, t_{12})[x := w]$ and $t_2 = (t_{11}[x := w])\ (t_{12}[x := w])$. Then, $W(t_{11}[x := w]\ t_{12}[x := w]) = W(t_{11}[x := w]) + W(t_{12}[x := w]) + 1 = W(t_{11}) \cdot (W(w) + 1) + W(t_{12}) \cdot (W(w) + 1) + 1 = (W(t_{11}) + W(t_{12})) \cdot (W(w) + 1) + 1$ and $W((t_{11}\ t_{12})[x := w]) = W(t_{11}\ t_{12}) \cdot (W(w) +$

$1) = (W(t_{11}) + W(t_{12}) + 1) \cdot (W(w) + 1) = (W(t_{11}) + W(t_{12})) \cdot (W(w) + 1) + W(w) + 1$. We have $W((t_{11}[x := w])\ (t_{12}[x := w])) = (W(t_{11}) + W(t_{12})) \cdot (W(w) + 1) + 1 < (W(t_{11}) + W(t_{12})) \cdot (W(w) + 1) + W(w) + 1 = W((t_{11}\, t_{12})[x := w])$.

Case 5. (plus-subst). Let $t_1 = (t_{11} + t_{12})[x := w]$ and $t_2 = t_{11}[x := w] + t_2[x := w])$. Then, $W(t_{11}[x := w] + t_{12}[x := w]) = W(t_{11}[x := w]) + W(t_{12}[x := w]) + 1 = W(t_{11}) \cdot (W(w) + 1) + W(t_{12}) \cdot (W(w) + 1) + 1 = (W(t_{11}) + W(t_{12})) \cdot (W(w) + 1) + 1$ and $W((t_{11} + t_{12})[x := w]) = W(t_{11} + t_{12}) \cdot (W(w) + 1) = (W(t_{11}) + W(t_{12}) + 1) \cdot (W(w) + 1) = (W(t_{11}) + W(t_{12})) \cdot (W(w) + 1) + W(w) + 1$. We have $W(t_{11}[x := w] + t_{12}[x := w]) = (W(t_{11}) + W(t_{12})) \cdot (W(w) + 1) + 1 < (W(t_{11}) + W(t_{12})) \cdot (W(w) + 1) + W(w) + 1 = W((t_{11} + t_{12})[x := w])$.

Case 6. (lam-subst). Let $t_1 = (\lambda x_1.t)[x_2 := w]$ and $t_2 = \lambda x_3.t[x_1 := x_3][x_2 := w]$ where $x_3 \notin FV(\lambda x_1.t) \cup FV(w) \cup \{x_2\}$. Then, $W((\lambda x_1.t_{11})[x_2 := w]) = W(\lambda x_1.t_{11}) \cdot (W(w) + 1) = W(w) + 1$. We have $W(\lambda x_3.t_{11}[x_1 := x_3][x_2 := w]) = 1 < W(w) + 1 = W((\lambda x_1.t_{11})[x_2 := w])$.

Case 7. (subst-subst). Let $t_1 = t_{11}[x_1 := w_1][x_2 := w_2]$ and $t_2 = t_{21}[x_2 := w_2]$ where $t_{11}[x_1 := w_1] \longrightarrow^{\mathrm{x}} t_{21}$. By the induction hypothesis, $W(t_{21}) < W(t_{11}[x_1 := w_1])$. Then, $W(t_{21}[x_2 := w_2]) = W(t_{21}) \cdot (W(w_2) + 1)$ and $W(t_{11}[x_1 := w_1][x_2 := w_2]) = W(t_{11}[x_1 := w_1]) \cdot (W(w_2) + 1)$. We have $W(t_{21}[x_2 := w_2]) = W(t_{21}) \cdot (W(w_2) + 1) = W(t_{21}) \cdot (W(w_2) + 1) < W(t_{11}[x_1 := w_1]) \cdot (W(w_2) + 1) = W(t_{11}[x_1 := w_1][x_2 := w_2])$.

□

Lemma 219 (Well-foundedness of Explicit Substitution Descendant Relation)**.** *The explicit substitution descendant relation $\prec^{\mathrm{x}}$ is well-founded.*

Proof. Lemma 218 has proved that if $t_1 \longrightarrow^{\mathrm{x}} t_2$, then $W(t_2) < W(t_1)$, for any $t_1.t_2 \in \mathrm{TERM}_{\mathrm{exp}}$. For any $t \in \mathrm{TERM}_{\mathrm{exp}}$, the length of the descending chain with respect to $\prec^{\mathrm{x}}$ starting from t is bound by $W(t)$. Hence, the explicit substitution descendant relation $\prec^{\mathrm{x}}$ is well-founded. □

B.1.6 Bisimulation

We demonstrate (Substitutional) ISWIM bisimulates Explicit ISWIM. Intuitively, given two related terms, if one term single-steps, then the other term multi-steps, and the resulting two terms are related.

Lemma 220 (Simulation: Explicit ISWIM simulates (Substitutional) ISWIM.)**.** *If $t_{a_1} \simeq t_{b_1}$ and $t_{a_1} \longrightarrow t_{a_2}$, then $t_{b_1} \longrightarrow^* t_{b_2}$ and $t_{a_2} \simeq t_{b_2}$.*

Proof. We proceed by induction on the structure of $t_{a_1} \simeq t_{b_1}$ and by induction on the explicit substitution descendants $\prec^{\mathrm{x}} t_{b_1}$ simultaneously.

Case 1. (var-sim). This case is vacuous.

Case 2. (app-sim). Let $t_{a_1} = t_{a_{11}}\, t_{a_{12}}$ and $t_{b_1} = t_{b_{11}}\, t_{b_{12}}$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $t_{a_{12}} \simeq t_{b_{12}}$. We proceed by cases on $t_{a_1} \longrightarrow t_{a_2}$.

Case i. (appL). Let $t_{a_{11}} \longrightarrow t_{a_{21}}$ and $t_{a_2} = t_{a_{21}}\ t_{a_{12}}$. By the induction hypothesis, $t_{b_{11}} \longrightarrow^* t_{b_{21}}$ and $t_{a_{21}} \simeq t_{b_{21}}$. Then, $t_{b_{11}}\ t_{b_{12}} \longrightarrow^* t_{b_{21}}\ t_{b_{12}}$ and by (app-sim) $t_{a_{21}}\ t_{a_{12}} \simeq t_{b_{21}}\ t_{b_{12}}$.

Case ii. (appR). Let $t_{a_{11}} = v_{a_{11}}, t_{a_{12}} \longrightarrow t_{a_{22}}$ and $t_{a_2} = v_{a_{11}}\ t_{a_{22}}$. Given $v_{a_{11}} \simeq t_{b_{11}}$, by Lemma 215, $t_{b_{11}} \longrightarrow^* v_{b_{11}}$ and $v_{a_{11}} \simeq v_{b_{11}}$. By the induction hypothesis, $t_{b_{12}} \longrightarrow^* t_{b_{22}}$ and $t_{a_{22}} \simeq t_{b_{22}}$. Then, $t_{b_{11}}\ t_{b_{12}} \longrightarrow^* v_{b_{11}}\ t_{b_{12}} \longrightarrow^* v_{b_{11}}\ t_{b_{22}}$ and by (app-sim) $v_{a_{11}}\ t_{a_{22}} \simeq v_{b_{11}}\ t_{b_{22}}$.

Case iii. (app). Let $t_{a_{11}} = \lambda x.t_{a_{111}}$, $t_{a_{12}} = v_{a_{12}}$, and $t_{a_2} = t_{a_{111}}[v_{a_{12}}/x]$. Given $\lambda x.t_{a_{111}} \simeq t_{b_{11}}$, by Lemma 215, $t_{b_{11}} \longrightarrow^* v_{b_{11}}$ and $\lambda x.t_{a_{111}} \simeq v_{b_{11}}$. Given $v_{a_{12}} \simeq t_{b_{12}}$, by Lemma 215, $t_{b_{12}} \longrightarrow^* v_{b_{12}}$ and $v_{a_{12}} \simeq v_{b_{12}}$. We proceed by cases on $\lambda x.t_{a_{111}} \simeq v_{b_{11}}$. The only case is (lam-sim), so let $v_{b_{11}} = \lambda x.t_{b_{111}}$ and $t_{a_{111}} \simeq t_{b_{111}}$. Then, $t_{b_{11}}\ t_{b_{12}} \longrightarrow^* (\lambda x.t_{b_{111}})\ t_{b_{12}} \longrightarrow^* (\lambda x.t_{b_{111}})\ v_{b_{12}} \longrightarrow t_{b_{111}}[x := v_{b_{12}}]$. By (subst-sim), we get $t_{a_{111}}[v_{a_{12}}/x] \simeq t_{b_{111}}[x := v_{b_{12}}]$.

Case 3. (lam-sim). This case is vacuous.

Case 4. (num-sim). This case is vacuous.

Case 5. (plus-sim). Let $t_{a_1} = t_{a_{11}} + t_{a_{12}}$ and $t_{b_1} = t_{b_{11}} + t_{b_{12}}$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $t_{a_{12}} \simeq t_{b_{12}}$. We proceed by cases on $t_{a_1} \longrightarrow t_{a_2}$.

Case i. (plusL). Let $t_{a_{11}} \longrightarrow t_{a_{21}}$ and $t_{a_2} = t_{a_{21}} + t_{a_{12}}$. By the induction hypothesis, $t_{b_{11}} \longrightarrow^* t_{b_{21}}$ and $t_{a_{21}} \simeq t_{b_{21}}$. Then, $t_{b_{11}} + t_{b_{12}} \longrightarrow^* t_{b_{21}} + t_{b_{12}}$ and by (plus-sim) $t_{a_{21}} + t_{a_{12}} \simeq t_{b_{21}} + t_{b_{12}}$.

Case ii. (plusR). Let $t_{a_{11}} = v_{a_{11}}, t_{a_{12}} \longrightarrow t_{a_{22}}$ and $t_{a_2} = v_{a_{11}} + t_{a_{22}}$. Given $v_{a_{11}} \simeq t_{b_{11}}$, by Lemma 215, $t_{b_{11}} \longrightarrow^* v_{b_{11}}$ and $v_{a_{11}} \simeq v_{b_{11}}$. By the induction hypothesis, $t_{b_{12}} \longrightarrow^* t_{b_{22}}$ and $t_{a_{22}} \simeq t_{b_{22}}$. Then, $t_{b_{11}} + t_{b_{12}} \longrightarrow^* v_{b_{11}} + t_{b_{12}} \longrightarrow^* v_{b_{11}} + t_{b_{22}}$ and by (plus-sim) $v_{a_{11}} + t_{a_{22}} \simeq v_{b_{11}} + t_{b_{22}}$.

Case iii. (plus). Let $t_{a_{11}} = n_1$, $t_{a_{12}} = n_2$, and $t_{a_2} = n$ where $n = n_1 + n_2$. Given $n_1 \simeq t_{b_{11}}$, by Lemma 215, $t_{b_{11}} \longrightarrow^* v_{b_{11}}$ and $n_1 \simeq v_{b_{11}}$. Given $n_2 \simeq t_{b_{12}}$, by Lemma 215, $t_{b_{12}} \longrightarrow^* v_{b_{12}}$ and $n_2 \simeq v_{b_{12}}$. We proceed by cases on $n_1 \simeq v_{b_{11}}$. The only case is (num-sim), so let $v_{b_{11}} = n_1$. We proceed by cases on $n_2 \simeq v_{b_{12}}$. The only case is (num-sim), so let $v_{b_{12}} = n_2$. Then, $t_{b_{11}} + t_{b_{12}} \longrightarrow^* n_1 + t_{b_{12}} \longrightarrow^* n_1 + n_2 \longrightarrow n$ where $n = n_1 + n_2$. By (num-sim), we get $n \simeq n$.

Case 6. (subst-sim). Let $t_{a_1} = t_{a_{11}}[w_{a_{11}}/x]$ and $t_{b_1} = t_{b_{11}}[x := w_{b_{11}}]$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $w_{a_{11}} \simeq w_{b_{11}}$. Given $t_{a_1} \simeq t_{b_{11}}[x := w_{b_{11}}]$, by Lemma 213, $t_{b_{11}}[x := w_{b_{11}}] \longrightarrow^{\mathrm{x}*} s_{b_{21}}$ and $t_{a_1} \simeq s_{b_{21}}$. Then, $s_{b_{21}} \prec^{\mathrm{x}} t_{b_1}$. If $t_{a_1} \longrightarrow t_{a_2}$, by the induction hypothesis, $s_{b_{21}} \longrightarrow^* t_{b_2}$ and $t_{a_2} \simeq t_{b_2}$. We have $t_{b_1} \longrightarrow^* s_{b_{21}} \longrightarrow^* t_{b_2}$.

□

Remark 221. In the last case of the proof, given $t_{a_1} \simeq t_{b_1}$, $t_{a_1} \simeq s_{b_{21}}$ and $s_{b_{21}} \prec^{\mathrm{x}} t_{b_1}$, if $t_{a_1} \longrightarrow t_{a_2}$, by the induction hypothesis, $s_{b_{21}} \longrightarrow^* t_{b_2}$ and $t_{a_2} \simeq t_{b_2}$.

Lemma 222 (Single-step explicit substitution reduction preserves simulation relation.)**.** *If* $t_{a_1} \simeq t_{b_1}$ *and* $t_{b_1} \longrightarrow^{\mathrm{x}} t_{b_2}$*, then* $t_{a_1} \simeq t_{b_2}$.

Proof. We proceed by structural induction on $t_{a_1} \simeq t_{b_1}$. Since $t_{b_1} \longrightarrow^{\mathrm{x}} t_{b_2}$, only (subst-sim) applies. Let $t_{a_1} = t_{a_{11}}[w_{a_{11}}/x_1]$ and $t_{b_1} = t_{b_{11}}[x_1 := w_{b_{11}}]$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $w_{a_{11}} \simeq w_{b_{11}}$. We proceed by cases on $t_{b_1} \longrightarrow^{\mathrm{x}} t_{b_2}$.

Case 1. (var-eq-subst). Let $t_{b_{11}} = x_1$. We have $x_1[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}} w_{b_{11}}$. We proceed by cases on $t_{a_{11}} \simeq x_1$. The only case is (var-sim), thus we get $t_{a_{11}} = x_1$. Then, $x_1[w_{a_{11}}/x_1] = w_{a_{11}}$ and $w_{a_{11}} \simeq w_{b_{11}}$.

Case 2. (var-dif-subst). Let $t_{b_{11}} = x_2$ and $x_2 \not\equiv x_1$. We have $x_2[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}} x_2$. We proceed by cases on $t_{a_{11}} \simeq x_2$. The only case is (var-sim), thus we get $t_{a_{11}} = x_2$. Then, $x_2[w_{a_{11}}/x_1] = x_2$ and $x_2 \simeq x_2$.

Case 3. (num-subst). Let $t_{b_{11}} = n$. We have $n[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}} n$. We proceed by cases on $t_{a_{11}} \simeq n$. The only case is (num-sim), thus we get $t_{a_{11}} = n$. Then, $n[w_{a_{11}}/x_1] = n$ and $n \simeq n$.

Case 4. (app-subst). Let $t_{b_{11}} = t_{b_{111}}\ t_{b_{112}}$. We have $(t_{b_{111}}\ t_{b_{112}})[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}} (t_{b_{111}}[x_1 := w_{b_{11}}])\ (t_{b_{112}}[x_1 := w_{b_{11}}])$. We proceed by cases on $t_{a_{11}} \simeq t_{b_{111}}\ t_{b_{112}}$. The only case is (app-sim), thus we get $t_{a_{11}} = t_{a_{111}}\ t_{a_{112}}$, $t_{a_{111}} \simeq t_{b_{111}}$ and $t_{a_{112}} \simeq t_{b_{112}}$. Then, $(t_{a_{111}}\ t_{a_{112}})[w_{a_{11}}/x_1] = (t_{a_{111}}[w_{a_{11}}/x_1])\ (t_{a_{112}}[w_{a_{11}}/x_1])$ and by (subst-sim) and (app-sim) $(t_{a_{111}}[w_{a_{11}}/x_1])\ (t_{a_{112}}[w_{a_{11}}/x_1]) \simeq (t_{b_{111}}[x_1 := w_{b_{11}}])\ (t_{b_{112}}[x_1 := w_{b_{11}}])$.

Case 5. (plus-subst). Let $t_{b_{11}} = t_{b_{111}} + t_{b_{112}}$. We have $(t_{b_{111}} + t_{b_{112}})[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}} t_{b_{111}}[x_1 := w_{b_{11}}] + t_{b_{112}}[x_1 := w_{b_{11}}]$. We proceed by cases on $t_{a_{11}} \simeq t_{b_{111}} + t_{b_{112}}$. The only case is (plus-sim), thus we get $t_{a_{11}} = t_{a_{111}} + t_{a_{112}}$, $t_{a_{111}} \simeq t_{b_{111}}$ and $t_{a_{112}} \simeq t_{b_{112}}$. Then, $(t_{a_{111}} + t_{a_{112}})[w_{a_{11}}/x_1] = t_{a_{111}}[w_{a_{11}}/x_1] + t_{a_{112}}[w_{a_{11}}/x_1]$ and by (subst-sim) and (plus-sim) $t_{a_{111}}[w_{a_{11}}/x_1] + t_{a_{112}}[w_{a_{11}}/x_1] \simeq t_{b_{111}}[x_1 := w_{b_{11}}] + t_{b_{112}}[x_1 := w_{b_{11}}]$.

Case 6. (lam-subst). Let $t_{b_{11}} = \lambda x_2.t_{b_{111}}$. We have $(\lambda x_2.t_{b_{111}})[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}} \lambda x_3.t_{b_{111}}[x_2 := x_3][x_1 := w_{b_{11}}]$ where $x_3 \notin FV(\lambda x_2.t_{b_{111}}) \cup FV(w_{b_{11}}) \cup \{x_1\}$. We proceed by cases on $t_{a_{11}} \simeq \lambda x_2.t_{b_{111}}$. The only case is (lam-sim), thus we get $t_{a_{11}} = \lambda x_2.t_{a_{111}}$ where $t_{a_{111}} \simeq t_{b_{111}}$. Then, $(\lambda x_2.t_{a_{111}})[w_{a_{11}}/x_1] = \lambda x_4.t_{a_{111}}[x_4/x_2][w_{a_{11}}/x_1]$ where $x_4 \notin FV(\lambda x_2.t_{a_{111}}) \cup FV(w_{a_{11}}) \cup \{x_1\}$.
Let $x_5 \notin FV(\lambda x_2.t_{b_{111}}) \cup FV(w_{b_{11}}) \cup FV(\lambda x_2.t_{a_{111}}) \cup FV(w_{a_{11}}) \cup \{x_1\}$, we have $\lambda x_3.t_{b_{111}}[x_2 := x_3][x_1 := w_{b_{11}}] \sim_\alpha \lambda x_5.t_{b_{111}}[x_2 := x_5][x_1 := w_{b_{11}}]$ and $\lambda x_4.t_{a_{111}}[x_4/x_2][w_{a_{11}}/x_1] \sim_\alpha \lambda x_5.t_{a_{111}}[x_5/x_2][w_{a_{11}}/x_1]$.
By (subst-sim) and (lam-sim), we get $\lambda x_5.t_{a_{111}}[x_5/x_2][w_{a_{11}}/x_1] \simeq \lambda x_5.t_{b_{111}}[x_2 := x_5][x_1 := w_{b_{11}}]$.
Hence $\lambda x_4.t_{a_{111}}[x_4/x_2][w_{a_{11}}/x_1] \simeq \lambda x_3.t_{b_{111}}[x_2 := x_3][x_1 := w_{b_{11}}]$.

Case 7. (subst-subst). Let $t_{b_{11}} = t_{b_{111}}[x_2 := w_{b_{12}}]$. We have $(t_{b_{111}}[x_2 := w_{b_{12}}])[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}} t_{b_{121}}[x_1 := w_{b_{11}}]$ where $t_{b_{111}}[x_2 := w_{b_{12}}] \longrightarrow^{\mathrm{x}} t_{b_{121}}$. By the induction hypothesis, $t_{a_{11}} \simeq t_{b_{121}}$. Then, by (subst-sim) $t_{a_{11}}[w_{a_{11}}/x_1] \simeq t_{b_{121}}[x_1 := w_{b_{11}}]$.

□

Lemma 223 (Simulation: (Substitutional) ISWIM simulates Explicit ISWIM.)**.** *If $t_{a_1} \simeq t_{b_1}$ and $t_{b_1} \longrightarrow t_{b_2}$, then $t_{a_1} \longrightarrow^* t_{a_2}$ and $t_{a_2} \simeq t_{b_2}$.*

Proof. We proceed by induction on the structure of $t_{a_1} \simeq t_{b_1}$.

Case 1. (var-sim). This case is vacuous.

Case 2. (app-sim). Let $t_{a_1} = t_{a_{11}}\ t_{a_{12}}$ and $t_{b_1} = t_{b_{11}}\ t_{b_{12}}$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $t_{a_{12}} \simeq t_{b_{12}}$. We proceed by cases on $t_{b_1} \longrightarrow t_{b_2}$.

Case i. (appL). Let $t_{b_{11}} \longrightarrow t_{b_{21}}$ and $t_{b_2} = t_{b_{21}}\ t_{b_{12}}$. By the induction hypothesis, $t_{a_{11}} \longrightarrow^* t_{a_{21}}$ and $t_{a_{21}} \simeq t_{b_{21}}$. Then, $t_{a_{11}}\ t_{a_{12}} \longrightarrow^* t_{a_{21}}\ t_{a_{12}}$ and by (app-sim) $t_{a_{21}}\ t_{a_{12}} \simeq t_{b_{21}}\ t_{b_{12}}$.

Case ii. (appR). Let $t_{b_{11}} = v_{b_{11}}$, $t_{b_{12}} \longrightarrow t_{b_{22}}$ and $t_{b_2} = v_{b_{11}}\ t_{b_{22}}$. Given $t_{a_{11}} \simeq v_{b_{11}}$, by Lemma 214, $t_{a_{11}} \longrightarrow^* v_{a_{11}}$ and $v_{a_{11}} \simeq v_{b_{11}}$. By the induction hypothesis, $t_{a_{12}} \longrightarrow^* t_{a_{22}}$ and $t_{a_{22}} \simeq t_{b_{22}}$. Then, $t_{a_{11}}\ t_{a_{12}} \longrightarrow^* v_{a_{11}}\ t_{a_{12}} \longrightarrow^* v_{a_{11}}\ t_{a_{22}}$ and by (app-sim) $v_{a_{11}}\ t_{a_{22}} \simeq v_{b_{11}}\ t_{b_{22}}$.

Case iii. (app). Let $t_{b_{11}} = \lambda x.t_{b_{111}}$, $t_{b_{12}} = v_{b_{12}}$, and $t_{b_2} = t_{b_{111}}[x := v_{b_{12}}]$. Given $t_{a_{11}} \simeq \lambda x.t_{b_{111}}$, by Lemma 214, $t_{a_{11}} \longrightarrow^* v_{a_{11}}$ and $v_{a_{11}} \simeq \lambda x.t_{b_{111}}$. Given $t_{a_{12}} \simeq v_{b_{12}}$, by Lemma 214, $t_{a_{12}} \longrightarrow^* v_{a_{12}}$ and $v_{a_{12}} \simeq v_{b_{12}}$. We proceed by cases on $v_{a_{11}} \simeq \lambda x.t_{b_{111}}$. The only case is (lam-sim), so let $v_{a_{11}} = \lambda x.t_{a_{111}}$ and $t_{a_{111}} \simeq t_{b_{111}}$. Then, $t_{a_{11}}\ t_{a_{12}} \longrightarrow^* (\lambda x.t_{a_{111}})\ t_{a_{12}} \longrightarrow^* (\lambda x.t_{a_{111}})\ v_{a_{12}} \longrightarrow t_{a_{111}}[v_{a_{12}}/x]$. By (subst-sim), we get $t_{a_{111}}[v_{a_{12}}/x] \simeq t_{b_{111}}[x := v_{b_{12}}]$.

Case 3. (lam-sim). This case is vacuous.

Case 4. (num-sim). This case is vacuous.

Case 5. (plus-sim). Let $t_{a_1} = t_{a_{11}} + t_{a_{12}}$ and $t_{b_1} = t_{b_{11}} + t_{b_{12}}$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $t_{a_{12}} \simeq t_{b_{12}}$. We proceed by cases on $t_{b_1} \longrightarrow t_{b_2}$.

Case i. (plusL). Let $t_{b_{11}} \longrightarrow t_{b_{21}}$ and $t_{b_2} = t_{b_{21}} + t_{b_{12}}$. By the induction hypothesis, $t_{a_{11}} \longrightarrow^* t_{a_{21}}$ and $t_{a_{21}} \simeq t_{b_{21}}$. Then, $t_{a_{11}} + t_{a_{12}} \longrightarrow^* t_{a_{21}} + t_{a_{12}}$ and by (plus-sim) $t_{a_{21}} + t_{a_{12}} \simeq t_{b_{21}} + t_{b_{12}}$.

Case ii. (plusR). Let $t_{b_{11}} = v_{b_{11}}$, $t_{b_{12}} \longrightarrow t_{b_{22}}$ and $t_{b_2} = v_{b_{11}} + t_{b_{22}}$. Given $t_{a_{11}} \simeq v_{b_{11}}$, by Lemma 214, $t_{a_{11}} \longrightarrow^* v_{a_{11}}$ and $v_{a_{11}} \simeq v_{b_{11}}$. By the induction hypothesis, $t_{a_{12}} \longrightarrow^* t_{a_{22}}$ and $t_{a_{22}} \simeq t_{b_{22}}$. Then, $t_{a_{11}} + t_{a_{12}} \longrightarrow^* v_{a_{11}} + t_{a_{12}} \longrightarrow^* v_{a_{11}} + t_{a_{22}}$ and by (plus-sim) $v_{a_{11}} + t_{a_{22}} \simeq v_{b_{11}} + t_{b_{22}}$.

Case iii. (plus). Let $t_{b_{11}} = n_1$, $t_{b_{12}} = n_2$, and $t_{b_2} = n$ where $n = n_1 + n_2$. Given $t_{a_{11}} \simeq n_1$, by Lemma 214, $t_{a_{11}} \longrightarrow^* v_{a_{11}}$ and $v_{a_{11}} \simeq n_1$. Given $t_{a_{12}} \simeq n_2$, by Lemma 214, $t_{a_{12}} \longrightarrow^* v_{a_{12}}$ and $v_{a_{12}} \simeq n_2$. We proceed by cases on $v_{a_{11}} \simeq n_1$. The only case is (num-sim), so let $v_{a_{11}} = n_1$. We proceed by cases on $v_{a_{12}} \simeq n_2$. The only case is (num-sim), so let $v_{a_{12}} = n_2$. Then, $t_{a_{11}} + t_{a_{12}} \longrightarrow^* n_1 + t_{a_{12}} \longrightarrow^* n_1 + n_2 \longrightarrow n$ where $n = n_1 + n_2$. By (num-sim), we get $n \simeq n$.

Case 6. (subst-sim). Let $t_{a_1} = t_{a_{11}}[w_{a_{11}}/x]$ and $t_{b_1} = t_{b_{11}}[x := w_{b_{11}}]$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $w_{a_{11}} \simeq w_{b_{11}}$. Given $t_{a_1} \simeq t_{b_{11}}[x := w_{b_{11}}]$, by Lemma 213, $t_{b_{11}}[x := w_{b_{11}}] \longrightarrow^{x*} s_{b_{21}}$ and $t_{a_1} \simeq s_{b_{21}}$. Since $t_{b_{11}}[x := w_{b_{11}}]$ is not in the substitution normal form but $s_{b_{21}}$ is in substitution normal form, we have $t_{b_{11}}[x := w_{b_{11}}] \longrightarrow^{x} t_{b_{21}} \longrightarrow^{x*} s_{b_{21}}$. By Lemma 222, $t_{a_1} \simeq t_{b_{21}}$. We also have $t_{a_1} \longrightarrow^* t_{a_1}$.

□

B.1.7 Soundness and Completeness

We demonstrate the soundness and completeness of Explicit ISWIM with respect to (Substitutional) ISWIM.

Theorem 224 (Soundness of Explicit ISWIM w.r.t. (Substitutional) ISWIM)**.** *If $t_{a_1} \simeq t_{b_1}$ and $t_{a_1} \longrightarrow^* v_{a_2}$ in (Substitutional) ISWIM, then $t_{b_1} \longrightarrow^* v_{b_2}$ in Explicit ISWIM and $v_{a_2} \simeq v_{b_2}$.*

Proof. We proceed by induction on the length of $t_{a_1} \longrightarrow^* v_{a_2}$.

Case 1. (0). Let $t_{a_1} = v_{a_2}$. By Lemma 215, $t_{b_1} \longrightarrow^* v_{b_2}$ and $v_{a_2} \simeq v_{b_2}$.

Case 2. $(n+1)$. Let $t_{a_1} \longrightarrow t_{a_2} \longrightarrow^{(n)} v_{a_2}$. Given $t_{a_1} \simeq t_{b_1}$ and $t_{a_1} \longrightarrow t_{a_2}$, by Lemma 220, $t_{b_1} \longrightarrow^* t_{b_2}$ and $t_{a_2} \simeq t_{b_2}$. Given $t_{a_2} \simeq t_{b_2}$ and $t_{a_2} \longrightarrow^{(n)} v_{a_2}$, by the induction hypothesis, $t_{b_2} \longrightarrow^* v_{b_2}$ and $v_{a_2} \simeq v_{b_2}$. We have $t_{b_1} \longrightarrow^* t_{b_2} \longrightarrow^* v_{b_2}$ and $v_{a_2} \simeq v_{b_2}$.

□

Remark 225. The parenthesised superscripted number n in $\longrightarrow^{(n)}$ denotes the number of single step is n.

Theorem 226 (Completeness of Explicit ISWIM w.r.t. (Substitutional) ISWIM)**.** *If $t_{a_1} \simeq t_{b_1}$ and $t_{b_1} \longrightarrow^* v_{b_2}$ in Explicit ISWIM, then $t_{a_1} \longrightarrow^* v_{a_2}$ in (Substitutional) ISWIM and $v_{a_2} \simeq v_{b_2}$.*

Proof. We proceed by induction on the length of $t_{b_1} \longrightarrow^* v_{b_2}$.

Case 1. (0). Let $t_{b_1} = v_{b_2}$. By Lemma 214, $t_{a_1} \in \text{VALUE}_{\text{sub}}$. Let $v_{a_2} = t_{a_1}$. We have $t_{a_1} \longrightarrow^* v_{a_2}$ and $v_{a_2} \simeq v_{b_2}$.

Case 2. $(n+1)$. Let $t_{b_1} \longrightarrow t_{b_2} \longrightarrow^{(n)} v_{b_2}$. Given $t_{a_1} \simeq t_{b_1}$ and $t_{b_1} \longrightarrow t_{b_2}$, by Lemma 223, $t_{a_1} \longrightarrow^* t_{a_2}$ and $t_{a_2} \simeq t_{b_2}$. Given $t_{a_2} \simeq t_{b_2}$ and $t_{b_2} \longrightarrow^{(n)} v_{b_2}$, by the induction hypothesis, $t_{a_2} \longrightarrow^* v_{a_2}$ and $v_{a_2} \simeq v_{b_2}$. We have $t_{a_1} \longrightarrow^* v_{a_2}$ and $v_{a_2} \simeq v_{b_2}$.

□

B.1.7.1 An alternative proof.

We demonstrate a different proof of Theorem 226 which does not use Lemma 223. We start with two lemmas. Their proofs are omitted.

Lemma 227. *If $t_{a_1} \simeq t_{b_1}$, $t_{b_1} \longrightarrow t_{b_2}$ and $t_{a_1} \not\simeq t_{b_2}$, then $t_{a_1} \longrightarrow t_{a_2}$.*

Remark 228. Lemma 227 does not imply whether or not $t_{a_2} \simeq t_{b_2}$.

Lemma 229. *If $t_{a_1} \simeq t_{b_1}$ and $t_{a_1} \longrightarrow t_{a_2}$, then $t_{b_1} \longrightarrow^+ t_{b_2}$ and $t_{a_2} \simeq t_{b_2}$.*

Remark 230. Lemma 229 is stronger than Lemma 220. In other words, Lemma 229 implies Lemma 220.

We restate and prove Theorem 226 as follows.

Theorem 231 (Completeness of Explicit ISWIM w.r.t. (Substitutional) ISWIM)**.** *If $t_{a_1} \simeq t_{b_1}$ and $t_{b_1} \longrightarrow^* v_{b_2}$ in Explicit ISWIM, then $t_{a_1} \longrightarrow^* v_{a_2}$ in (Substitutional) ISWIM and $v_{a_2} \simeq v_{b_2}$.*

Proof. We proceed by induction on the length of $t_{b_1} \longrightarrow^* v_{b_2}$.

Case 1. (0). Let $t_{b_1} = v_{b_2}$. By Lemma 214, $t_{a_1} \in \text{VALUE}_{\text{sub}}$. Let $v_{a_2} = t_{a_1}$. We have $t_{a_1} \longrightarrow^* v_{a_2}$ and $v_{a_2} \simeq v_{b_2}$.

Case 2. $(n+1)$. Let $t_{b_1} \longrightarrow t_{b_2} \longrightarrow^{(n)} v_{b_2}$. We proceed by cases on $t_{a_1} \simeq t_{b_1}$, in particular on whether it is (subst-sim) or not.

Case i. (subst-sim). Let $t_{a_1} = t_{a_{11}}[w_{a_1}/x]$ and $t_{b_1} = t_{b_{11}}[x := w_{b_1}]$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $w_{a_1} \simeq w_{b_1}$. By Lemma 213, $t_{b_1} \longrightarrow^{\text{x}*} s_{b_2}$. Observe that t_{b_1} is not in substitution normal form but s_{b_2} is in substitution normal form. By the determinism of the small-step semantics, $t_{b_1} \longrightarrow^{\text{x}} t_{b_2} \longrightarrow^{\text{x}(p)} s_{b_2} \longrightarrow^{(q)} v_{b_2}$ and $p+q=n$ where $p,q \geq 0$. By Lemma 222, $t_{a_1} \simeq t_{b_2}$. By the induction hypothesis, $t_{a_1} \longrightarrow^* v_{a_2}$ and $v_{a_2} \simeq v_{b_2}$.

Case ii. (other cases). We proceed by cases on whether $t_{a_1} \simeq t_{b_2}$.

Case a. $(t_{a_1} \simeq t_{b_2})$. Then, by the induction hypothesis, $t_{a_1} \longrightarrow^* v_{a_2}$ and $v_{a_2} \simeq v_{b_2}$.

Case b. $(t_{a_1} \not\simeq t_{b_2})$. By Lemma 227, $t_{a_1} \longrightarrow t_{a_2}$. By Lemma 229, $t_{b_1} \longrightarrow^+ t_{b_3}$ and $t_{a_2} \simeq t_{b_3}$. By the determinism of the small-step semantics, $t_{b_1} \longrightarrow t_{b_2} \longrightarrow^{(p)} t_{b_3} \longrightarrow^{(q)} v_{b_2}$ and $p+q=n$ where $p,q \geq 0$. By the induction hypothesis, $t_{a_2} \longrightarrow^* v_{a_2}$ and $v_{a_2} \simeq v_{b_2}$.

□

B.1.8 Kleene Equality of Evaluators

We prove the Kleene equality of evaluators $eval_{\text{ISWIM:SubSOS}}(t)$ and $eval_{\text{ISWIM:ExpSOS}}(t)$.

Theorem 232 (Kleene Equality of Evaluators)**.** *For any $t \in \text{PRGM}_{\text{ISWIM}}$, $eval_{\text{ISWIM:SubSOS}}(t)$ is Kleene equal to $eval_{\text{ISWIM:ExpSOS}}(t)$.*

Proof. We first show if $eval_{\text{ISWIM:SubSOS}}(t) = a$ where $a \in \text{ANS}_{\text{ISWIM}}$, then $eval_{\text{ISWIM:ExpSOS}}(t) = a$.

Case 1. If $eval_{\text{ISWIM:SubSOS}}(t) = \texttt{function}$, then $t \longrightarrow^* \lambda x.t'$. By Theorem 224, $t \longrightarrow^* v'$ and $\lambda x.t' \sim v'$. Proceed by induction on $\lambda x.t' \sim v'$. The only case is (lam-sim). Then $v' = \lambda x.t''$ and $t' \sim t''$. We have $eval_{\text{ISWIM:ExpSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{ISWIM:SubSOS}}(t) = n$, then $t \longrightarrow^* n$. By Theorem 224, $t \longrightarrow^* v'$ and $n \sim v'$. Proceed by induction on $n \sim v'$. The only case is (num-sim). Then $v' = n$. We have $eval_{\text{ISWIM:ExpSOS}}(t) = n$.

We then show if $eval_{\text{ISWIM:ExpSOS}}(t) = a$ where $a \in \text{ANS}_{\text{ISWIM}}$, then $eval_{\text{ISWIM:SubSOS}}(t) = a$.

Case 1. If $eval_{\text{ISWIM:ExpSOS}}(t) = \texttt{function}$, then $t \longrightarrow^* \lambda x.t'$. By Theorem 226, $t \longrightarrow^* v'$ and $v' \sim \lambda x.t'$. Proceed by induction on $v' \sim \lambda x.t'$. The only case is (lam-sim). Then $v' = \lambda x.t''$ and $t'' \sim t'$. We have $eval_{\text{ISWIM:SubSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{ISWIM:ExpSOS}}(t) = n$, then $t \longrightarrow^* n$. By Theorem 226, $t \longrightarrow^* v'$ and $v' \sim n$. Proceed by induction on $v' \sim n$. The only case is (num-sim). Then $v' = n$. We have $eval_{\text{ISWIM:SubSOS}}(t) = n$.

We observe that $eval_{\text{ISWIM:SubSOS}}(t)$ is undefined if and only if $eval_{\text{ISWIM:ExpSOS}}(t)$ is undefined. Therefore, $eval_{\text{ISWIM:SubSOS}}(t)$ is Kleene equal to $eval_{\text{ISWIM:ExpSOS}}(t)$. □

B.2 Equivalence of ISWIM and Suspended ISWIM

We demonstrate the equivalence of the substitutional structural operational semantics of ISWIM and the structural operational semantics of Suspended ISWIM. We use subscripts "$_{\text{sub}}$" and "$_{\text{sus}}$" to differentiate the syntax of Substitutional ISWIM from the syntax of Suspended ISWIM.

B.2.1 Simulation Relation

We first introduce a bisimulation relation that relates (Substitutional) ISWIM terms to Suspended ISWIM terms.

Definition 233 (Bisimulation Relation)**.** Define the bisimulation relation $\simeq$ to be a binary relation up to alpha equivalence between the set of terms in (Substitutional) ISWIM and the set of terms in Suspended ISWIM.

$$\simeq \subseteq \text{TERM}_{\text{sub}} \times \text{TERM}_{\text{sus}}$$

$$\frac{}{x \simeq x}\ \text{(var-sim)} \qquad \frac{t_{a_1} \simeq t_{b_1} \quad t_{a_2} \simeq t_{b_2}}{t_{a_1}\, t_{a_2} \simeq t_{b_1}\, t_{b_2}}\ \text{(app-sim)} \qquad \frac{t_a \simeq t_b}{(\lambda x.t_a) \simeq (\lambda x.t_b)}\ \text{(lam-sim)}$$

$$\frac{}{n \simeq n}\ \text{(num-sim)} \qquad \frac{t_{a_1} \simeq t_{b_1} \quad t_{a_2} \simeq t_{b_2}}{t_{a_1} + t_{a_2} \simeq t_{b_1} + t_{b_2}}\ \text{(plus-sim)} \qquad \frac{t_a \simeq t_b \quad w_a \simeq w_b}{t_a[w_a/x] \simeq t_b[x := w_b]}\ \text{(subst-sim)}$$

Remark 234. The bisimulation relation is the same as the one in proving the equivalence of (Substitutional) ISWIM and Explicit ISWIM.

Remark 235. The bisimulation relation $\sim$ is up to alpha equivalence. We immediately have: (1) if $t_{a_1} \simeq t_b$ and $t_{a_1} \sim_\alpha t_{a_2}$ then $t_{a_2} \simeq t_b$, and (2) if $t_a \simeq t_{b_1}$ and $t_{b_1} \sim_\alpha t_{b_2}$ then $t_a \simeq t_{b_2}$.

B.2.2 Unloading Function

We define $U(t)$ to unload an Suspended ISWIM term t to (Substitutional) ISWIM.

Definition 236 (Unloading Function)**.** Define the unloading function U to be a total function from the set of terms in Suspended ISWIM to the set of terms in (Substitutional) ISWIM.

$$U \ : \ \text{TERM}_{\text{sus}} \longrightarrow \text{TERM}_{\text{sub}}$$

$$\begin{array}{rcl} U(x) & = & x \\ U(t_1\ t_2) & = & U(t_1)\,U(t_2) \\ U(\lambda x.t) & = & \lambda x.U(t) \\ U(n) & = & n \\ U(t_1+t_2) & = & U(t_1)+U(t_2) \\ U(t[x:=w]) & = & U(t)[U(w)/x] \end{array}$$

Lemma 237 (Equality of Related Terms w.r.t. Unloading Function)**.** *If $t_a \simeq t_b$, then $t_a = U(t_b)$.*

Proof. We proceed by structural induction on $t_a \simeq t_b$.

Case 1. (var-sim). Then, $t_a = t_b = x$. We immediately get $x = U(x)$.

Case 2. (app-sim). Then, $t_a = t_{a_1}\ t_{a_2}$ and $t_b = t_{b_1}\ t_{b_2}$ where $t_{a_1} \simeq t_{b_1}$ and $t_{a_2} \simeq t_{b_2}$. By the induction hypothesis, $t_{a_1} = U(t_{b_1})$ and $t_{a_2} = U(t_{b_2})$. Hence $U(t_{b_1}\ t_{b_2}) = U(t_{b_1})\,U(t_{b_2}) = t_{a_1}\ t_{a_2}$.

Case 3. (lam-sim). Then, $t_a = \lambda x.t_{a_1}$ and $t_b = \lambda x.t_{b_1}$ where $t_{a_1} \simeq t_{b_1}$. By the induction hypothesis, $t_{a_1} = U(t_{b_1})$. Hence $U(\lambda x.t_{b_1}) = \lambda x.U(t_{b_1}) = \lambda x.t_{b_1}$.

Case 4. (num-sim). Then, $t_a = t_b = n$. We immediately get $U(n) = n$.

Case 5. (plus-sim). Then, $t_a = t_{a_1} + t_{a_2}$ and $t_b = t_{b_1} + t_{b_2}$ where $t_{a_1} \simeq t_{b_1}$ and $t_{a_2} \simeq t_{b_2}$. By the induction hypothesis, $t_{a_1} = U(t_{b_1})$ and $t_{a_2} = U(t_{b_2})$. Hence $U(t_{b_1} + t_{b_2}) = U(t_{b_1}) + U(t_{b_2}) = t_{a_1} + t_{a_2}$.

Case 6. (subst-sim). Then, $t_a = t_{a_1}[w_{a_1}/x]$ and $t_b = t_{b_1}[x := w_{b_1}]$ where $t_{a_1} \simeq t_{b_1}$ and $w_{a_1} \simeq w_{b_1}$. By the induction hypothesis, $t_{a_1} = U(t_{b_1})$ and $w_{a_1} = U(w_{b_1})$. Hence $U(t_{b_1}[x := w_{b_1}]) = U(t_{b_1})[U(w_{b_1})/x] = t_{a_1}[w_{a_1}/x]$

□

B.2.3 Substitution Normal Form

In Suspended ISWIM, the terms that cannot perform substitution reduction are in substitution normal form.

Definition 238 (Substitution Normal Form)**.** A term $t \in \text{TERM}_{\text{sus}}$ is in *substitution normal form* if and only if $t \not\longrightarrow^{\text{x}}$.

Remark 239. We use the metavariable s with or without any subscript or superscript to range over the terms in substitution normal form.

Remark 240. A Suspended ISWIM term in substitution normal is not necessarily in the normal form with respect to the single-step relation $\longrightarrow$. For example, $(\lambda x.t)\ v$ is in substitution normal form but is not in the normal form with respect to the single-step relation $\longrightarrow$.

Remark 241. In Suspended ISWIM, $(\lambda x.t)\overline{[x_i := v_i]}$ is in substitution normal form. However, in Explicit ISWIM, $(\lambda x.t)\overline{[x_i := v_i]}$ is not in substitution normal form.

Lemma 242. *If* $t_{a_1} \simeq t_{b_1}$ *and* $t_{b_1} = (\lambda x.t_{b_{11}})\overline{[x_i := w_{b_i}]}_{i=1}^{n}$, *then* $t_{a_1} = (\lambda x.t_{a_{11}})\overline{[w_{a_i}/x_i]}_{i=1}^{n}$, $t_{a_{11}} \simeq t_{b_{11}}$ *and* $w_{a_i} \simeq w_{b_i}$ *for any* $i = 1,2,...,n$.

Proof. We proceed by induction n.

Case 1. (0). Let $t_{b_1} = \lambda x.t_{b_{11}}$. We proceed by cases on $t_{a_1} \simeq \lambda x.t_{b_{11}}$. The only case is (lam-subst). Then, $t_{a_1} = \lambda x.t_{a_{11}}$ and $t_{a_{11}} \simeq t_{b_{11}}$.

Case 2. $(n+1)$. Let $t_{b_1} = (\lambda x.t_{b_{11}})\overline{[x_i := w_{b_i}]}_{i=1}^{n}[x_{n+1} := w_{b_{n+1}}]$.

We proceed by cases on $t_{a_1} \simeq (\lambda x.t_{b_{11}})\overline{[x_i := w_{b_i}]}_{i=1}^{n}[x_{n+1} := w_{b_{n+1}}]$. The only case is (subst-subst). Then, $t_{a_1} = t_{a_2}[w_{a_{n+1}}/x_{n+1}]$, $t_{a_2} \simeq (\lambda x.t_{b_{11}})\overline{[x_i := w_{b_i}]}_{i=1}^{n}$, and $w_{a_{n+1}} \simeq w_{b_{n+1}}$. By the induction hypothesis, $t_{a_2} = (\lambda x.t_{a_{11}})\overline{[w_{a_i}/x_i]}_{i=1}^{n}$, $t_{a_{11}} \simeq t_{b_{11}}$, and $w_{a_i} \simeq w_{b_i}$ for any $i = 1,2,...,n$. Therefore, $t_{a_1} = (\lambda x.t_{a_{11}})\overline{[w_{a_i}/x_i]}_{i=1}^{n+1}$, $t_{a_{11}} \simeq t_{b_{11}}$, and $w_{a_i} \simeq w_{b_i}$ for any $i = 1,2,...,n,n+1$.

□

Lemma 243. *If* $t_{a_1} \simeq w_{b_1}$ *and* $w_{b_1} \in \text{VALUE}_{\text{sus}}$, *then* $t_{a_1} \in \text{VALUE}_{\text{sub}}$.

Proof. We proceed by cases on $w_{b_1} \in \text{VALUE}_{\text{sus}}$.

Case 1. $(w_{b_1} = n)$. We proceed by cases on $t_{a_1} \simeq n$. The only case is (num-sim). Then, $t_{a_1} = n$ and $t_{a_1} \in \text{VALUE}_{\text{sub}}$.

Case 2. $(w_{b_1} = (\lambda x.t_{b_{11}})\overline{[x_i := w_{b_{0i}}]}_{i=1}^{n})$. By Lemma 242, $t_{a_1} = (\lambda x.t_{a_{11}})\overline{[w_{a_{0i}}/x_i]}_{i=1}^{n}$, $t_{a_{11}} \simeq t_{b_{11}}$ and $w_{a_{0i}} \simeq w_{b_{0i}}$ for any $i = 1,2,...,n$. Then, $t_{a_1} \in \text{VALUE}_{\text{sub}}$.

□

Lemma 244. *If* $t_a \simeq t_{b_1}[x := w_{b_1}]$, *then* $t_{b_1} \longrightarrow^{\text{x}*} s_{b_1}$, $s_{b_1}[x := w_{b_1}] \longrightarrow^{\text{x}*} s_{b_2}$, *and* $t_a \simeq s_{b_2}$.

Proof. We proceed by structural induction on $t_a \simeq t_{b_1}[x := w_{b_1}]$. Only (subst-sim) applies. Let $t_a = t_{a_1}[w_{a_1}/x]$ and we have

$$\frac{t_{a_1} \simeq t_{b_1} \quad w_{a_1} \simeq w_{b_1}}{t_{a_1}[w_{a_1}/x] \simeq t_{b_1}[x := w_{b_1}],}$$

We proceed by cases on $t_{a_1} \in \text{TERM}_{\text{sub}}$.

Case 1. $(t_{a_1} = x)$ We have

$$\frac{x \simeq t_{b_1} \quad w_{a_1} \simeq w_{b_1}}{x[w_{a_1}/x] \simeq t_{b_1}[x := w_{b_1}].}$$

Then, $x[w_{a_1}/x] = w_{a_1}$. We proceed by cases on $x \simeq t_{b_1}$.

Case i. (var-sim). Let $t_{b_1} = x$. Then, $x \longrightarrow^{\mathrm{x}*} x$, $x[x := w_{b_1}] \longrightarrow^{\mathrm{x}} w_{b_1}$, and $w_{a_1} \simeq w_{b_1}$.

Case ii. (subst-sim). Let $t_{b_1} = t_{b_{11}}[x_1 := w_{b_{11}}]$. Given $x \simeq t_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t_{b_{11}} \longrightarrow^{\mathrm{x}*} s_{b_{11}}$, $s_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}*} s_{b_{12}}$, and $x \simeq s_{b_{12}}$. We proceed by cases on $x \simeq s_{b_{12}}$.

Case a. (var-sim). Let $s_{b_{12}} = x$. Then, $t_{b_1} \longrightarrow^{\mathrm{x}*} x$, $x[x := w_{b_1}] \longrightarrow^{\mathrm{x}} w_{b_1}$, and $w_{a_1} \simeq w_{b_1}$.

Case b. (subst-sim). This case is vacuous by Lemma 242.

Case 2. ($t_{a_1} = x_0$ and $x_0 \not\equiv x$). We have

$$\frac{x_0 \simeq t_{b_1} \quad w_{a_1} \simeq w_{b_1}}{x_0[w_{a_1}/x] \simeq t_{b_1}[x := w_{b_1}]}.$$

Then, $x_0[w_{a_1}/x] = x_0$. We proceed by cases on $x_0 \simeq t_{b_1}$.

Case i. (var-sim). Let $t_{b_1} = x_0$. Then, $x_0 \longrightarrow^{\mathrm{x}*} x_0$, $x_0[x := w_{b_1}] \longrightarrow^{\mathrm{x}} x_0$, and $x_0 \simeq x_0$.

Case ii. (subst-sim). Let $t_{b_1} = t_{b_{11}}[x_1 := w_{b_{11}}]$. Given $x_0 \simeq t_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t_{b_{11}} \longrightarrow^{\mathrm{x}*} s_{b_{11}}$, $s_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}*} s_{b_{12}}$, and $x_0 \simeq s_{b_{12}}$. We proceed by cases on $x_0 \simeq s_{b_{12}}$.

Case a. (var-sim). Let $s_{b_{12}} = x_0$. Then, $t_{b_1} \longrightarrow^{\mathrm{x}*} x_0$, $x_0[x := w_{b_1}] \longrightarrow^{\mathrm{x}*} x_0$, and by (var-sim) $x_0 \simeq x_0$.

Case b. (subst-sim). This case is vacuous by Lemma 242.

Case 3. ($t_{a_1} = (t_{a_{11}}\ t_{a_{12}})$). We have

$$\frac{(t_{a_{11}}\ t_{a_{12}}) \simeq t_{b_1} \quad w_{a_1} \simeq w_{b_1}}{(t_{a_{11}}\ t_{a_{12}})[w_{a_1}/x] \simeq t_{b_1}[x := w_{b_1}]}.$$

Then, $(t_{a_{11}}\ t_{a_{12}})[w_{a_1}/x] = (t_{a_{11}}[w_{a_1}/x])\ (t_{a_{12}}[w_{a_1}/x])$. We proceed by cases on $(t_{a_{11}}\ t_{a_{12}}) \simeq t_{b_1}$.

Case i. (app-sim). Let $t_{b_1} = (t_{b_{11}}\ t_{b_{12}})$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $t_{a_{12}} \simeq t_{b_{12}}$. We have $(t_{b_{11}}\ t_{b_{12}}) \longrightarrow^{\mathrm{x}*} (t_{b_{11}}\ t_{b_{12}})$ and $(t_{b_{11}}\ t_{b_{12}})[x := w_{b_1}] \longrightarrow^{\mathrm{x}} (t_{b_{11}}[x := w_{b_1}])\ (t_{b_{12}}[x := w_{b_1}])$. By (subst-sim) and (app-sim), we get $(t_{a_{11}}[w_{a_1}/x])\ (t_{a_{12}}[w_{a_1}/x]) \simeq (t_{b_{11}}[x := w_{b_1}])\ (t_{b_{12}}[x := w_{b_1}])$.

Case ii. (subst-sim). Let $t_{b_1} = t_{b_{11}}[x_1 := w_{b_{11}}]$. Given $(t_{a_{11}}\ t_{a_{12}}) \simeq t_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t_{b_{11}} \longrightarrow^{\mathrm{x}*} s_{b_{11}}$, $s_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}*} s_{b_{12}}$, and $(t_{a_{11}}\ t_{a_{12}}) \simeq s_{b_{12}}$. We proceed by cases on $(t_{a_{11}}\ t_{a_{12}}) \simeq s_{b_{12}}$.

Case a. (app-sim). Let $s_{b_{12}} = (t_{b_{121}}\ t_{b_{122}})$ where $t_{a_{11}} \simeq t_{b_{121}}$ and $t_{a_{12}} \simeq t_{b_{122}}$. Then, $t_{b_1} \longrightarrow^{\mathrm{x}*} (t_{b_{121}}\ t_{b_{122}})$ and $(t_{b_{121}}\ t_{b_{122}})[x := w_{b_1}] \longrightarrow^{\mathrm{x}} (t_{b_{121}}[x := w_{b_1}])\ (t_{b_{122}}[x := w_{b_1}])$. By (subst-sim) and (app-sim), we get $(t_{a_{11}}[w_{a_1}/x])\ (t_{a_{12}}[w_{a_1}/x]) \simeq (t_{b_{121}}[x := w_{b_1}])\ (t_{b_{122}}[x := w_{b_1}])$.

Case b. (subst-sim). This case is vacuous by Lemma 242.

Case 4. $(t_{a_1} = \lambda x_0.t_{a_{11}})$. We have

$$\frac{(\lambda x_0.t_{a_{11}}) \simeq t_{b_1} \quad w_{a_1} \simeq w_{b_1}}{(\lambda x_0.t_{a_{11}})[w_{a_1}/x] \simeq t_{b_1}[x := w_{b_1}]}.$$

Then, $(\lambda x_0.t_{a_{11}})[w_{a_1}/x] = \lambda x_1.t_{a_{11}}[x_1/x_0][w_{a_1}/x]$ where $x_1 \notin FV(\lambda x_0.t_{a_{11}}) \cup FV(w_{a_1}) \cup \{x\}$. We proceed by cases on $(\lambda x_0.t_{a_{11}}) \simeq t_{b_1}$.

Case i. (lam-sim). Let $t_{b_1} = \lambda x_0.t_{b_{11}}$ where $t_{a_{11}} \simeq t_{b_{11}}$. We have $\lambda x_0.t_{b_{11}} \longrightarrow^{x*} \lambda x_0.t_{b_{11}}$, $(\lambda x_0.t_{b_{11}})[x := w_{b_1}] \longrightarrow^{x*} (\lambda x_0.t_{b_{11}})[x := w_{b_1}]$, and $(\lambda x_0.t_{a_{11}})[w_{a_1}/x] \simeq (\lambda x_0.t_{b_{11}})[x := w_{b_1}]$.

Case ii. (subst-sim). Let $t_{b_1} = t_{b_{11}}[x_1 := w_{b_{11}}]$. Given $\lambda x_0.t_{a_{11}} \simeq t_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t_{b_{11}} \longrightarrow^{x*} s_{b_{11}}$, $s_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{x*} s_{b_{12}}$, and $\lambda x_0.t_{a_{11}} \simeq s_{b_{12}}$. We proceed by cases on $(\lambda x_0.t_{a11}) \simeq s_{b12}$.

Case a. (lam-sim). Let $s_{b_{12}} = \lambda x_0.t_{b_{121}}$ and $t_{a_{11}} \simeq t_{b_{121}}$.
We have $t_{b_1} \longrightarrow^{x*} \lambda x_0.t_{b_{121}}$, $(\lambda x_0.t_{b_{121}})[x := w_{b_1}] \longrightarrow^{x*} (\lambda x_0.t_{b_{121}})[x := w_{b_1}]$, and by (subst-sim) $(\lambda x_0.t_{a_{11}})[w_{a_1}/x] \simeq (\lambda x_0.t_{b_{121}})[x := w_{b_1}]$.

Case b. (subst-sim). Let $s_{b_{12}} = (\lambda x_{-1}.t_{b_{121}})\overline{[x_i := w_{b_{0i}}]}_{i=1}^{+n}$.
By Lemma 242, $\lambda x.t_{a_{11}} = (\lambda x_{-1}.t_{a_{111}})\overline{[w_{a_{0i}}/x_i]}_{i=1}^{+n}$, $t_{a_{111}} \simeq t_{b_{121}}$, and $w_{a_{0i}} \simeq w_{b_{0i}}$ for any $i = 1,2,...,n$. We have $t_{b_1} \longrightarrow^{x*} (\lambda x_{-1}.t_{b_{121}})\overline{[x_i := w_{b_{0i}}]}_{i=1}^{+n}$, $(\lambda x_{-1}.t_{b_{121}})\overline{[x_i := w_{b_{0i}}]}_{i=1}^{+n}[x := w_{b_1}] \longrightarrow^{x*} (\lambda x_{-1}.t_{b_{121}})\overline{[x_i := w_{b_{0i}}]}_{i=1}^{+n}[x := w_{b_1}]$, and by (subst-sim) $(\lambda x_0.t_{a_{11}})[w_{a_1}/x_1] \simeq (\lambda x_{-1}.t_{b_{121}})\overline{[x_i := w_{b_i}]}_{i=1}^{+n}[x := w_{b_1}]$.

Case 5. $(t_{a_1} = n)$. We have

$$\frac{n \simeq t_{b_1} \quad w_{a_1} \simeq w_{b_1}}{n[w_{a_1}/x] \simeq t_{b_1}[x := w_{b_1}]}.$$

Then, $n[w_{a_1}/x] = n$. We proceed by cases on $n \simeq t_{b_1}$.

Case i. (num-sim). Let $t_{b_1} = n$. We have $n \longrightarrow^{x*} n$, $n[x := w_{b1}] \longrightarrow^{x} n$ and $n \simeq n$.

Case ii. (subst-sim). Let $t_{b_1} = t_{b_{11}}[x_1 := w_{b_{11}}]$. Given $n \simeq t_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t_{b_{11}} \longrightarrow^{x*} s_{b_{11}}$, $s_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{x*} s_{b_{12}}$, and $n \simeq s_{b_{12}}$. We proceed by cases on $n \simeq s_{b_{12}}$.

Case a. (num-sim). Let $s_{b_{12}} = n$. We have $t_{b_1} \longrightarrow^{x*} n$, $n[x := w_{b_1}] \longrightarrow^{x*} n$ and by (num-sim) $n \simeq n$.

Case b. (subst-sim). This case is vacuous by Lemma 242.

Case 6. $(t_{a_1} = t_{a_{11}} + t_{a_{12}})$. We have

$$\frac{t_{a_{11}} + t_{a_{12}} \simeq t_{b_1} \quad w_{a_1} \simeq w_{b_1}}{(t_{a_{11}} + t_{a_{12}})[w_{a_1}/x] \simeq t_{b_1}[x := w_{b_1}]}.$$

Then, $(t_{a_{11}} + t_{a_{12}})[w_{a_1}/x] = t_{a_{11}}[w_{a_1}/x] + t_{a_{12}}[w_{a_1}/x]$. We proceed by cases on $t_{a_{11}} + t_{a_{12}} \simeq t_{b_1}$.

Case i. (plus-sim). Let $t_{b_1} = t_{b_{11}} + t_{b_{12}}$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $t_{a_{12}} \simeq t_{b_{12}}$. We have $t_{b_{11}} + t_{b_{12}} \longrightarrow^{\mathrm{x}*} t_{b_{11}}\ t_{b_{12}}$ and $(t_{b_{11}} + t_{b_{12}})[x := w_{b_1}] \longrightarrow^{\mathrm{x}} t_{b_{11}}[x := w_{b_1}] + t_{b_{12}}[x := w_{b_1}]$. By (subst-sim) and (plus-sim), we get $t_{a_{11}}[w_{a_1}/x] + t_{a_{12}}[w_{a_1}/x] \simeq t_{b_{11}}[x := w_{b_1}] + t_{b_{12}}[x := w_{b_1}]$.

Case ii. (subst-sim). Let $t_{b_1} = t_{b_{11}}[x_1 := w_{b_{11}}]$. Given $t_{a_{11}} + t_{a_{12}} \simeq t_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t_{b_{11}} \longrightarrow^{\mathrm{x}*} s_{b_{11}}$, $s_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}*} s_{b_{12}}$, and $t_{a_{11}} + t_{a_{12}} \simeq s_{b_{12}}$. We proceed by cases on $t_{a_{11}} + t_{a_{12}} \simeq s_{b_{12}}$.

Case a. (plus-sim). Let $s_{b_{12}} = t_{b_{121}} + t_{b_{122}}$ where $t_{a_{11}} \simeq t_{b_{121}}$ and $t_{a_{12}} \simeq t_{b_{122}}$. Then, $t_{b_1} \longrightarrow^{\mathrm{x}*} t_{b_{121}} + t_{b_{122}}$ and $(t_{b_{121}} + t_{b_{122}})[x := w_{b_1}] \longrightarrow^{\mathrm{x}} t_{b_{121}}[x := w_{b_1}] + t_{b_{122}}[x := w_{b_1}]$. By (subst-sim) and (plus-sim), we get $t_{a_{11}}[w_{a_1}/x] + t_{a_{12}}[w_{a_1}/x] \simeq t_{b_{121}}[x := w_{b_1}] + t_{b_{122}}[x := w_{b_1}]$.

Case b. (subst-sim). This case is vacuous by Lemma 242.

□

B.2.4 Canonisation

Given two related terms, if one term is a value, then the other term is a value or multi-steps to a value. We have the following two lemmas.

Lemma 245 (Canonisation of (Substitutional) ISWIM)**.** *If $t_{a_1} \simeq v_{b_1}$, then $t_{a_1} \in \mathrm{VALUE}_{\mathrm{sub}}$.*

Proof. We proceed by structural induction on $t_{a_1} \simeq v_{b_1}$.

Case 1. (var-sim). This case is vacuous.

Case 2. (app-sim). This case is vacuous.

Case 3. (lam-sim). Let $t_{a_1} = \lambda x.t_{a_{11}}$ and $v_{b_1} = \lambda x.t_{b_{11}}$ where $t_{a_{11}} \simeq t_{b_{11}}$. We have $\lambda x.t_{a_{11}} \in \mathrm{VALUE}_{\mathrm{sub}}$.

Case 4. (num-sim). Let $t_{a_1} = v_{b_1} = n$. We have $n \in \mathrm{VALUE}_{\mathrm{sub}}$.

Case 5. (plus-sim). This case is vacuous.

Case 6. (subst-sim). Let $v_{b_1} = (\lambda x.t_{b_{11}})\overline{[x_i := w_{b_i}]}_{i=1}^{+n}$. By Lemma 242, $t_{a_1} = (\lambda x.t_{a_{11}})\overline{[w_{a_i}/x_i]}_{i=1}^{+n}$, $t_{a_{11}} \simeq t_{b_{11}}$, and $w_{a_i} \simeq w_{b_i}$ for any $i = 1, 2, ..., n$. We have $(\lambda x.t_{a_{11}})\overline{[w_{a_i}/x_i]}_{i=1}^{+n} \in \mathrm{VALUE}_{\mathrm{sub}}$.

□

Lemma 246 (Canonisation of Suspended ISWIM)**.** *If $v_{a_1} \simeq t_{b_1}$, then $t_{b_1} \longrightarrow^{*} v_{b_2}$ and $v_{a_1} \simeq v_{b_2}$.*

Proof. We proceed by structural induction on $v_{a_1} \simeq t_{b_1}$.

Case 1. (var-sim). This case is vacuous.

Case 2. (app-sim). This case is vacuous.

Case 3. (lam-sim). Let $v_{a_1} = \lambda x.t_{a_{11}}$ and $t_{b_1} = \lambda x.t_{b_{11}}$ where $t_{a_{11}} \simeq t_{b_{11}}$. We have $\lambda x.t_{b_{11}} \longrightarrow^* \lambda x.t_{b_{11}}$, $\lambda x.t_{b_{11}} \in \text{VALUE}_{\text{sus}}$ and $v_{a_1} \simeq \lambda x.t_{b_{11}}$.

Case 4. (num-sim). Let $v_{a_1} = t_{b_1} = n$. We have $n \longrightarrow^* n$, $n \in \text{VALUE}_{\text{exp}}$, and $v_{a_1} \simeq n$.

Case 5. (plus-sim). This case is vacuous.

Case 6. (subst-sim). Let $v_{a_1} = t_{a_{11}}[w_{a_1}/x]$ and $t_{b_1} = t_{b_{11}}[x := w_{b_1}]$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $w_{a_1} \simeq w_{b_1}$. By Lemma 244, we get $t_{b_{11}}[x := w_{b_1}] \longrightarrow^* s_{b_{12}}$ and $t_{a_{11}}[w_{a_1}/x] \simeq s_{b_{12}}$. We proceed by cases on $t_{a_{11}}[w_{a_1}/x] \simeq s_{b_{12}}$ where $t_{a_{11}}[w_{a_1}/x] \in \text{VALUE}_{\text{sub}}$.

Case i. (var-sim). This case is vacuous.

Case ii. (app-sim). This case is vacuous.

Case iii. (lam-sim). Let $t_{a_{11}}[w_{a_1}/x] = \lambda x_1.t_{a_{111}}$ and $s_{b_{12}} = \lambda x_1.t_{b_{121}}$ where $t_{a_{111}} \simeq t_{b_{121}}$. We have $t_{b_{11}}[x := w_{b_1}] \longrightarrow^* \lambda x_1.t_{b_{121}}$, $\lambda x_1.t_{b121} \in \text{VALUE}_{\text{sus}}$ and $v_{a_1} \simeq \lambda x_1.t_{b_{121}}$.

Case iv. (num-sim). Let $t_{a_{11}}[w_{a_1}/x] = s_{b_{12}} = n$. We have $t_{b_{11}}[x := w_{b_1}] \longrightarrow^* n$, $n \in \text{VALUE}_{\text{exp}}$ and $v_{a_1} \simeq n$.

Case v. (plus-sim). This case is vacuous.

Case vi. (subst-sim). Let $s_{b_{12}} = (\lambda x.t_{b_{121}})\overline{[x_i := w_{b_{0i}}]}_{i=1}^{n}$. We have $t_{b_{11}}[x := w_{b_1}] \longrightarrow^* (\lambda x.t_{b_{121}})\overline{[x_i := w_{b_{0i}}]}_{i=1}^{n}$, $(\lambda x.t_{b_{121}})\overline{[x_i := w_{b_{0i}}]}_{i=1}^{n} \in \text{VALUE}_{\text{sus}}$, and $v_{a_1} \simeq (\lambda x.t_{b_{121}})\overline{[x_i := w_{b_{0i}}]}_{i=1}^{n}$.

□

B.2.5 Explicit Substitution Descendant Relation

We define the explicit substitution descendant relation and show its well-foundedness. As a result, we can do induction on explicit substitution descendants.

Definition 247 (Explicit Substitution Descendant Relation)**.** For any $t_1, t_2 \in \text{TERM}_{\text{sus}}$, $t_1 \prec^{\text{x}} t_2$ if and only if $t_2 \longrightarrow^{\text{x}} t_1$. We call $\prec^{\text{x}}$ the explicit substitution descendant relation.

Definition 248 (Weight Function)**.** For any $t \in \text{TERM}_{\text{sus}}$, its weight is $W(t)$ where W is a function defined as follows.

$$W \ : \ \text{TERM}_{\text{sus}} \longrightarrow \mathbb{Z}^+$$

$$\begin{aligned} W(x) &= 1 \\ W(t_1\ t_2) &= W(t_1)+W(t_2)+1 \\ W(\lambda x.t) &= 1 \\ W(n) &= 1 \\ W(t_1+t_2) &= W(t_1)+W(t_2)+1 \\ W(t[x:=w]) &= W(t)\cdot(W(w)+1) \end{aligned}$$

Lemma 249 (Substitution reduction decreases weight.)**.** *For any $t_1, t_2 \in \text{TERM}_{\text{sus}}$, if $t_1 \longrightarrow^{\text{x}} t_2$, then $W(t_2) < W(t_1)$.*

Proof. We proceed by structural induction on $t_1 \longrightarrow^{\text{x}} t_2$.

Case 1. (var-eq-subst). Let $t_1 = x[x:=w]$ and $t_2 = w$. Then, $W(x[x:=w]) = W(x)\cdot(W(w)+1) = W(w)+1$. We have $W(w) < W(w)+1 = W(x[x:=w])$.

Case 2. (var-dif-subst). Let $t_1 = x_1[x_2:=w]$ and $t_2 = x_1$ where $x_1 \not\equiv x_2$. Then, $W(x_1[x_2:=w]) = W(x_1)\cdot(W(w)+1) = W(w)+1$. We have $W(x_1) = 1 < W(w)+1 = W(x_1[x_2:=w])$.

Case 3. (num-subst). Let $t_1 = n[x:=w]$ and $t_2 = n$. Then, $W(n[x:=w]) = W(n)\cdot(W(w)+1) = W(w)+1$. We have $W(n) = 1 < W(w)+1 = W(n[x:=w])$.

Case 4. (app-subst). Let $t_1 = (t_{11}\,t_{12})[x:=w]$ and $t_2 = (t_{11}[x:=w])\ (t_{12}[x:=w])$. Then, $W(t_{11}[x:=w]\ t_{12}[x:=w]) = W(t_{11}[x:=w]) + W(t_{12}[x:=w]) + 1 = W(t_{11})\cdot(W(w)+1) + W(t_{12})\cdot(W(w)+1) + 1 = (W(t_{11})+W(t_{12}))\cdot(W(w)+1)+1$ and $W((t_{11}\ t_{12})[x:=w]) = W(t_{11}\ t_{12})\cdot(W(w)+1) = (W(t_{11})+W(t_{12})+1)\cdot(W(w)+1) = (W(t_{11})+W(t_{12}))\cdot(W(w)+1) + W(w)+1$. We have $W((t_{11}[x:=w])\ (t_{12}[x:=w])) = (W(t_{11})+W(t_{12}))\cdot(W(w)+1)+1 < (W(t_{11})+W(t_{12}))\cdot(W(w)+1)+W(w)+1 = W((t_{11}\,t_{12})[x:=w])$.

Case 5. (plus-subst). Let $t_1 = (t_{11}+t_{12})[x:=w]$ and $t_2 = t_{11}[x:=w]+t_2[x:=w])$. Then, $W(t_{11}[x:=w]+t_{12}[x:=w]) = W(t_{11}[x:=w])+W(t_{12}[x:=w])+1 = W(t_{11})\cdot(W(w)+1)+W(t_{12})\cdot(W(w)+1)+1 = (W(t_{11})+W(t_{12}))\cdot(W(w)+1)+1$ and $W((t_{11}+t_{12})[x:=w]) = W(t_{11}+t_{12})\cdot(W(w)+1) = (W(t_{11})+W(t_{12})+1)\cdot(W(w)+1) = (W(t_{11})+W(t_{12}))\cdot(W(w)+1)+W(w)+1$. We have $W(t_{11}[x:=w]+t_{12}[x:=w]) = (W(t_{11})+W(t_{12}))\cdot(W(w)+1)+1 < (W(t_{11})+W(t_{12}))\cdot(W(w)+1)+W(w)+1 = W((t_{11}+t_{12})[x:=w])$.

Case 6. (subst-subst). Let $t_1 = t_{11}[x_1:=w_1][x_2:=w_2]$ and $t_2 = t_{21}[x_2:=w_2]$ where $t_{11}[x_1:=w_1] \longrightarrow^{\text{x}} t_{21}$. By the induction hypothesis, $W(t_{21}) < W(t_{11}[x_1:=w_1])$. Then, $W(t_{21}[x_2:=w_2]) = W(t_{21})\cdot(W(w_2)+1)$ and $W(t_{11}[x_1:=w_1][x_2:=w_2]) = W(t_{11}[x_1:=w_1])\cdot(W(w_2)+1)$. We have $W(t_{21}[x_2:=w_2]) = W(t_{21})\cdot(W(w_2)+1) = W(t_{21})\cdot(W(w_2)+1) < W(t_{11}[x_1:=w_1])\cdot(W(w_2)+1) = W(t_{11}[x_1:=w_1][x_2:=w_2])$.

□

Lemma 250 (Well-foundedness of Explicit Substitution Descendant Relation)**.** *The explicit substitution descendant relation $\prec^{\text{x}}$ is well-founded.*

Proof. Lemma 249 has proved that if $t_1 \longrightarrow^{\mathrm{x}} t_2$, then $W(t_2) < W(t_1)$, for any $t_1.t_2 \in \mathrm{TERM}_{\mathrm{sus}}$. For any $t \in \mathrm{TERM}_{\mathrm{sus}}$, the length of the descending chain with respect to $\prec^{\mathrm{x}}$ starting from t is bound by $W(t)$. Hence, the explicit substitution descendant relation $\prec^{\mathrm{x}}$ is well-founded. □

B.2.6 Bisimulation

We demonstrate (Substitutional) ISWIM bisimulates Suspended ISWIM. Intuitively, given two related terms, if one term single-steps, then the other term multi-steps, and the resulting two terms are related.

Lemma 251 (Simulation: Suspended ISWIM simulates (Substitutional) ISWIM.)**.** *If $t_{a_1} \simeq t_{b_1}$ and $t_{a_1} \longrightarrow t_{a_2}$ where $FV(t_{a_1}) = \emptyset$ and $FV(t_{b_1}) = \emptyset$, then $t_{b_1} \longrightarrow^* t_{b_2}$ and $t_{a_2} \simeq t_{b_2}$ where $FV(t_{a_2}) = \emptyset$ and $FV(t_{b_2}) = \emptyset$.*

Proof. We proceed by induction on the structure of $t_{a_1} \simeq t_{b_1}$ and by induction on the explicit substitution descendants $\prec^{\mathrm{x}} t_{b_1}$ simultaneously.

Case 1. (var-sim). This case is vacuous.

Case 2. (app-sim). Let $t_{a_1} = t_{a_{11}}\ t_{a_{12}}$ and $t_{b_1} = t_{b_{11}}\ t_{b_{12}}$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $t_{a_{12}} \simeq t_{b_{12}}$. We proceed by cases on $t_{a_1} \longrightarrow t_{a_2}$.

Case i. (appL). Let $t_{a_{11}} \longrightarrow t_{a_{21}}$ and $t_{a_2} = t_{a_{21}}\ t_{a_{12}}$. By the induction hypothesis, $t_{b_{11}} \longrightarrow^* t_{b_{21}}$ and $t_{a_{21}} \simeq t_{b_{21}}$. Then, $t_{b_{11}}\ t_{b_{12}} \longrightarrow^* t_{b_{21}}\ t_{b_{12}}$ and by (app-sim) $t_{a_{21}}\ t_{a_{12}} \simeq t_{b_{21}}\ t_{b_{12}}$.

Case ii. (appR). Let $t_{a_{11}} = v_{a_{11}}, t_{a_{12}} \longrightarrow t_{a_{22}}$ and $t_{a_2} = v_{a_{11}}\ t_{a_{22}}$. Given $v_{a_{11}} \simeq t_{b_{11}}$, by Lemma 246, $t_{b_{11}} \longrightarrow^* v_{b_{11}}$ and $v_{a_{11}} \simeq v_{b_{11}}$. By the induction hypothesis, $t_{b_{12}} \longrightarrow^* t_{b_{22}}$ and $t_{a_{22}} \simeq t_{b_{22}}$. Then, $t_{b_{11}}\ t_{b_{12}} \longrightarrow^* v_{b_{11}}\ t_{b_{12}} \longrightarrow^* v_{b_{11}}\ t_{b_{22}}$ and by (app-sim) $v_{a_{11}}\ t_{a_{22}} \simeq v_{b_{11}}\ t_{b_{22}}$.

Case iii. (app). Let $t_{a_{11}} = \lambda x.t_{a_{111}}$, $t_{a_{12}} = v_{a_{12}}$, and $t_{a_2} = t_{a_{111}}[v_{a_{12}}/x]$. Given $\lambda x.t_{a_{111}} \simeq t_{b_{11}}$, by Lemma 246, $t_{b_{11}} \longrightarrow^* v_{b_{11}}$ and $\lambda x.t_{a_{111}} \simeq v_{b_{11}}$. Given $v_{a_{12}} \simeq t_{b_{12}}$, by Lemma 246, $t_{b_{12}} \longrightarrow^* v_{b_{12}}$ and $v_{a_{12}} \simeq v_{b_{12}}$. We proceed by cases on $\lambda x.t_{a_{111}} \simeq v_{b_{11}}$.

Case a. (lam-sim). Let $v_{b_{11}} = \lambda x.t_{b_{111}}$ and $t_{a_{111}} \simeq t_{b_{111}}$. Then, $t_{b_{11}}\ t_{b_{12}} \longrightarrow^* (\lambda x.t_{b_{111}})\ t_{b_{12}} \longrightarrow^* (\lambda x.t_{b_{111}})\ v_{b_{12}} \longrightarrow t_{b_{111}}[x := v_{b_{12}}]$. By (subst-sim), we get $t_{a_{111}}[v_{a_{12}}/x] \simeq t_{b_{111}}[x := v_{b_{12}}]$.

Case b. (subst-sim). Let $v_{b_{11}} = (\lambda x_0.t_{b_{111}})\overline{[x_i := w_{b_i}]}^n_{i=1}$. By Lemma 242, we have $\lambda x.t_{a_{111}} = (\lambda x_0.t_{a_{1111}})\overline{[w_{a_i}/x_i]}^n_{i=1}$, $t_{b_{111}} \simeq t_{a_{111}}$ and $w_{a_i} \simeq w_{b_i}$ for any $i = 1, 2, ..., n$. We know $(\lambda x_0.t_{a_{1111}})\overline{[w_{a_i}/x_i]}^n_{i=1} \sim_\alpha \lambda x_{-1}.t_{a_{1111}}[x_{-1}/x_0]\overline{[w_{a_i}/x_i]}^n_{i=1}$ where $x_{-1} \notin FV(\lambda x_0.t_{a_{1111}}) \cup (\bigcup_i (FV(w_{a_i}) \cup \{x_i\}))$. We have $\lambda x.t_{a_{111}} \sim_\alpha \lambda x_{-1}.t_{a_{1111}}[x_{-1}/x_0]\overline{[w_{a_i}/x_i]}^n_{i=1}$. Then, $(\lambda x.t_{a_{111}})\ v_{a_{12}} \sim_\alpha (\lambda x_{-1}.t_{a_{1111}}[x_{-1}/x_0]\overline{[w_{a_i}/x_i]}^n_{i=1})\ v_{a_{12}}$.
We have $(\lambda x_{-1}.t_{a_{1111}}[x_{-1}/x_0]\overline{[w_{a_i}/x_i]}^n_{i=1})\ v_{a_{12}} \longrightarrow t_{a_{1111}}[x_{-1}/x_0]\overline{[w_{a_i}/x_i]}^n_{i=1}[v_{a_{12}}/x_{-1}]$ and $(\lambda x.t_{a_{111}})\ v_{a_{12}} \longrightarrow t_{a_{111}}[v_{a_{12}}/x]$. By Proposition 58, $t_{a_{1111}}[x_{-1}/x_0]\overline{[w_{a_i}/x_i]}^n_{i=1}[v_{a_{12}}/x_{-1}] \sim_\alpha t_{a_{111}}[v_{a_{12}}/x]$.
Given $FV(t_{a_1}) = \emptyset$ and $FV(t_{a_1}) = FV(t_{a_{11}}) \cup FV(t_{a_{12}})$, we have $FV(v_{a_{12}}) = FV(t_{a_{12}}) = \emptyset$.

Given $x_{-1} \not\equiv x_i$ for any $i = 1,2,...,n$, $x_{-1} \notin FV(w_{a_i})$ for any $i = 1,2,...,n$ and $FV(v_{a_{12}}) = \emptyset$, by Proposition 53, we have $t_{a_{1111}}[x_{-1}/x_0]\overline{[w_{a_i}/x_i]}^n_{i=1}[v_{a_{12}}/x_{-1}] \sim_\alpha t_{a_{1111}}[x_{-1}/x_0][v_{a_{12}}/x_{-1}]\overline{[w_{a_i}/x_i]}^n_{i=1} \sim_\alpha t_{a_{1111}}[v_{a_{12}}/x_0]\overline{[w_{a_i}/x_i]}^n_{i=1}$.
Then, $t_{a_{1111}}[v_{a_{12}}/x_0]\overline{[w_{a_i}/x_i]}^n_{i=1} \sim_\alpha t_{a_{111}}[v_{a_{12}}/x]$. By (subst-sim), $t_{a_{1111}}[v_{a_{12}}/x_0]\overline{[w_{a_i}/x_i]}^n_{i=1} \simeq t_{b_{111}}[x_0 := v_{b_{12}}]\overline{[x_i := w_{b_i}]}^n_{i=1}$. Then, $t_{a_{111}}[v_{a_{12}}/x] \simeq t_{b_{111}}[x_0 := v_{b_{12}}]\overline{[x_i := w_{b_i}]}^n_{i=1}$.

Case 3. (lam-sim). This case is vacuous.

Case 4. (num-sim). This case is vacuous.

Case 5. (plus-sim). Let $t_{a_1} = t_{a_{11}} + t_{a_{12}}$ and $t_{b_1} = t_{b_{11}} + t_{b_{12}}$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $t_{a_{12}} \simeq t_{b_{12}}$. We proceed by cases on $t_{a_1} \longrightarrow t_{a_2}$.

Case i. (plusL). Let $t_{a_{11}} \longrightarrow t_{a_{21}}$ and $t_{a_2} = t_{a_{21}} + t_{a_{12}}$. By the induction hypothesis, $t_{b_{11}} \longrightarrow^* t_{b_{21}}$ and $t_{a_{21}} \simeq t_{b_{21}}$. Then, $t_{b_{11}} + t_{b_{12}} \longrightarrow^* t_{b_{21}} + t_{b_{12}}$ and by (plus-sim) $t_{a_{21}} + t_{a_{12}} \simeq t_{b_{21}} + t_{b_{12}}$.

Case ii. (plusR). Let $t_{a_{11}} = v_{a_{11}}$, $t_{a_{12}} \longrightarrow t_{a_{22}}$ and $t_{a_2} = v_{a_{11}} + t_{a_{22}}$. Given $v_{a_{11}} \simeq t_{b_{11}}$, by Lemma 246, $t_{b_{11}} \longrightarrow^* v_{b_{11}}$ and $v_{a_{11}} \simeq v_{b_{11}}$. By the induction hypothesis, $t_{b_{12}} \longrightarrow^* t_{b_{22}}$ and $t_{a_{22}} \simeq t_{b_{22}}$. Then, $t_{b_{11}} + t_{b_{12}} \longrightarrow^* v_{b_{11}} + t_{b_{12}} \longrightarrow^* v_{b_{11}} + t_{b_{22}}$ and by (plus-sim) $v_{a_{11}} + t_{a_{22}} \simeq v_{b_{11}} + t_{b_{22}}$.

Case iii. (plus). Let $t_{a_{11}} = n_1$, $t_{a_{12}} = n_2$, and $t_{a_2} = n$ where $n = n_1 + n_2$. Given $n_1 \simeq t_{b_{11}}$, by Lemma 246, $t_{b_{11}} \longrightarrow^* v_{b_{11}}$ and $n_1 \simeq v_{b_{11}}$. Given $n_2 \simeq t_{b_{12}}$, by Lemma 246, $t_{b_{12}} \longrightarrow^* v_{b_{12}}$ and $n_2 \simeq v_{b_{12}}$. We proceed by cases on $n_1 \simeq v_{b_{11}}$.

Case a. (num-sim). Let $v_{b_{11}} = n_1$. We proceed by cases on $n_2 \simeq v_{b_{12}}$.

Case 1. (num-sim). Let $v_{b_{12}} = n_2$. Then, $t_{b_{11}} + t_{b_{12}} \longrightarrow^* n_1 + t_{b_{12}} \longrightarrow^* n_1 + n_2 \longrightarrow n$ where $n = n_1 + n_2$. By (num-sim), we get $n \simeq n$.

Case 2. (subst-sim). This case is vacuous by Lemma 242.

Case b. (subst-sim). This case is vacuous by Lemma 242.

Case 6. (subst-sim). Let $t_{a_1} = t_{a_{11}}[w_{a_{11}}/x]$ and $t_{b_1} = t_{b_{11}}[x := w_{b_{11}}]$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $w_{a_{11}} \simeq w_{b_{11}}$. Given $t_{a_1} \simeq t_{b_{11}}[x := w_{b_{11}}]$, by Lemma 244, $t_{b_{11}}[x := w_{b_{11}}] \longrightarrow^{x*} s_{b_{21}}$ and $t_{a_1} \simeq s_{b_{21}}$. Then, $s_{b_{21}} \prec^x t_{b_1}$. If $t_{a_1} \longrightarrow t_{a_2}$, by the induction hypothesis, $s_{b_{21}} \longrightarrow^* t_{b_2}$ and $t_{a_2} \simeq t_{b_2}$. We have $t_{b_1} \longrightarrow^* s_{b_{21}} \longrightarrow^* t_{b_2}$.

By Propositions 59 and 84, we know $FV(t_{a_2}) = \emptyset$ and $FV(t_{b_2}) = \emptyset$. □

Remark 252. In the last case of the proof, given $t_{a_1} \simeq t_{b_1}$, $t_{a_1} \simeq s_{b_{21}}$ and $s_{b_{21}} \prec^x t_{b_1}$, if $t_{a_1} \longrightarrow t_{a_2}$, by the induction hypothesis, $s_{b_{21}} \longrightarrow^* t_{b_2}$ and $t_{a_2} \simeq t_{b_2}$.

Lemma 253 (Explicit substitution reduction preserves simulation relation.)**.** *If $t_{a_1} \simeq t_{b_1}$ and $t_{b_1} \longrightarrow^x t_{b_2}$, then $t_{a_1} \simeq t_{b_2}$.*

Proof. We proceed by structural induction on $t_{a_1} \simeq t_{b_1}$. Since $t_{b_1} \longrightarrow^{\mathrm{x}} t_{b_2}$, only (subst-sim) applies. Let $t_{a_1} = t_{a_{11}}[w_{a_{11}}/x_1]$ and $t_{b_1} = t_{b_{11}}[x_1 := w_{b_{11}}]$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $w_{a_{11}} \simeq w_{b_{11}}$. We proceed by cases on $t_{b_1} \longrightarrow^{\mathrm{x}} t_{b_2}$.

Case 1. (var-eq-subst). Let $t_{b_{11}} = x_1$. We have $x_1[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}} w_{b_{11}}$. We proceed by cases on $t_{a_{11}} \simeq x_1$. The only case is (var-sim), thus we get $t_{a_{11}} = x_1$. Then, $x_1[w_{a_{11}}/x_1] = w_{a_{11}}$ and $w_{a_{11}} \simeq w_{b_{11}}$.

Case 2. (var-df-subst). Let $t_{b_{11}} = x_2$ and $x_2 \not\equiv x_1$. We have $x_2[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}} x_2$. We proceed by cases on $t_{a_{11}} \simeq x_2$. The only case is (var-sim), thus we get $t_{a_{11}} = x_2$. Then, $x_2[w_{a_{11}}/x_1] = x_2$ and $x_2 \simeq x_2$.

Case 3. (num-subst). Let $t_{b_{11}} = n$. We have $n[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}} n$. We proceed by cases on $t_{a_{11}} \simeq n$. The only case is (num-sim), thus we get $t_{a_{11}} = n$. Then, $n[w_{a_{11}}/x_1] = n$ and $n \simeq n$.

Case 4. (app-subst). Let $t_{b_{11}} = t_{b_{111}}\ t_{b_{112}}$. We have $(t_{b_{111}}\ t_{b_{112}})[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}} (t_{b_{111}}[x_1 := w_{b_{11}}])\ (t_{b_{112}}[x_1 := w_{b_{11}}])$. We proceed by cases on $t_{a_{11}} \simeq t_{b_{111}}\ t_{b_{112}}$. The only case is (app-sim), thus we get $t_{a_{11}} = t_{a_{111}}\ t_{a_{112}}$, $t_{a_{111}} \simeq t_{b_{111}}$ and $t_{a_{112}} \simeq t_{b_{112}}$. Then, $(t_{a_{111}}\ t_{a_{112}})[w_{a_{11}}/x_1] = (t_{a_{111}}[w_{a_{11}}/x_1])\ (t_{a_{112}}[w_{a_{11}}/x_1])$ and by (subst-sim) and (app-sim) $(t_{a_{111}}[w_{a_{11}}/x_1])\ (t_{a_{112}}[w_{a_{11}}/x_1]) \simeq (t_{b_{111}}[x_1 := w_{b_{11}}])\ (t_{b_{112}}[x_1 := w_{b_{11}}])$.

Case 5. (plus-subst). Let $t_{b_{11}} = t_{b_{111}} + t_{b_{112}}$. We have $(t_{b_{111}} + t_{b_{112}})[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}} t_{b_{111}}[x_1 := w_{b_{11}}] + t_{b_{112}}[x_1 := w_{b_{11}}]$. We proceed by cases on $t_{a_{11}} \simeq t_{b_{111}} + t_{b_{112}}$. The only case is (plus-sim), thus we get $t_{a_{11}} = t_{a_{111}} + t_{a_{112}}$, $t_{a_{111}} \simeq t_{b_{111}}$ and $t_{a_{112}} \simeq t_{b_{112}}$. Then, $(t_{a_{111}} + t_{a_{112}})[w_{a_{11}}/x_1] = t_{a_{111}}[w_{a_{11}}/x_1] + t_{a_{112}}[w_{a_{11}}/x_1]$ and by (subst-sim) and (plus-sim) $t_{a_{111}}[w_{a_{11}}/x_1] + t_{a_{112}}[w_{a_{11}}/x_1] \simeq t_{b_{111}}[x_1 := w_{b_{11}}] + t_{b_{112}}[x_1 := w_{b_{11}}]$.

Case 6. (subst-subst). Let $t_{b_{11}} = t_{b_{111}}[x_2 := w_{b_{12}}]$. We have $(t_{b_{111}}[x_2 := w_{b_{12}}])[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}} t_{b_{121}}[x_1 := w_{b_{11}}]$ where $t_{b_{111}}[x_2 := w_{b_{12}}] \longrightarrow^{\mathrm{x}} t_{b_{121}}$. By the induction hypothesis, $t_{a_{11}} \simeq t_{b_{121}}$. Then, by (subst-sim) $t_{a_{11}}[w_{a_{11}}/x_1] \simeq t_{b_{121}}[x_1 := w_{b_{11}}]$.

□

Lemma 254 (Simulation: (Substitutional) ISWIM simulates Suspended ISWIM.)**.** *If $t_{a_1} \simeq t_{b_1}$ and $t_{b_1} \longrightarrow t_{b_2}$ where $FV(t_{a_1}) = \emptyset$ and $FV(t_{b_1}) = \emptyset$, then $t_{a_1} \longrightarrow^* t_{a_2}$ and $t_{a_2} \simeq t_{b_2}$ where $FV(t_{a_2}) = \emptyset$ and $FV(t_{b_2}) = \emptyset$.*

Proof. We proceed by induction on the structure of $t_{a_1} \simeq t_{b_1}$.

Case 1. (var-sim). This case is vacuous.

Case 2. (app-sim). Let $t_{a_1} = t_{a_{11}}\ t_{a_{12}}$ and $t_{b_1} = t_{b_{11}}\ t_{b_{12}}$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $t_{a_{12}} \simeq t_{b_{12}}$. We proceed by cases on $t_{b_1} \longrightarrow t_{b_2}$.

Case i. (appL). Let $t_{b_{11}} \longrightarrow t_{b_{21}}$ and $t_{b_2} = t_{b_{21}}\ t_{b_{12}}$. By the induction hypothesis, $t_{a_{11}} \longrightarrow^* t_{a_{21}}$ and $t_{a_{21}} \simeq t_{b_{21}}$. Then, $t_{a_{11}}\ t_{a_{12}} \longrightarrow^* t_{a_{21}}\ t_{a_{12}}$ and by (app-sim) $t_{a_{21}}\ t_{a_{12}} \simeq t_{b_{21}}\ t_{b_{12}}$.

Case ii. (appR). Let $t_{b_{11}} = v_{b_{11}}$, $t_{b_{12}} \longrightarrow t_{b_{22}}$ and $t_{b_2} = v_{b_{11}}\ t_{b_{22}}$. Given $t_{a_{11}} \simeq v_{b_{11}}$, by Lemma 245, $t_{a_{11}} \longrightarrow^* v_{a_{11}}$ and $v_{a_{11}} \simeq v_{b_{11}}$. By the induction hypothesis, $t_{a_{12}} \longrightarrow^* t_{a_{22}}$ and $t_{a_{22}} \simeq t_{b_{22}}$. Then, $t_{a_{11}}\ t_{a_{12}} \longrightarrow^* v_{a_{11}}\ t_{a_{12}} \longrightarrow^* v_{a_{11}}\ t_{a_{22}}$ and by (app-sim) $v_{a_{11}}\ t_{a_{22}} \simeq v_{b_{11}}\ t_{b_{22}}$.

Case iii. (app). Let $t_{b_{11}} = (\lambda x.t_{b_{111}})\overline{[x_i := w_{b_i}]}_{i=1}^{n}$, $t_{b_{12}} = v_{b_{12}}$, and $t_{b_2} = t_{b_{111}}[x := v_{b_{12}}]\overline{[x_i := w_{b_i}]}_{i=1}^{n}$. Given $t_{a_{11}} \simeq (\lambda x.t_{b_{111}})\overline{[x_i := w_{b_i}]}_{i=1}^{n}$, by Lemma 242, $t_{a_{11}} = (\lambda x.t_{a_{111}})\overline{[w_{a_i}/x_i]}_{i=1}^{n}$, $t_{a_{111}} \simeq t_{b_{111}}$ and $w_{a_i} \simeq w_{b_i}$ for any $i = 1, 2, ..., n$. Given $t_{a_{12}} \simeq v_{b_{12}}$, by Lemma 245, $t_{a_{12}} \longrightarrow^* v_{a_{12}}$ and $v_{a_{12}} \simeq v_{b_{12}}$.
We know $(\lambda x.t_{a_{111}})\overline{[w_{a_i}/x_i]}_{i=1}^{n} \sim_\alpha \lambda x_{-1}.t_{a_{111}}[x_{-1}/x]\overline{[w_{a_i}/x_i]}_{i=1}^{n}$ where $x_{-1} \notin FV(\lambda x.t_{a_{111}}) \cup (\bigcup_i (FV(w_{a_i}) \cup \{x_i\}))$. Then, $(\lambda x.t_{a_{111}})\overline{[w_{a_i}/x_i]}_{i=1}^{n}\ v_{a_{12}} \sim_\alpha \lambda x_{-1}.t_{a_{111}}[x_{-1}/x]\overline{[w_{a_i}/x_i]}_{i=1}^{n}\ v_{a_{12}}$.
We have $(\lambda x_{-1}.t_{a_{111}}[x_{-1}/x]\overline{[w_{a_i}/x_i]}_{i=1}^{n})\ v_{a_{12}} \longrightarrow t_{a_{111}}[x_{-1}/x]\overline{[w_{a_i}/x_i]}_{i=1}^{n}[v_{a_{12}}/x_{-1}]$. By Proposition 58, $(\lambda x.t_{a_{111}})\overline{[w_{a_i}/x_i]}_{i=1}^{n}\ v_{a_{12}} \longrightarrow t_{a_2}$ and $t_{a_{111}}[x_{-1}/x]\overline{[w_{a_i}/x_i]}_{i=1}^{n}[v_{a_{12}}/x_{-1}] \sim_\alpha t_{a_2}$.
Given $FV(t_{a_1}) = \emptyset$ and $FV(t_{a_1}) = FV(t_{a_{11}}) \cup FV(t_{a_{12}})$, we know $FV(t_{a_{12}}) = \emptyset$. Given $t_{a_{12}} \longrightarrow^* v_{a_{12}}$, by Proposition 59, $FV(v_{a_{12}}) = \emptyset$.
Given $x_{-1} \not\equiv x_i$ for any $i = 1, 2, ..., n$, $x_{-1} \notin FV(w_{a_i})$ for any $i = 1, 2, ..., n$ and $FV(v_{a_{12}}) = \emptyset$, by Proposition 53, we have $t_{a_{111}}[x_{-1}/x]\overline{[w_{a_i}/x_i]}_{i=1}^{n}[v_{a_{12}}/x_{-1}] \sim_\alpha t_{a_{111}}[x_{-1}/x][v_{a_{12}}/x_{-1}]\overline{[w_{a_i}/x_i]}_{i=1}^{n} \sim_\alpha t_{a_{111}}[v_{a_{12}}/x]\overline{[w_{a_i}/x_i]}_{i=1}^{n}$. Then, $t_{a_{111}}[v_{a_{12}}/x]\overline{[w_{a_i}/x_i]}_{i=1}^{n} \sim_\alpha t_{a_2}$.
By (subst-sim), $t_{a_{111}}[v_{a_{12}}/x]\overline{[w_{a_i}/x_i]}_{i=1}^{n} \simeq t_{b_{111}}[x := v_{b_{12}}]\overline{[x_i := w_{b_i}]}_{i=1}^{n}$. Then, $t_{a_2} \simeq t_{b_{111}}[x_0 := v_{b_{12}}]\overline{[x_i := w_{b_i}]}_{i=1}^{n}$.

Case 3. (lam-sim). This case is vacuous.

Case 4. (num-sim). This case is vacuous.

Case 5. (plus-sim). Let $t_{a_1} = t_{a_{11}} + t_{a_{12}}$ and $t_{b_1} = t_{b_{11}} + t_{b_{12}}$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $t_{a_{12}} \simeq t_{b_{12}}$. We proceed by cases on $t_{b_1} \longrightarrow t_{b_2}$.

Case i. (plusL). Let $t_{b_{11}} \longrightarrow t_{b_{21}}$ and $t_{b_2} = t_{b_{21}} + t_{b_{12}}$. By the induction hypothesis, $t_{a_{11}} \longrightarrow^* t_{a_{21}}$ and $t_{a_{21}} \simeq t_{b_{21}}$. Then, $t_{a_{11}} + t_{a_{12}} \longrightarrow^* t_{a_{21}} + t_{a_{12}}$ and by (plus-sim) $t_{a_{21}} + t_{a_{12}} \simeq t_{b_{21}} + t_{b_{12}}$.

Case ii. (plusR). Let $t_{b_{11}} = v_{b_{11}}$, $t_{b_{12}} \longrightarrow t_{b_{22}}$ and $t_{b_2} = v_{b_{11}} + t_{b_{22}}$. Given $t_{a_{11}} \simeq v_{b_{11}}$, by Lemma 245, $t_{a_{11}} \longrightarrow^* v_{a_{11}}$ and $v_{a_{11}} \simeq v_{b_{11}}$. By the induction hypothesis, $t_{a_{12}} \longrightarrow^* t_{a_{22}}$ and $t_{a_{22}} \simeq t_{b_{22}}$. Then, $t_{a_{11}} + t_{a_{12}} \longrightarrow^* v_{a_{11}} + t_{a_{12}} \longrightarrow^* v_{a_{11}} + t_{a_{22}}$ and by (plus-sim) $v_{a_{11}} + t_{a_{22}} \simeq v_{b_{11}} + t_{b_{22}}$.

Case iii. (plus). Let $t_{b_{11}} = n_1$, $t_{b_{12}} = n_2$, and $t_{b_2} = n$ where $n = n_1 + n_2$. Given $t_{a_{11}} \simeq n_1$, by Lemma 245, $t_{a_{11}} \longrightarrow^* v_{a_{11}}$ and $v_{a_{11}} \simeq n_1$. Given $t_{a_{12}} \simeq n_2$, by Lemma 245, $t_{a_{12}} \longrightarrow^* v_{a_{12}}$ and $v_{a_{12}} \simeq n_2$. We proceed by cases on $v_{a_{11}} \simeq n_1$. The only case is (num-sim), so let $v_{a_{11}} = n_1$. We proceed by cases on $v_{a_{12}} \simeq n_2$. The only case is (num-sim), so let $v_{a_{12}} = n_2$. Then, $t_{a_{11}} + t_{a_{12}} \longrightarrow^* n_1 + t_{a_{12}} \longrightarrow^* n_1 + n_2 \longrightarrow n$ where $n = n_1 + n_2$. By (num-sim), we get $n \simeq n$.

Case 6. (subst-sim). Let $t_{a_1} = t_{a_{11}}[w_{a_{11}}/x]$ and $t_{b_1} = t_{b_{11}}[x := w_{b_{11}}]$ where $t_{a_{11}} \simeq t_{b_{11}}$ and $w_{a_{11}} \simeq w_{b_{11}}$. Given $t_{a_1} \simeq t_{b_{11}}[x := w_{b_{11}}]$, by Lemma 244, $t_{b_{11}}[x := w_{b_{11}}] \longrightarrow^{\mathrm{x}*} s_{b_{21}}$ and $t_{a_1} \simeq s_{b_{21}}$. We proceed by cases on whether or not $t_{b_{11}}[x := w_{b_{11}}]$ is in the substitution normal form.

Case i. Let $t_{b_{11}}[x := w_{b_{11}}]$ be not in the substitution normal form. Since $s_{b_{21}}$ is in substitution normal form, we shall have $t_{b_{11}}[x := w_{b_{11}}] \longrightarrow^{\mathrm{x}} t_{b_{21}} \longrightarrow^{\mathrm{x}*} s_{b_{21}}$. By Lemma 253, $t_{a_1} \simeq t_{b_{21}}$. We also have $t_{a_1} \longrightarrow^* t_{a_1}$.

Case ii. Let $t_{b_{11}}[x := w_{b_{11}}]$ be in the substitution normal form. Then, $s_{b_{21}} = t_{b_{11}}[x := w_{b_{11}}] = (\lambda x_0.t_{b_{111}})\overline{[x_i := w_{b_i}]}_{i=1}^{+n}$ where $x_n = x$ and $w_{b_n} = w_{b_{11}}$. This case is vacuous because $t_{b_1} \not\longrightarrow$.

By Propositions 59 and 84, we know $FV(t_{a_2}) = \emptyset$ and $FV(t_{b_2}) = \emptyset$. □

B.2.7 Soundness and Completeness

We demonstrate the soundness and completeness of Suspended ISWIM with respect to (Substitutional) ISWIM.

Theorem 255 (Soundness of Suspended ISWIM w.r.t. (Substitutional) ISWIM)**.** *If $t_{a_1} \simeq t_{b_1}$ and $t_{a_1} \longrightarrow^* v_{a_2}$ in (Substitutional) ISWIM where $FV(t_{a_1}) = \emptyset$ and $FV(t_{b_1}) = \emptyset$, then $t_{b_1} \longrightarrow^* v_{b_2}$ in Suspended ISWIM and $v_{a_2} \simeq v_{b_2}$ where $FV(v_{a_2}) = \emptyset$ and $FV(v_{b_2}) = \emptyset$.*

Proof. We proceed by induction on the length of $t_{a_1} \longrightarrow^* v_{a_2}$.

Case 1. (0). Let $t_{a_1} = v_{a_2}$. By Lemma 246, $t_{b_1} \longrightarrow^* v_{b_2}$ and $v_{a_2} \simeq v_{b_2}$. By Propositions 59 and 84, we know $FV(v_{a_2}) = \emptyset$ and $FV(v_{b_2}) = \emptyset$.

Case 2. $(n+1)$. Let $t_{a_1} \longrightarrow t_{a_2} \longrightarrow^{(n)} v_{a_2}$. Given $t_{a_1} \simeq t_{b_1}$ and $t_{a_1} \longrightarrow t_{a_2}$, by Lemma 251, $t_{b_1} \longrightarrow^* t_{b_2}$ and $t_{a_2} \simeq t_{b_2}$. Given $t_{a_2} \simeq t_{b_2}$ and $t_{a_2} \longrightarrow^{(n)} v_{a_2}$, by the induction hypothesis, $t_{b_2} \longrightarrow^* v_{b_2}$ and $v_{a_2} \simeq v_{b_2}$. We have $t_{b_1} \longrightarrow^* t_{b_2} \longrightarrow^* v_{b_2}$ and $v_{a_2} \simeq v_{b_2}$. By Propositions 59 and 84, we know $FV(v_{a_2}) = \emptyset$ and $FV(v_{b_2}) = \emptyset$.

□

Theorem 256 (Completeness of Suspended ISWIM w.r.t. (Substitutional) ISWIM)**.** *If $t_{a_1} \simeq t_{b_1}$ and $t_{b_1} \longrightarrow^* v_{b_2}$ in Suspended ISWIM where $FV(t_{a_1}) = \emptyset$ and $FV(t_{b_1}) = \emptyset$, then $t_{a_1} \longrightarrow^* v_{a_2}$ in (Substitutional) ISWIM and $v_{a_2} \simeq v_{b_2}$ where $FV(v_{a_2}) = \emptyset$ and $FV(v_{b_2}) = \emptyset$.*

Proof. We proceed by induction on the length of $t_{b_1} \longrightarrow^* v_{b_2}$.

Case 1. (0). Let $t_{b_1} = v_{b_2}$. By Lemma 245, $t_{a_1} \in \text{VALUE}_{\text{sub}}$. Let $v_{a_2} = t_{a_1}$. We have $t_{a_1} \longrightarrow^* v_{a_2}$ and $v_{a_2} \simeq v_{b_2}$. By Propositions 59 and 84, we know $FV(v_{a_2}) = \emptyset$ and $FV(v_{b_2}) = \emptyset$.

Case 2. $(n+1)$. Let $t_{b_1} \longrightarrow t_{b_2} \longrightarrow^{(n)} v_{b_2}$. Given $t_{a_1} \simeq t_{b_1}$ and $t_{b_1} \longrightarrow t_{b_2}$, by Lemma 254, $t_{a_1} \longrightarrow^* t_{a_2}$ and $t_{a_2} \simeq t_{b_2}$. Given $t_{a_2} \simeq t_{b_2}$ and $t_{b_2} \longrightarrow^{(n)} v_{b_2}$, by the induction hypothesis, $t_{a_2} \longrightarrow^* v_{a_2}$ and $v_{a_2} \simeq v_{b_2}$. We have $t_{a_1} \longrightarrow^* v_{a_2}$ and $v_{a_2} \simeq v_{b_2}$. By Propositions 59 and 84, we know $FV(v_{a_2}) = \emptyset$ and $FV(v_{b_2}) = \emptyset$.

□

B.2.8 Kleene Equality of Evaluators

We prove the Kleene equality of evaluators $eval_{\text{ISWIM:SubSOS}}(t)$ and $eval_{\text{ISWIM:SusSOS}}(t)$.

Theorem 257 (Kleene Equality of Evaluators)**.** *For any $t \in \text{PRGM}_{\text{ISWIM}}$, $eval_{\text{ISWIM:SubSOS}}(t)$ is Kleene equal to $eval_{\text{ISWIM:SusSOS}}(t)$.*

Proof. We first show if $eval_{\text{ISWIM:SubSOS}}(t) = a$ where $a \in \text{ANS}_{\text{ISWIM}}$, then $eval_{\text{ISWIM:SusSOS}}(t) = a$.

Case 1. If $eval_{\text{ISWIM:SubSOS}}(t) = \texttt{function}$, then $t \longrightarrow^* \lambda x.t'$. By Theorem 255, $t \longrightarrow^* v'$ and $\lambda x.t' \simeq v'$. Proceed by induction on $\lambda x.t' \sim v'$.

- *Case* i. (lam-sim). Then $v' = \lambda x.t''$ and $t' \simeq t''$. We have $eval_{\text{ISWIM:SusSOS}}(t) = \texttt{function}$.
- *Case* ii. (subst-sim). Then $v' = \lambda x.t''\overline{[x_i := w_i]}^{+}$. We have $eval_{\text{ISWIM:SusSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{ISWIM:SubSOS}}(t) = n$, then $t \longrightarrow^* n$. By Theorem 255, $t \longrightarrow^* v'$ and $n \simeq v'$. Proceed by induction on $n \sim v'$.

- *Case* i. (num-sim). Then $v' = n$. We have $eval_{\text{ISWIM:SusSOS}}(t) = n$.
- *Case* ii. (subst-sim). Then $v' = \lambda x.t''\overline{[x_i := w_{b_i}]}^{+}$. Given $n \simeq \lambda x.t''\overline{[x_i := w_{b_i}]}^{+}$, by Lemma 242, $n = (\lambda x.t''')\overline{[w_{a_i}/x_i]}^{+}$, $t''' \simeq t''$ and $w_{a_i} \simeq w_{b_i}$ for any i. This case is vacuous because $n = (\lambda x.t''')\overline{[w_{a_i}/x_i]}^{+}$ does not hold.

We then show if $eval_{\text{ISWIM:SusSOS}}(t) = a$ where $a \in \text{ANS}_{\text{ISWIM}}$, then $eval_{\text{ISWIM:SubSOS}}(t) = a$.

Case 1. If $eval_{\text{ISWIM:SusSOS}}(t) = \texttt{function}$, then $t \longrightarrow^* \lambda x.t'$. By Theorem 256, $t \longrightarrow^* v'$ and $v' \simeq \lambda x.t'$. Proceed by induction on $v' \simeq \lambda x.t'$. The only case is (lam-sim). Then $v' = \lambda x.t''$ and $t'' \simeq t'$. We have $eval_{\text{ISWIM:SubSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{ISWIM:SusSOS}}(t) = n$, then $t \longrightarrow^* n$. By Theorem 256, $t \longrightarrow^* v'$ and $v' \simeq n$. Proceed by induction on $v' \simeq n$. The only case is (num-sim). Then $v' = n$. We have $eval_{\text{ISWIM:SubSOS}}(t) = n$.

We observe that $eval_{\text{ISWIM:SubSOS}}(t)$ is undefined if and only if $eval_{\text{ISWIM:SusSOS}}(t)$ is undefined. Therefore, $eval_{\text{ISWIM:SubSOS}}(t)$ is Kleene equal to $eval_{\text{ISWIM:SusSOS}}(t)$. □

B.3 Equivalence of ISWIM and Environmental ISWIM

We demonstrate the equivalence of the substitutional structural operational semantics of ISWIM and the structural operational semantics of Environmental ISWIM. We use subscripts "$_{\text{sub}}$" and "$_{\text{env}}$" to differentiate the syntax of (Substitutional) ISWIM from the syntax of Environmental ISWIM.

B.3.1 Unload Function

We define $U(c)$ to unload an Environmental ISWIM configuration c to a (Substitutional) ISWIM term.

Definition 258 (Unload Function)**.** Define the unload function U to be a total function from the set of configurations in Environmental ISWIM to the set of terms in (Substitutional) ISWIM.

$$U \ : \ \textsc{Conf}_{\text{env}} \longrightarrow \textsc{Term}_{\text{sub}}$$

$$\begin{array}{rcl} U(n) & = & n \\ U(\triangleleft \lambda x.t,\ \rho \triangleright) & = & U(\langle \lambda x.t,\ \rho \rangle) \\ U(c_1\ c_2) & = & U(c_1)\,U(c_2) \\ U(c_1 + c_2) & = & U(c_1) + U(c_2) \\ U(\langle t,\ \{(x_1,w_1),(x_2,w_2),...,(x_n,w_n)\}\rangle) & = & t[U(w_1)/x_1][U(w_2)/x_2]...[U(w_n)/x_n] \end{array}$$

B.3.2 Bisimulation Relation

We introduce a bisimulation relation that relates (Substitutional) ISWIM terms to Environmental ISWIM configurations.

Definition 259 (Bisimulation Relation)**.** Define the bisimulation relation $\simeq$ to be a binary relation up to alpha equivalence between the set of terms in (Substitutional) ISWIM and the set of configurations in Environmental ISWIM.

$$\simeq \subseteq \textsc{Term}_{\text{sub}} \times \textsc{Conf}_{\text{env}}$$

$$t \simeq c \text{ if and only if } t = U(c)$$

B.3.3 Canonisation

If a term and a configuration are relationed and one of them is a value, then the other one is a value or multi-steps to a value. We have the following two lemmas.

Lemma 260 (Canonisation of (Substitutional) ISWIM)**.** *If $t_{a_1} \simeq v_{b_1}$, then $t_{a_1} \in \textsc{Value}_{\text{sub}}$.*

Proof. We proceed by cases on $v_{b_1} \in \textsc{Value}_{\text{env}}$.

Case 1. $(v_{b_1} = \triangleleft \lambda x.t_{b_{11}},\ \rho \triangleright)$. Then $t_{a_1} = U(\triangleleft \lambda x.t_{b_{11}},\ \rho \triangleright) = U(\langle \lambda x.t_{b_{11}},\ \rho \rangle)$. Suppose $\rho = \{(x_1,w_1),(x_2,w_2),...,(x_n,w_n)\}$. We have $t_{a_1} = (\lambda x.t_{b_{11}})[U(w_1)/x_1][U(w_2)/x_2]...[U(w_n)/x_n] \in \textsc{Value}_{\text{sub}}$.

Case 2. $(v_{b_1} = n)$. Then $t_{a_1} = U(n) = n$ and $n \in \textsc{Value}_{\text{sub}}$.

□

Lemma 261 (Canonisation of Environmental ISWIM)**.** *If $v_{a_1} \simeq c_{b_1}$, then $c_{b_1} \longrightarrow^* v_{b_2}$ and $v_{a_1} \simeq v_{b_2}$.*

Proof. We proceed by cases on $v_{a_1} \in \text{VALUE}_{\text{sub}}$ and by cases on $c_{b_1} \in \text{CONF}_{\text{env}}$. The only possible cases are the following ones.

Case 1. $(v_{a_1} = \lambda x.t_{a_{11}})$.

Case i. $(c_{b_1} = \lhd\lambda x_0.t_{b_{11}},\ \rho\rhd$ where $FV(\lambda x_0.t_{b_{11}}) \subseteq \text{dom}(\rho))$. Then $\lhd\lambda x_0.t_{b_{11}},\ \rho\rhd \in \text{VALUE}_{\text{env}}$.

Case ii. $(c_{b_1} = \langle x_0,\ \rho\rangle$ where $FV(x_0) \subseteq \text{dom}(\rho))$. Given $\lambda x.t_{a_{11}} \simeq \langle x_0,\ \rho\rangle$, we have $\lambda x.t_{a_{11}} = U(\langle x_0,\ \rho\rangle) = x_0\overline{[U(w_i)/x_i]} = U(w_{\text{p}})$ where $x_0 \equiv x_{\text{p}}$. Then by (var-env), $\langle x_0,\ \rho\rangle \longrightarrow w_{\text{p}}$ where $\rho(x_0) = w_{\text{p}}$. We have $\lambda x.t_{a_{11}} \simeq w_{\text{p}}$.

Case iii. $(c_{b_1} = \langle\lambda x_0.t_{b_{11}},\ \rho\rangle$ where $FV(\lambda x_0.t_{b_{11}}) \subseteq \text{dom}(\rho))$. Then by (clos-env), $\langle\lambda x_0.t_{b_{11}},\ \rho\rangle \longrightarrow \lhd\lambda x_0.t_{b_{11}},\ \rho\rhd$ and $\lhd\lambda x_0.t_{b_{11}},\ \rho\rhd \in \text{VALUE}_{\text{env}}$.

Case 2. $(v_{a_1} = n)$.

Case i. $(c_{b_1} = n)$. Then $c_{b_1} \in \text{VALUE}_{\text{env}}$.

Case ii. $(c_{b_1} = \langle x_0,\ \rho\rangle$ where $FV(x_0) \subseteq \text{dom}(\rho))$. Given $n \simeq \langle x_0,\ \rho\rangle$, we have $n = U(\langle x_0,\ \rho\rangle) = x_0\overline{[U(w_i)/x_i]} = U(w_{\text{p}})$ where $x_0 \equiv x_{\text{p}}$. Then by (var-env), $\langle x_0,\ \rho\rangle \longrightarrow w_{\text{p}}$ where $\rho(x_0) = w_{\text{p}}$. We have $n \simeq w_{\text{p}}$.

Case iii. $(c_{b_1} = \langle n_0,\ \rho\rangle)$. Given $n \simeq \langle n_0,\ \rho\rangle$, we have $n = U(\langle n_0,\ \rho\rangle) = n_0\overline{[U(w_i)/x_i]} = n_0$. Then by (num-env), $\langle n_0,\ \rho\rangle \longrightarrow n_0$. We have $n \simeq n_0$.

□

B.3.4 Bisimulation

We demonstrate (Substitutional) ISWIM bisimulates Environmental ISWIM. Intuitively, if a term relates to a configuration and one of them single-steps, then the other one multi-steps, and their results are related. .

Lemma 262 (Simulation: (Substitutional) ISWIM simulates Environmental ISWIM.)**.** *If $t_{a_1} \simeq c_{b_1}$ and $c_{b_1} \longrightarrow c_{b_2}$ where $FV(t_{a_1}) = \emptyset$ and $FV(c_{b_1}) = \emptyset$, then $t_{a_1} \longrightarrow^* t_{a_2}$ and $t_{a_2} \simeq c_{b_2}$ where $FV(t_{a_2}) = \emptyset$ and $FV(c_{b_2}) = \emptyset$.*

Proof. We proceed by induction on the structure of $c_{b_1} \in \text{CONF}_{\text{env}}$.

Case 1. $(c_{b_1} = v_{b_1})$. This case is vacuous because $c_{b_1} \not\longrightarrow$.

Case 2. $(c_{b_1} = c_{b_{11}}\ c_{b_{12}})$. Then $t_{a_1} = t_{a_{11}}\ t_{a_{12}}$ where $t_{a_{11}} = U(c_{b_{11}})$ and $t_{a_{12}} = U(c_{b_{12}})$. We proceed by cases on $c_{b_1} \longrightarrow c_{b_2}$.

Case i. (appL). Then $c_{b_{11}}\ c_{b_{12}} \longrightarrow c_{b_{21}}\ c_{b_{12}}$ where $c_{b_{11}} \longrightarrow c_{b_{21}}$. By the induction hypothesis, $t_{a_{11}} \longrightarrow^* t_{a_{21}}$ and $t_{a_{21}} \simeq c_{b_{21}}$. Then $t_{a_{11}}\ t_{a_{12}} \longrightarrow^* t_{a_{21}}\ t_{a_{12}}$ and $t_{a_{21}}\ t_{a_{12}} \simeq c_{b_{21}}\ c_{b_{12}}$.

Case ii. (appR). Then $c_{b_{11}} = v_{b_{11}}$ and $v_{b_{11}}\ c_{b_{12}} \longrightarrow v_{b_{11}}\ c_{b_{22}}$ where $c_{b_{12}} \longrightarrow c_{b_{22}}$. By Lemma 260, $t_{a_{11}} = v_{a_{11}}$. By the induction hypothesis, $t_{a_{12}} \longrightarrow^* t_{a_{22}}$ and $t_{a_{22}} \simeq c_{b_{22}}$. Then $v_{a_{11}}\ t_{a_{12}} \longrightarrow^* v_{a_{11}}\ t_{a_{22}}$ and $v_{a_{11}}\ t_{a_{12}} \simeq v_{b_{11}}\ c_{b_{22}}$.

Case iii. (app). Then $c_{b_{11}} = \triangleleft\lambda x.t_{b_{11}},\ \rho\triangleright$, $c_{b_{12}} = v_{b_{12}}$ and $\triangleleft\lambda x.t_{b_{11}},\ \rho \triangleright\ v_{b_{12}} \longrightarrow \langle t_{b_{11}},\ \rho[x \mapsto v_{b_{12}}]\rangle$. Suppose $\rho = \{(x_1, w_1), (x_2, w_2), ..., (x_n, w_n)\}$. We have $t_{a_{11}} = U(\triangleleft\lambda x.t_{b_{11}},\ \rho\triangleright) = U(\langle\lambda x.t_{b_{11}},\ \rho\rangle = (\lambda x.t_{b_{11}})\overline{[U(w_i)/x_i]}_{i=1}^{n}$ and $(\lambda x.t_{b_{11}})\overline{[U(w_i)/x_i]}_{i=1}^{n} =_\alpha \lambda x_0.t_{b_{11}}[x_0/x]\overline{[U(w_i)/x_i]}_{i=1}^{n}$ where $x_0 \notin FV(\lambda x.t_{b_{11}}) \cup (\bigcup_i (FV(U(w_i)) \cup x_i))$. By Lemma 260, we know $t_{a_{12}} = v_{a_{12}}$. Then we have $(\lambda x_0.t_{b_{11}}[x_0/x]\overline{[U(w_i)/x_i]}_{i=1}^{n})\ v_{a_{12}} \longrightarrow t_{b_{11}}[x_0/x]\overline{[U(w_i)/x_i]}_{i=1}^{n}[v_{a_{12}}/x_0]$. Given $FV(t_{a_1}) = \emptyset$, we know $FV(t_{a_{11}}) = FV(\lambda x_0.t_{b_{11}}[x_0/x]\overline{[U(w_i)/x_i]}_{i=1}^{n}) = \emptyset$ and $FV(t_{a_{12}}) = FV(v_{a_{12}}) = \emptyset$. By Proposition 53, $t_{b_{11}}[x_0/x]\overline{[U(w_i)/x_i]}_{i=1}^{n}[v_{a_{12}}/x_0] =_\alpha t_{b_{11}}[v_{a_{12}}/x]\overline{[U(w_i)/x_i]}_{i=1}^{n}$. Then $U(\langle t_{b_{11}},\ \rho[x \mapsto v_{b_{12}}]\rangle) = t_{b_{11}}[U(v_{b_{12}})/x]\overline{[U(w_i)/x_i]}_{i=1}^{n} = t_{b_{11}}[v_{a_{12}}/x]\overline{[U(w_i)/x_i]}_{i=1}^{n}$. Hence $t_{b_{11}}[v_{a_{12}}/x]\overline{[U(w_i)/x_i]}_{i=1}^{n} \simeq \langle t_{b_{11}},\ \rho[x \mapsto v_{b_{12}}]\rangle$.

Case 3. ($c_{b_1} = c_{b_{11}} + c_{b_{12}}$). Then $t_{a_1} = t_{a_{11}} + t_{a_{12}}$ where $t_{a_{11}} = U(c_{b_{11}})$ and $t_{a_{12}} = U(c_{b_{12}})$. We proceed by cases on $c_{b_1} \longrightarrow c_{b_2}$.

Case i. (plusL). Then $c_{b_{11}} + c_{b_{12}} \longrightarrow c_{b_{21}} + c_{b_{12}}$ where $c_{b_{11}} \longrightarrow c_{b_{21}}$. By the induction hypothesis, $t_{a_{11}} \longrightarrow^* t_{a_{21}}$ and $t_{a_{21}} \simeq c_{b_{21}}$. Then $t_{a_{11}} + t_{a_{12}} \longrightarrow^* t_{a_{21}} + t_{a_{12}}$ and $t_{a_{21}} + t_{a_{12}} \simeq c_{b_{21}} + c_{b_{12}}$.

Case ii. (plusR). Then $c_{b_{11}} = v_{b_{11}}$ and $v_{b_{11}} + c_{b_{12}} \longrightarrow v_{b_{11}} + c_{b_{22}}$ where $c_{b_{12}} \longrightarrow c_{b_{22}}$. By Lemma 260, $t_{a_{11}} = v_{a_{11}}$. By the induction hypothesis, $t_{a_{12}} \longrightarrow^* t_{a_{22}}$ and $t_{a_{22}} \simeq c_{b_{22}}$. Then $v_{a_{11}} + t_{a_{12}} \longrightarrow^* v_{a_{11}} + t_{a_{22}}$ and $v_{a_{11}} + t_{a_{12}} \simeq v_{b_{11}} + c_{b_{22}}$.

Case iii. (plus). Then $c_{b_{11}} = n_1$, $c_{b_{12}} = n_2$, and $n_1 + n_2 \longrightarrow n$ where $n = n_1 + n_2$. We have $t_{a_{11}} = U(c_{b_{11}}) = n_1$, $t_{a_{12}} = U(c_{b_{12}}) = n_2$, and $n_1 + n_2 \longrightarrow n$ where $n = n_1 + n_2$. Then $n \simeq n$.

Case 4. ($c_{b_1} = \langle t_{b_1},\ \rho\rangle$ where $FV(t_{b_1}) \subseteq \mathrm{dom}(\rho)$). Suppose $\rho = \{(x_1, w_1), (x_2, w_2), ..., (x_n, w_n)\}$. Then $t_{a_1} = t_{b_1}\overline{[U(w_i)/x_i]}$. We proceed by cases on $c_{b_1} \longrightarrow c_{b_2}$.

Case i. (clos-env). Then $c_{b_1} = \langle(\lambda x.t_{b_{11}}),\ \rho\rangle$ and $c_{b_2} = \triangleleft(\lambda x.t_{b_{11}}),\ \rho\triangleright$. We have $t_{a_1} = U(\langle(\lambda x.t_{b_{11}}),\ \rho\rangle)$ and $t_{a_2} = U(\triangleleft(\lambda x.t_{b_{11}}),\ \rho\triangleright) = U(\langle(\lambda x.t_{b_{11}}),\ \rho\rangle)$. Thus $t_{a_1} \longrightarrow^* t_{a_2}$ and $t_{a_2} \simeq c_{b_2}$.

Case ii. (var-env). Then $c_{b_1} = \langle x,\ \rho\rangle$ and $c_{b_2} = w$ where $\rho(x) = w$. We have $t_{a_1} = U(\langle x,\ \rho\rangle) = x\overline{[U(w_i)/x_i]} = U(w)$ and $t_{a_2} = U(w)$. Thus $t_{a_1} \longrightarrow^* t_{a_2}$ and $t_{a_2} \simeq c_{b_2}$.

Case iii. (num-env). Then $c_{b_1} = \langle n,\ \rho\rangle$ and $c_{b_2} = n$. We have $t_{a_1} = U(\langle n,\ \rho\rangle) = n$ and $t_{a_2} = U(n) = n$. Thus $t_{a_1} \longrightarrow^* t_{a_2}$ and $t_{a_2} \simeq c_{b_2}$.

Case iv. (app-env). Then $c_{b_1} = \langle(t_1\ t_2),\ \rho\rangle$ and $c_{b_2} = \langle t_1,\ \rho\rangle\ \langle t_2,\ \rho\rangle$. We have $t_{a_1} = U(\langle(t_1\ t_2),\ \rho\rangle) = (t_1\ t_2)\overline{[U(w_i)/x_i]} = (t_1\overline{[U(w_i)/x_i]})\ (t_2\overline{[U(w_i)/x_i]}) = U(\langle t_1,\ \rho\rangle)\ U(\langle t_2,\ \rho\rangle) = U(\langle t_1,\ \rho\rangle\ \langle t_2,\ \rho\rangle) = t_{a_2}$. Thus $t_{a_1} \longrightarrow^* t_{a_2}$ and $t_{a_2} \simeq c_{b_2}$.

Case v. (plus-env). Then $c_{b_1} = \langle (t_1 + t_2),\, \rho \rangle$ and $c_{b_2} = \langle t_1,\, \rho \rangle + \langle t_2,\, \rho \rangle$. We have $t_{a_1} = U(\langle (t_1 + t_2),\, \rho \rangle) = (t_1 + t_2)\overline{[U(w_i)/x_i]} = (t_1\overline{[U(w_i)/x_i]}) + (t_2\overline{[U(w_i)/x_i]}) = U(\langle t_1,\, \rho \rangle) + U(\langle t_2,\, \rho \rangle) = U(\langle t_1,\, \rho \rangle + \langle t_2,\, \rho \rangle) = t_{a_2}$. Thus $t_{a_1} \longrightarrow^* t_{a_2}$ and $t_{a_2} \simeq c_{b_2}$.

By Propositions 59 and 94, we know $FV(t_{a_2}) = \emptyset$ and $FV(t_{b_2}) = \emptyset$. □

Lemma 263 (Simulation: Environmental ISWIM simulates (Substitutional) ISWIM.)**.** *If $t_{a_1} \simeq c_{b_1}$ and $t_{a_1} \longrightarrow t_{a_2}$ where $FV(t_{a_1}) = \emptyset$ and $FV(c_{b_1}) = \emptyset$, then $c_{b_1} \longrightarrow^* c_{b_2}$ and $t_{a_2} \simeq c_{b_2}$ where $FV(t_{a_2}) = \emptyset$ and $FV(c_{b_2}) = \emptyset$.*

Proof. We proceed by induction on the structure of $t_{a_1} \in \text{TERM}_{\text{sub}}$.

Case 1. ($t_{a_1} = x$). This case is vacuous because $x \not\longrightarrow$.

Case 2. ($t_{a_1} = t_{a_{11}}\ t_{a_{12}}$). Let $c_{b_1} = c_{b_{11}}\ c_{b_{12}}$, $t_{a_{11}} = U(c_{b_{11}})$ and $t_{a_{12}} = U(c_{b_{12}})$. We proceed by cases on $t_{a_1} \longrightarrow t_{a_2}$.

Case i. (appL). Then $t_{a_{11}}\ t_{a_{12}} \longrightarrow t_{a_{21}}\ t_{a_{12}}$ where $t_{a_{11}} \longrightarrow t_{a_{21}}$. By the induction hypothesis, $c_{b_{11}} \longrightarrow c_{b_{21}}$ and $t_{a_{21}} \simeq c_{b_{21}}$. Then $c_{b_{11}}\ c_{b_{12}} \longrightarrow c_{b_{21}}\ c_{b_{12}}$ and $t_{a_{21}}\ t_{a_{12}} \simeq c_{b_{21}}\ c_{b_{12}}$.

Case ii. (appR). Then $t_{a_{11}} = v_{a_{11}}$ and $v_{a_{11}}\ t_{a_{12}} \longrightarrow v_{a_{11}}\ t_{a_{22}}$ where $t_{a_{12}} \longrightarrow t_{a_{22}}$. By Lemma 261, $c_{b_{11}} \longrightarrow^* v_{b_{11}}$. By the induction hypothesis, $c_{b_{12}} \longrightarrow c_{b_{22}}$ and $t_{a_{22}} \simeq c_{b_{22}}$. Then $c_{b_{11}}\ c_{b_{12}} \longrightarrow^* v_{b_{11}}\ c_{b_{12}} \longrightarrow v_{b_{11}}\ c_{b_{12}}$ and $t_{a_{11}}\ t_{a_{22}} \simeq c_{b_{11}}\ c_{b_{22}}$.

Case iii. (app). Then $t_{a_{11}} = \lambda x.t_{a_{111}}$, $t_{a_{12}} = v_{a_{12}}$ and $(\lambda x.t_{a_{111}})\ v_{a_{12}} \longrightarrow t_{a_{111}}[v_{a_{12}}/x]$. Given $\lambda x.t_{a_{111}} \simeq c_{b_{11}}$, by Lemma 261, $c_{b_{11}} \longrightarrow^* v_{b_{11}}$ and $\lambda x.t_{a_{111}} \simeq v_{b_{11}}$. Given $v_{a_{12}} \simeq c_{b_{12}}$, by Lemma 261, $c_{b_{12}} \longrightarrow^* v_{b_{12}}$ and $v_{a_{12}} \simeq v_{b_{12}}$. We proceed by cases on $v_{b_{11}}$ in $\lambda x.t_{a_{111}} \simeq v_{b_{11}}$. The only possible case is $v_{b_{11}} = \triangleleft \lambda x_0.t_{b_{111}},\, \rho \triangleright$ where $FV(\lambda x_0.t_{b_{111}}) \in \text{dom}(\rho)$ and $U(\triangleleft \lambda x_0.t_{b_{111}},\, \rho \triangleright) = \lambda x.t_{a_{111}}$.
We have $\triangleleft \lambda x_0.t_{b_{111}},\, \rho \triangleright\ v_{b_{12}} \longrightarrow \langle t_{b_{111}},\, \rho[x_0 \mapsto v_{b_{12}}] \rangle$ and we need to show $t_{a_{111}}[v_{a_{12}}/x] \simeq \langle t_{b_{111}},\, \rho[x_0 \mapsto v_{b_{12}}] \rangle$.
Suppose $\rho = \{(x_1, w_1), (x_2, w_2), ..., (x_n, w_n)\}$. Given $\lambda x.t_{a_{111}} \simeq \triangleleft \lambda x_0.t_{b_{111}},\, \rho \triangleright$,
we have $U(\triangleleft \lambda x_0.t_{b_{111}},\, \rho \triangleright) = U(\langle \lambda x_0.t_{b_{111}},\, \rho \rangle) = (\lambda x_0.t_{b_{111}})\overline{[U(w_i)/x_i]}_{i=1}^{n} = \lambda x.t_{a_{111}}$.
Then we get $t_{b_{111}}[x_{-1}/x_0]\overline{[U(w_i)/x_i]}_{i=1}^{n}[v_{a_{12}}/x_{-1}] = t_{a_{111}}[v_{a_{12}}/x]$ where $x_{-1} \notin FV(\lambda x_0.t_{b_{111}}) \cup (\bigcup_i (FV(U(w_i)) \cup x_i))$.
Given $FV(t_{a_1}) = \emptyset$, we know $FV(t_{a_{11}}) = \lambda x.t_{a_{111}} = \emptyset$ and $FV(t_{a_{12}}) = FV(v_{a_{12}}) = \emptyset$.
By Proposition 53, $t_{b_{111}}[x_{-1}/x_0]\overline{[U(w_i)/x_i]}_{i=1}^{n}[v_{a_{12}}/x_{-1}] \sim_\alpha t_{b_{111}}[v_{a_{12}}/x_0]\overline{[U(w_i)/x_i]}_{i=1}^{n}$.
Then $t_{b_{111}}[U(v_{b_{12}})/x_0]\overline{[U(w_i)/x_i]}_{i=1}^{n} = t_{b_{111}}[v_{a_{12}}/x_0]\overline{[U(w_i)/x_i]}_{i=1}^{n}$.
Hence, $t_{a_{111}}[v_{a_{12}}/x] \simeq \langle t_{b_{111}},\, \rho[x_0 \mapsto v_{b_{12}}] \rangle$.

Case 3. ($t_{a_1} = \lambda x.t_{a_{11}}$). This case is vacuous because $\lambda x.t_{a_{11}} \not\longrightarrow$.

Case 4. ($t_{a_1} = n$). This case is vacuous because $n \not\longrightarrow$.

Case 5. $(t_{a_1} = t_{a_{11}} + t_{a_{12}})$. Let $c_{b_1} = c_{b_{11}} + c_{b_{12}}$, $t_{a_{11}} = U(c_{b_{11}})$ and $t_{a_{12}} = U(c_{b_{12}})$. We proceed by cases on $t_{a_1} \longrightarrow t_{a_2}$.

Case i. (appL). Then $t_{a_{11}} + t_{a_{12}} \longrightarrow t_{a_{21}} + t_{a_{12}}$ where $t_{a_{11}} \longrightarrow t_{a_{21}}$. By the induction hypothesis, $c_{b_{11}} \longrightarrow c_{b_{21}}$ and $t_{a_{21}} \simeq c_{b_{21}}$. Then $c_{b_{11}} + c_{b_{12}} \longrightarrow c_{b_{21}} + c_{b_{12}}$ and $t_{a_{21}} + t_{a_{12}} \simeq c_{b_{21}} + c_{b_{12}}$.

Case ii. (appR). Then $t_{a_{11}} = v_{a_{11}}$ and $v_{a_{11}} + t_{a_{12}} \longrightarrow v_{a_{11}} + t_{a_{22}}$ where $t_{a_{12}} \longrightarrow t_{a_{22}}$. By Lemma 261, $c_{b_{11}} \longrightarrow^* v_{b_{11}}$. By the induction hypothesis, $c_{b_{12}} \longrightarrow c_{b_{22}}$ and $t_{a_{22}} \simeq c_{b_{22}}$. Then $v_{b_{11}} + c_{b_{12}} \longrightarrow v_{b_{11}} + c_{b_{12}}$ and $t_{a_{11}} + t_{a_{22}} \simeq c_{b_{11}} + c_{b_{22}}$.

Case iii. (plus). Then $t_{a_{11}} = n_1$, $t_{a_{12}} = n_2$ and $n_1 + n_2 \longrightarrow n$ where $n = n_1 + n_2$. Given $n_1 \simeq c_{b_{11}}$, by Lemma 261, $c_{b_{11}} \longrightarrow^* v_{b_{11}}$ and $n_1 \simeq v_{b_{11}}$. Given $n_2 \simeq c_{b_{12}}$, by Lemma 261, $c_{b_{12}} \longrightarrow^* v_{b_{12}}$ and $n_2 \simeq v_{b_{12}}$. We proceed by cases on $v_{b_{11}}$ in $n_1 \simeq v_{b_{11}}$ and on $v_{b_{12}}$ in $n_2 \simeq v_{b_{12}}$. The only case is $v_{b_{11}} = n_1$ and $v_{b_{12}} = n_2$. Then $n_1 + n_2 \longrightarrow n$ where $n = n_1 + n_2$ and $n \simeq n$.

By Propositions 59 and 94, we know $FV(t_{a_2}) = \emptyset$ and $FV(t_{b_2}) = \emptyset$. □

B.3.5 Soundness and Completeness

We demonstrate the soundness and completeness of Environmental ISWIM with respect to (Substitutional) ISWIM.

Theorem 264 (Soundness of Environmental ISWIM w.r.t. (Substitutional) ISWIM)**.** *If $t_{a_1} \simeq c_{b_1}$ and $t_{a_1} \longrightarrow^* v_{a_2}$ in (Substitutional) ISWIM where $FV(t_{a_1}) = \emptyset$ and $FV(c_{b_1}) = \emptyset$, then $c_{b_1} \longrightarrow^* v_{b_2}$ in Environmental ISWIM and $v_{a_2} \simeq v_{b_2}$ where $FV(v_{a_2}) = \emptyset$ and $FV(v_{b_2}) = \emptyset$.*

Proof. We proceed by induction on the length of $t_{a_1} \longrightarrow^* v_{a_2}$.

Case 1. (0). Let $t_{a_1} = v_{a_2}$. By Lemma 261, $c_{b_1} \longrightarrow^* v_{b_2}$ and $v_{a_2} \simeq v_{b_2}$. By Propositions 59 and 94, we know $FV(v_{a_2}) = \emptyset$ and $FV(v_{b_2}) = \emptyset$.

Case 2. $(n+1)$. Let $t_{a_1} \longrightarrow t_{a_2} \longrightarrow^{(n)} v_{a_2}$. Given $t_{a_1} \simeq c_{b_1}$ and $t_{a_1} \longrightarrow t_{a_2}$, by Lemma 263, $c_{b_1} \longrightarrow^* c_{b_2}$ and $t_{a_2} \simeq c_{b_2}$. Given $t_{a_2} \simeq c_{b_2}$ and $t_{a_2} \longrightarrow^{(n)} v_{a_2}$, by the induction hypothesis, $c_{b_2} \longrightarrow^* v_{b_2}$ and $v_{a_2} \simeq v_{b_2}$. We have $c_{b_1} \longrightarrow^* c_{b_2} \longrightarrow^* v_{b_2}$ and $v_{a_2} \simeq v_{b_2}$. By Propositions 59 and 94, we know $FV(v_{a_2}) = \emptyset$ and $FV(v_{b_2}) = \emptyset$.

□

Remark 265. The parenthesised superscripted number n in $\longrightarrow^{(n)}$ denotes the number of single step is n.

Theorem 266 (Completeness of Environmental ISWIM w.r.t. (Substitutional) ISWIM)**.** *If $t_{a_1} \simeq c_{b_1}$ and $c_{b_1} \longrightarrow^* v_{b_2}$ in Environmental ISWIM where $FV(t_{a_1}) = \emptyset$ and $FV(c_{b_1}) = \emptyset$, then $t_{a_1} \longrightarrow^* v_{a_2}$ in (Substitutional) ISWIM and $v_{a_2} \simeq v_{b_2}$ where $FV(v_{a_2}) = \emptyset$ and $FV(v_{b_2}) = \emptyset$.*

Proof. We proceed by induction on the length of $c_{b_1} \longrightarrow^* v_{b_2}$.

Case 1. (0). Let $c_{b_1} = v_{b_2}$. By Lemma 260, $t_{a_1} \in \text{VALUE}_{\text{sub}}$. Let $t_{a_1} = v_{a_2}$. We have $t_{a_1} \longrightarrow^* v_{a_2}$ and $v_{a_2} \simeq v_{b_2}$. By Propositions 59 and 94, we know $FV(v_{a_2}) = \emptyset$ and $FV(v_{b_2}) = \emptyset$.

Case 2. $(n+1)$. Let $c_{b_1} \longrightarrow c_{b_2} \longrightarrow^{(n)} v_{b_2}$. Given $t_{a_1} \simeq c_{b_1}$ and $c_{b_1} \longrightarrow c_{b_2}$, by Lemma 262, $t_{a_1} \longrightarrow^* t_{a_2}$ and $t_{a_2} \simeq c_{b_2}$. Given $t_{a_2} \simeq c_{b_2}$ and $c_{b_2} \longrightarrow^{(n)} v_{b_2}$, by the induction hypothesis, $t_{a_2} \longrightarrow^* v_{a_2}$ and $v_{a_2} \simeq v_{b_2}$. We have $t_{a_1} \longrightarrow^* v_{a_2}$ and $v_{a_2} \simeq v_{b_2}$. By Propositions 59 and 94, we know $FV(v_{a_2}) = \emptyset$ and $FV(v_{b_2}) = \emptyset$.

□

B.3.6 Kleene Equality of Evaluators

We prove the Kleene equality of evaluators $eval_{\text{ISWIM:SubSOS}}(t)$ and $eval_{\text{ISWIM:EnvSOS}}(t)$.

Theorem 267 (Kleene Equality of Evaluators)**.** *For any $t \in \text{PRGM}_{\text{ISWIM}}$, $eval_{\text{ISWIM:SubSOS}}(t)$ is Kleene equal to $eval_{\text{ISWIM:EnvSOS}}(t)$.*

Proof. We first show if $eval_{\text{ISWIM:SubSOS}}(t) = a$ where $a \in \text{ANS}_{\text{ISWIM}}$, then $eval_{\text{ISWIM:EnvSOS}}(t) = a$.

Case 1. If $eval_{\text{ISWIM:SubSOS}}(t) = \texttt{function}$, then $t \longrightarrow^* \lambda x.t'$. By Theorem 264, $\langle t,\, \emptyset\rangle \longrightarrow^* v'$ and $\lambda x.t' \simeq v'$. Proceed by cases on v' in $\lambda x.t' \simeq v'$. The only case is $v' = \lhd\lambda x_0.t''$, $\rho\rhd$ and $U(\lhd\lambda x_0.t'',\, \rho\rhd) = \lambda x.t'$. We have $eval_{\text{ISWIM:EnvSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{ISWIM:SubSOS}}(t) = n$, then $t \longrightarrow^* n$. By Theorem 264, $\langle t,\, \emptyset\rangle \longrightarrow^* v'$ and $n \simeq v'$. Proceed by cases on v' in $n \simeq v'$. The only case is $v' = n_0$ and $n_0 = U(n_0) = n$. We have $eval_{\text{ISWIM:EnvSOS}}(t) = n$.

We then show if $eval_{\text{ISWIM:EnvSOS}}(t) = a$ where $a \in \text{ANS}_{\text{ISWIM}}$, then $eval_{\text{ISWIM:SubSOS}}(t) = a$.

Case 1. If $eval_{\text{ISWIM:EnvSOS}}(t) = \texttt{function}$, then $\langle t,\, \emptyset\rangle \longrightarrow^* \lhd\lambda x.t',\, \rho\rhd$. By Theorem 266, $t \longrightarrow^* v'$ and $v' \simeq \lhd\lambda x.t',\, \rho\rhd$. Proceed by cases on v' in $v' \simeq \lhd\lambda x.t',\, \rho\rhd$. The only case is $v' = \lambda x.t''$ and $U(\lhd\lambda x.t',\, \rho\rhd) = \lambda x.t''$. We have $eval_{\text{ISWIM:SubSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{ISWIM:EnvSOS}}(t) = n$, then $\langle t,\, \emptyset\rangle \longrightarrow^* n$. By Theorem 266, $t \longrightarrow^* v'$ and $v' \sim n$. Proceed by cases on v' in $v' \simeq n$. The only case is $v' = n_0$ and $n_0 = U(n_0) = n$. We have $eval_{\text{ISWIM:SubSOS}}(t) = n$.

We observe that $eval_{\text{ISWIM:SubSOS}}(t)$ is undefined if and only if $eval_{\text{ISWIM:EnvSOS}}(t)$ is undefined. Therefore, $eval_{\text{ISWIM:SubSOS}}(t)$ is Kleene equal to $eval_{\text{ISWIM:EnvSOS}}(t)$. □

B.4 Equivalence of Structural Operational Semantics and Reduction Semantics of Environmental ISWIM

We demonstrate the equivalence of the structural operational semantics of Environmental ISWIM and reduction semantics of Environmental ISWIM.

Lemma 268. *If $c_1 \longrightarrow c_2$ and $E \in \text{ECXT}$, then $E[c_1] \longmapsto E[c_2]$.*

Proof. Suppose $c_1 \longrightarrow c_2$ and $E \in \text{ECXT}$. We show that there exists some $E_0 \in \text{ECXT}$ such that $E[c_1] = E_0[c_{01}]$ and $E[c_2] = E_0[c_{02}]$ where $c_{01} \longrightarrow c_{02}$. We proceed by induction on the structure of the derivation of $c_1 \longrightarrow c_2$.

Case 1. (appL). Let $c_1 = c_{11}\ c_{12}$, $c_2 = c_{21}\ c_{12}$ and $c_{11} \longrightarrow c_{21}$. Let $E_0 = E[\Box\ c_{12}]$. By the induction hypothesis, $E_0[c_{11}] \longmapsto E_0[c_{21}]$. Thus $E[c_{11}\ c_{12}] \longmapsto E[c_{21}\ c_{12}]$.

Case 2. (appR). Let $c_1 = v_{11}\ c_{12}$, $c_2 = v_{11}\ c_{22}$ and $c_{12} \longrightarrow c_{22}$. Let $E_0 = E[v_{11}\ \Box]$. By the induction hypothesis, $E_0[c_{12}] \longmapsto E_0[c_{22}]$. Thus $E[v_{11}\ c_{12}] \longmapsto E[v_{11}\ c_{22}]$.

Case 3. (app). Let $c_1 = \triangleleft(\lambda x.t_{11}),\ \rho \triangleright\ v_{12}$ and $c_2 = \langle t_{11},\ \rho[x \mapsto v_{12}]\rangle$. We have $E[c_1] \longmapsto E[c_2]$ because $c_1\ \mathscr{R}\ c_2$.

Case 4. (plusL). Let $c_1 = c_{11} + c_{12}$, $c_2 = c_{21} + c_{12}$ and $c_{11} \longrightarrow c_{21}$. Let $E_0 = E[\Box + c_{12}]$. By the induction hypothesis, $E_0[c_{11}] \longmapsto E_0[c_{21}]$. Thus $E[c_{11} + c_{12}] \longmapsto E[c_{21} + c_{12}]$.

Case 5. (plusR). Let $c_1 = v_{11} + c_{12}$, $c_2 = v_{11} + c_{22}$ and $c_{12} \longrightarrow c_{22}$. Let $E_0 = E[v_{11} + \Box]$. By the induction hypothesis, $E_0[c_{12}] \longmapsto E_0[c_{22}]$. Thus $E[v_{11} + c_{12}] \longmapsto E[v_{11} + c_{22}]$.

Case 6. (plus). Let $c_1 = n_1 + n_2$, $c_2 = n$ and $n = n_1 + n_2$. We have $E[c_1] \longmapsto E[c_2]$ because $c_1\ \mathscr{R}\ c_2$.

Case 7. (clos-env). Let $c_1 = \langle(\lambda x.t),\ \rho\rangle$ and $c_2 = \triangleleft(\lambda x.t),\ \rho\triangleright$. We have $E[c_1] \longmapsto E[c_2]$ because $c_1\ \mathscr{R}\ c_2$.

Case 8. (var-env). Let $c_1 = \langle x,\ \rho\rangle$, $c_2 = w$ and $\rho(x) = w$. We have $E[c_1] \longmapsto E[c_2]$ because $c_1\ \mathscr{R}\ c_2$.

Case 9. (num-env). Let $c_1 = \langle n,\ \rho\rangle$ and $c_2 = n$. We have $E[c_1] \longmapsto E[c_2]$ because $c_1\ \mathscr{R}\ c_2$.

Case 10. (app-env). Let $c_1 = \langle(t_1\ t_2),\ \rho\rangle$ and $c_2 = \langle t_1,\ \rho\rangle\ \langle t_2,\ \rho\rangle$. We have $E[c_1] \longmapsto E[c_2]$ because $c_1\ \mathscr{R}\ c_2$.

Case 11. (plus-env). Let $c_1 = \langle(t_1 + t_2),\ \rho\rangle$ and $c_2 = \langle t_1,\ \rho\rangle + \langle t_2,\ \rho\rangle$. We have $E[c_1] \longmapsto E[c_2]$ because $c_1\ \mathscr{R}\ c_2$.

□

Corollary 269. *If $c_1 \longrightarrow c_2$, then $c_1 \longmapsto c_2$.*

Proof. Suppose $c_1 \longrightarrow c_2$. Let $E = \Box$ in Lemma 268. We get $\Box[c_1] \longmapsto \Box[c_2]$. Hence $c_1 \longmapsto c_2$. □

Lemma 270. *If $c_1 \longrightarrow c_2$ and $E \in \text{ECXT}$, then $E[c_1] \longrightarrow E[c_2]$.*

Proof. Suppose $c_1 \longrightarrow c_2$ and $E \in \text{ECXT}$. We proceed by induction on the structure of the derivation of $E \in \text{ECXT}$.

Case 1. ($E = \Box$). Observe that $c_1 = \Box[c_1]$ and $c_2 = \Box[c_2]$. We have $\Box[c_1] \longrightarrow \Box[c_2]$.

Case 2. $(E = E_0[\Box\ c_0])$. By (appL), $c_1\ c_0 \longrightarrow c_2\ c_0$. By the induction hypothesis, $E_0[c_1\ c_0] \longrightarrow E_0[c_2\ c_0]$. We have $E_0[\Box\ c_0][c_1] \longrightarrow E_0[\Box\ c_0][c_2]$.

Case 3. $(E = E_0[v_0\ \Box])$. By (appR), $v_0\ c_1 \longrightarrow v_0\ c_2$. By the induction hypothesis, $E_0[v_0\ c_1] \longrightarrow E_0[v_0\ c_2]$. We have $E_0[v_0\ \Box][c_1] \longrightarrow E_0[v_0\ \Box][c_2]$.

Case 4. $(E = E_0[\Box + c_0])$. By (appL), $c_1 + c_0 \longrightarrow c_2 + c_0$. By the induction hypothesis, $E_0[c_1 + c_0] \longrightarrow E_0[c_2 + c_0]$. We have $E_0[\Box + c_0][c_1] \longrightarrow E_0[\Box + c_0][c_2]$.

Case 5. $(E = E_0[v_0 + \Box])$. By (appR), $v_0 + c_1 \longrightarrow v_0 + c_2$. By the induction hypothesis, $E_0[v_0 + c_1] \longrightarrow E_0[v_0 + c_2]$. We have $E_0[v_0 + \Box][c_1] \longrightarrow E_0[v_0 + \Box][c_2]$.

□

Corollary 271. *If $c_1 \longmapsto c_2$, then $c_1 \longrightarrow c_2$.*

Proof. Suppose $c_1 \longmapsto c_2$. We know $c_1 = E_0[c_{01}]$, $c_2 = E_0[c_{02}]$ and $c_{01}\ \mathscr{R}\ c_{02}$. Observe that $c_{01}\ \mathscr{R}\ c_{02}$ implies $c_{01} \longrightarrow c_{02}$. Let $E = E_0$ in Lemma 270, we get $E_0[c_{01}] \longrightarrow E_0[c_{02}]$. Hence $c_1 \longrightarrow c_2$. □

Theorem 272. *$c_1 \longrightarrow c_2$ if and only if $c_1 \longmapsto c_2$.*

Proof. The theorem is implied by Corollaries 269 and 271. □

We demonstrate the soundness and completeness of the reduction semantics with respect to the structural operational semantics of Environmental ISWIM.

Theorem 273 (Soundness and Completeness of Reduction Semantics w.r.t Structural Operational Semantics of Environmental ISWIM)**.** *$c_1 \longrightarrow^* c_2$ if and only if $c_1 \longmapsto^* c_2$.*

Proof. We first show that if $c_1 \longrightarrow^* c_2$ then $c_1 \longmapsto^* c_2$. Suppose $c_1 \longrightarrow^{(n)} c_2$. We proceed by induction on n.

Case 1. When $n = 0$, $c_1 = c_2$. We have $c_1 \longmapsto^* c_2$ immediately.

Case 2. Let $c_1 \longrightarrow c_3 \longrightarrow^{(n)} c_2$.
Given $c_1 \longrightarrow c_3$, by Corollary 269, $c_1 \longmapsto c_3$. Given $c_3 \longrightarrow^{(n)} c_2$, by the induction hypothesis, $c_3 \longmapsto^* c_2$. We get $c_1 \longmapsto c_3 \longmapsto^* c_2$, that is $c_1 \longmapsto^* c_2$.

Now we show that if $c_1 \longmapsto^* c_2$ then $c_1 \longrightarrow^* c_2$. Suppose $c_1 \longmapsto^{(n)} c_2$. We proceed by induction on n.

Case 1. When $n = 0$, $c_1 = c_2$. We have $c_1 \longrightarrow^* c_2$ immediately.

Case 2. Let $c_1 \longmapsto c_3 \longmapsto^{(n)} c_2$.
Given $c_1 \longmapsto c_3$, by Corollary 271, $c_1 \longrightarrow c_3$. Given $c_3 \longmapsto^{(n)} c_2$, by the induction hypothesis, $c_3 \longrightarrow^* c_2$. We get $c_1 \longrightarrow c_3 \longrightarrow^* c_2$, that is $c_1 \longrightarrow^* c_2$.

Therefore, $c_1 \longrightarrow^* c_2$ if and only if $c_1 \longmapsto^* c_2$. □

.

We prove the Kleene equality of evaluators $eval_{\text{ISWIM:EnvSOS}}(t)$ and $eval_{\text{ISWIM:EnvRed}}(t)$.

Theorem 274 (Kleene Equality of Evaluators)**.** *For any $t \in \text{PRGM}_{\text{ISWIM}}$, $eval_{\text{ISWIM:EnvSOS}}(t)$ is Kleene equal to $eval_{\text{ISWIM:EnvRed}}(t)$.*

Proof. We first show if $eval_{\text{ISWIM:EnvSOS}}(t) = a$ where $a \in \text{ANS}_{\text{ISWIM}}$, then $eval_{\text{ISWIM:EnvRed}}(t) = a$.

Case 1. If $eval_{\text{ISWIM:EnvSOS}}(t) = \texttt{function}$, then $\langle t,\, \emptyset\rangle \longrightarrow^* \triangleleft\lambda x.t',\, \rho\triangleright$. By Theorem 273, $\langle t,\, \emptyset\rangle \longmapsto^* \triangleleft\lambda x.t',\, \rho\triangleright$. We have $eval_{\text{ISWIM:EnvRed}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{ISWIM:EnvSOS}}(t) = n$, then $\langle t,\, \emptyset\rangle \longrightarrow^* n$. By Theorem 273, $\langle t,\, \emptyset\rangle \longmapsto^* n$. We have $eval_{\text{ISWIM:EnvRed}}(t) = n$.

We then show if $eval_{\text{ISWIM:EnvRed}}(t) = a$ where $a \in \text{ANS}_{\text{ISWIM}}$, then $eval_{\text{ISWIM:EnvSOS}}(t) = a$.

Case 1. If $eval_{\text{ISWIM:EnvRed}}(t) = \texttt{function}$, then $\langle t,\, \emptyset\rangle \longmapsto^* \triangleleft\lambda x.t',\, \rho\triangleright$. By Theorem 273, $\langle t,\, \emptyset\rangle \longrightarrow^* \triangleleft\lambda x.t',\, \rho\triangleright$. We have $eval_{\text{ISWIM:EnvSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{ISWIM:EnvRed}}(t) = n$, then $\langle t,\, \emptyset\rangle \longmapsto^* n$. By Theorem 273, $\langle t,\, \emptyset\rangle \longrightarrow^* n$. We have $eval_{\text{ISWIM:EnvSOS}}(t) = n$.

We observe that $eval_{\text{ISWIM:EnvSOS}}(t)$ is undefined if and only if $eval_{\text{ISWIM:EnvRed}}(t)$ is undefined. Therefore, $eval_{\text{ISWIM:EnvSOS}}(t)$ is Kleene equal to $eval_{\text{ISWIM:EnvRed}}(t)$. □

B.5 Equivalence of Reduction Semantics and Abstract Machine (CEK Machine) of Environmental ISWIM

We demonstrate the equivalence of the structural operational semantics of Environmental ISWIM and the abstract machine (CEK machine) of Environmental ISWIM.

Lemma 275. $\langle E,\, v\rangle_{\text{f}} \longmapsto^*_{\text{cek}} \langle E,\, v\rangle_{\text{b}}$.

Proof. We proceed by induction on the structure of the derivation of $v \in \text{VALUE}$.

Case 1. ($\triangleleft\lambda x.t_1,\, \rho\triangleright \in \text{VALUE}$). We immediately have $\langle E,\, \triangleleft\lambda x.t_1,\, \rho\triangleright\rangle_{\text{f}} \longmapsto_{\text{cek}} \langle E,\, \triangleleft\lambda x.t_1,\, \rho\triangleright\rangle_{\text{b}}$.

Case 2. ($n \in \text{VALUE}$). We immediately have $\langle E,\, n\rangle_{\text{f}} \longmapsto_{\text{cek}} \langle E,\, n\rangle_{\text{b}}$.

□

Lemma 276. *If $c_1 \mathrel{\mathscr{R}} c_2$, then $\langle E,\, c_1\rangle_{\text{f}} \longmapsto^*_{\text{cek}} \langle E,\, c_1\rangle_{\text{r}} \longmapsto_{\text{cek}} \langle E,\, c_2\rangle_{\text{f}}$.*

Proof. We proceed by cases on $c_1 \mathrel{\mathscr{R}} c_2$.

Case 1. (app). Let $c_1 = \triangleleft(\lambda x.t_{11}),\, \rho \triangleright v_{12}$ and $c_2 = \langle t_{11},\, \rho[x \mapsto v_{12}]\rangle$. We have:

$$\begin{array}{rll}
 & \langle E,\ \lhd(\lambda x.t_{11}),\ \rho \rhd v_{12}\rangle_{\mathrm{f}} & \\
\longmapsto_{\mathrm{cek}} & \langle E[\Box\ v_{12}],\ \lhd(\lambda x.t_{11}),\ \rho\rhd\rangle_{\mathrm{f}} & \\
\longmapsto_{\mathrm{cek}} & \langle E[\Box\ v_{12}],\ \lhd(\lambda x.t_{11}),\ \rho\rhd\rangle_{\mathrm{b}} & \\
\longmapsto_{\mathrm{cek}} & \langle E[\lhd(\lambda x.t_{11}),\ \rho \rhd \Box],\ v_{12}\rangle_{\mathrm{f}} & \\
\longmapsto^{*}_{\mathrm{cek}} & \langle E[\lhd(\lambda x.t_{11}),\ \rho \rhd \Box],\ v_{12}\rangle_{\mathrm{b}} & \text{by Lemma 275} \\
\longmapsto_{\mathrm{cek}} & \langle E,\ \lhd(\lambda x.t_{11}),\ \rho \rhd v_{12}\rangle_{\mathrm{r}} & \\
\longmapsto_{\mathrm{cek}} & \langle E,\ \langle t_{11},\ \rho[x \mapsto v_{12}]\rangle\rangle_{\mathrm{f}} &
\end{array}$$

Case 2. (plus). Let $c_1 = n_1 + n_2$ and $c_2 = n$ where $n = n_1 + n_2$. We have:

$$\begin{array}{rll}
 & \langle E,\ n_1 + n_2\rangle_{\mathrm{f}} & \\
\longmapsto_{\mathrm{cek}} & \langle E[\Box + n_2],\ n_1\rangle_{\mathrm{f}} & \\
\longmapsto_{\mathrm{cek}} & \langle E[\Box + n_2],\ n_1\rangle_{\mathrm{b}} & \\
\longmapsto_{\mathrm{cek}} & \langle E[n_1 + \Box],\ n_2\rangle_{\mathrm{f}} & \\
\longmapsto_{\mathrm{cek}} & \langle E[n_1 + \Box],\ n_2\rangle_{\mathrm{b}} & \\
\longmapsto_{\mathrm{cek}} & \langle E,\ n_1 + n_2\rangle_{\mathrm{r}} & \\
\longmapsto_{\mathrm{cek}} & \langle E,\ n\rangle_{\mathrm{f}} & \text{where } n = n_1 + n_2
\end{array}$$

Case 3. (other cases). Let $c_1 = \langle t_{11},\ \rho\rangle$ and $\langle t_{11},\ \rho\rangle\ \mathscr{R}\ c_2$. We have:

$$\begin{array}{rl}
 & \langle E,\ \langle t_{11},\ \rho\rangle\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle E,\ \langle t_{11},\ \rho\rangle\rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{cek}} & \langle E,\ c_2\rangle_{\mathrm{f}}
\end{array}$$

□

Lemma 277. *If $c = E_1[c_1]$ and $c_1\ \mathscr{R}\ c_2$, then $\langle E,\ c\rangle_{\mathrm{f}} \longmapsto^{*}_{\mathrm{cek}} \langle EE_1,\ c_1\rangle_{\mathrm{f}}$.*

Proof. Suppose $c = E_1[c_1]$ and $c_1\ \mathscr{R}\ c_2$. We want to show $\langle E,\ c\rangle_{\mathrm{f}} \longmapsto^{*}_{\mathrm{cek}} \langle EE_1,\ c_1\rangle_{\mathrm{f}}$. We proceed by induction on the structure of the derivation of E_1.

Case 1. ($E_1 = \Box$). We know $\langle E,\ c\rangle_{\mathrm{f}} = \langle E,\ c_1\rangle_{\mathrm{f}}$ and $\langle E,\ c_1\rangle_{\mathrm{f}} \longmapsto^{*}_{\mathrm{cek}} \langle E,\ c_1\rangle_{\mathrm{f}}$.

Case 2. ($E_1 = E_{11}\ c_{11}$). We know $\langle E,\ c\rangle_{\mathrm{f}} = \langle E,\ E_1[c_1]\rangle_{\mathrm{f}} = \langle E,\ (E_{11}\ c_{11})[c_1]\rangle_{\mathrm{f}}$. We have:

$$\begin{array}{rl}
 & \langle E,\ (E_{11}\ c_{11})[c_1]\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{cek}} & \langle E[\Box\ c_{11}],\ E_{11}[c_1]\rangle_{\mathrm{f}}
\end{array}$$

Since E_{11} is a component of E_1, by the induction hypothesis, we have $\langle E[\Box\ c_{11}],\ E_{11}[c_1]\rangle_{\mathrm{f}} \longmapsto^{*}_{\mathrm{cek}} \langle E[E_{11}\ c_{11}],\ c_1\rangle_{\mathrm{f}}$.

Case 3. ($E_1 = v_{11}\ E_{11}$). We know $\langle E,\ c\rangle_{\mathrm{f}} = \langle E,\ E_1[c_1]\rangle_{\mathrm{f}} = \langle E,\ (v_{11}\ E_{11})[c_1]\rangle_{\mathrm{f}}$. We have:

$$\begin{array}{rl} & \langle E,\ (v_{11}\ E_{11})[c_1]\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{cek}} & \langle E[\square\ E_{11}[c_1]],\ v_{11}\rangle_{\mathrm{f}} \\ \longmapsto^*_{\mathrm{cek}} & \langle E[\square\ E_{11}[c_1]],\ v_{11}\rangle_{\mathrm{b}} \quad \text{by Lemma 275} \\ \longmapsto_{\mathrm{cek}} & \langle E[v_{11}\ \square],\ E_{11}[c_1]\rangle_{\mathrm{f}} \end{array}$$

Since E_{11} is a component of E_1, by the induction hypothesis, we have $\langle E[v_{11}\ \square],\ E_{11}[c_1]\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} \langle E[v_{11}\ E_{11}],\ c_1\rangle_{\mathrm{f}}$.

Case 4. $(E_1 = E_{11} + c_{11})$. We know $\langle E,\ c\rangle_{\mathrm{f}} = \langle E,\ E_1[c_1]\rangle_{\mathrm{f}} = \langle E,\ (E_{11}+c_{11})[c_1]\rangle_{\mathrm{f}}$. We have:

$$\begin{array}{rl} & \langle E,\ (E_{11}+c_{11})[c_1]\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{cek}} & \langle E[\square + c_{11}],\ E_{11}[c_1]\rangle_{\mathrm{f}} \end{array}$$

Since E_{11} is a component of E_1, by the induction hypothesis, we have $\langle E[\square+c_{11}],\ E_{11}[c_1]\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} \langle E[E_{11}+c_{11}],\ c_1\rangle_{\mathrm{f}}$.

Case 5. $(E_1 = v_{11} + E_{11})$. We know $\langle E,\ c\rangle_{\mathrm{f}} = \langle E,\ E_1[c_1]\rangle_{\mathrm{f}} = \langle E,\ (v_{11}+E_{11})[c_1]\rangle_{\mathrm{f}}$. We have:

$$\begin{array}{rl} & \langle E,\ (v_{11}+E_{11})[c_1]\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{cek}} & \langle E[\square + E_{11}[c_1]],\ v_{11}\rangle_{\mathrm{f}} \\ \longmapsto^*_{\mathrm{cek}} & \langle E[\square + E_{11}[c_1]],\ v_{11}\rangle_{\mathrm{b}} \quad \text{by Lemma 275} \\ \longmapsto_{\mathrm{cek}} & \langle E[v_{11} + \square],\ E_{11}[c_1]\rangle_{\mathrm{f}} \end{array}$$

Since E_{11} is a component of E_1, by the induction hypothesis, we have $\langle E[v_{11}+\square],\ E_{11}[c_1]\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} \langle E[v_{11}+E_{11}],\ c_1\rangle_{\mathrm{f}}$.

□

Lemma 278. *If $E_0[c_0] = E_1[c_1]$ and $E_1[c_1] \longmapsto E_1[c_2]$ where $c_1\ \mathscr{R}\ c_2$, then $\langle E_0,\ c_0\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} \langle E_1,\ c_2\rangle_{\mathrm{f}}$.*

Proof. If c_1 is inside c_0 (or the same as c_0), E_1 extends E_0 (or is the same as E_0). Otherwise, because c_0 is not reduced, it must be a value.

Case 1. Suppose c_1 is inside c_0 (or the same as c_0). Let $c_0 = E_2[c_1]$. Then $E_1 = E_0E_2$. We have $\langle E_0,\ c_0\rangle_{\mathrm{f}} = \langle E_0,\ E_2[c_1]\rangle_{\mathrm{f}}$. By Lemma 277, $\langle E_0,\ E_2[c_1]\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} \langle E_0E_2,\ c_1\rangle_{\mathrm{f}}$. Given $c_1\ \mathscr{R}\ c_2$, by Lemma 276, $\langle E_0E_2,\ c_1\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} \langle E_0E_2,\ c_1\rangle_{\mathrm{r}} \longmapsto_{\mathrm{cek}} \langle E_0E_2,\ c_2\rangle_{\mathrm{f}}$.

Case 2. Otherwise, $c_0 \in \textsc{Value}$. By Lemma 275, $\langle E_0,\ c_0\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} \langle E_0,\ c_0\rangle_{\mathrm{b}}$. We prove the following statement by induction on the structure of the derivation of $E_0 \in \textsc{Ecxt}$.

Statement: If $E_0[c_0] = E_1[c_1]$ where $c_0 \in \textsc{Value}$ and $E_1[c_1] \longmapsto E_1[c_2]$ where $c_1\ \mathscr{R}\ c_2$, then $\langle E_0,\ c_0\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{cek}} \langle E_1,\ c_2\rangle_{\mathrm{f}}$.

Case i. $(E_0 = \square)$. This case is vacuous.

Case ii. $(E_0 = E_2[\Box\ c_{22}])$. We have:

$$\begin{array}{rl} & \langle E_0,\ c_0\rangle_{\mathrm{b}} \\ = & \langle E_2[\Box\ c_{22}],\ c_0\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{cek}} & \langle E_2[c_0\ \Box],\ c_{22}\rangle_{\mathrm{f}} \end{array}$$

Case a. If $c_{22}\ \mathscr{R}\ c_{23}$, then $c_1 = c_{22}$ and $c_2 = c_{23}$.
By Lemma 276, $\langle E_2[c_0\ \Box],\ c_{22}\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} \langle E_2[c_0\ \Box],\ c_2\rangle_{\mathrm{f}}$.

Case b. If $c_{22}\ \not\mathscr{R}$ and $c_{22} \in$ VALUE, we have:

$$\begin{array}{rl} & \langle E_2[c_0\ \Box],\ c_{22}\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{cek}} & \langle E_2[c_0\ \Box],\ c_{22}\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{cek}} & \langle E_2,\ c_0\ c_{22}\rangle_{\mathrm{r}} \end{array}$$

Then $c_1 = c_0\ c_{22}$. By Lemma 276, $\langle E_2,\ c_0\ c_{22}\rangle_{\mathrm{r}} \longmapsto_{\mathrm{cek}} \langle E_2,\ c_2\rangle_{\mathrm{f}}$.

Case c. If $c_{22}\ \not\mathscr{R}$ and $c_{22} \notin$ VALUE, then $c_{22} = E_3[c_1]$. Hence $E_1 = E_2[c_0\ E_3]$. We have:

$$\begin{array}{rll} & \langle E_2[c_0\ \Box],\ c_{22}\rangle_{\mathrm{f}} & \\ = & \langle E_2[c_0\ \Box],\ E_3[c_1]\rangle_{\mathrm{f}} & \\ \longmapsto^*_{\mathrm{cek}} & \langle E_2[c_0\ E_3],\ c_1\rangle_{\mathrm{f}} & \text{by Lemma 277} \\ \longmapsto^*_{\mathrm{cek}} & \langle E_2[c_0\ E_3],\ c_2\rangle_{\mathrm{f}} & \text{by Lemma 276} \end{array}$$

Case iii. $(E_0 = E_2[v_{21}\ \Box])$. We have:

$$\begin{array}{rl} & \langle E_0,\ c_0\rangle_{\mathrm{b}} \\ = & \langle E_2[v_{21}\ \Box],\ c_0\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{cek}} & \langle E_2,\ v_{21}\ c_0\rangle_{\mathrm{r}} \end{array}$$

Then $c_1 = v_{21}\ c_0$. By Lemma 276, $\langle E_2,\ v_{21}\ c_0\rangle_{\mathrm{r}} \longmapsto_{\mathrm{cek}} \langle E_2,\ c_2\rangle_{\mathrm{f}}$.

Case iv. $(E_0 = E_2[\Box + c_{22}])$. We have:

$$\begin{array}{rl} & \langle E_0,\ c_0\rangle_{\mathrm{b}} \\ = & \langle E_2[\Box + c_{22}],\ c_0\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{cek}} & \langle E_2[c_0 + \Box],\ c_{22}\rangle_{\mathrm{f}} \end{array}$$

Case a. If $c_{22}\ \mathscr{R}\ c_{23}$, then $c_1 = c_{22}$ and $c_2 = c_{23}$.
By Lemma 276, $\langle E_2[c_0 + \Box],\ c_{22}\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} \langle E_2[c_0 + \Box],\ c_2\rangle_{\mathrm{f}}$.

Case b. If $c_{22}\ \not\mathscr{R}$ and $c_{22} \in$ VALUE, we have:

$$\begin{array}{rl} & \langle E_2[c_0 + \Box],\ c_{22}\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{cek}} & \langle E_2[c_0 + \Box],\ c_{22}\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{cek}} & \langle E_2,\ c_0 + c_{22}\rangle_{\mathrm{r}} \end{array}$$

Then $c_1 = c_0 + c_{22}$. By Lemma 276, $\langle E_2,\ c_0 + c_{22}\rangle_{\mathrm{r}} \longmapsto_{\mathrm{cek}} \langle E_2,\ c_2\rangle_{\mathrm{f}}$.

Case c. If $c_{22}\ \not\mathscr{R}$ and $c_{22} \notin$ VALUE, then $c_{22} = E_3[c_1]$. Hence $E_1 = E_2[c_0 + E_3]$. We have:

$$\begin{array}{lll} & \langle E_2[c_0 + \Box],\ c_{22}\rangle_{\mathrm{f}} & \\ = & \langle E_2[c_0 + \Box],\ E_3[c_1]\rangle_{\mathrm{f}} & \\ \longmapsto^*_{\mathrm{cek}} & \langle E_2[c_0 + E_3],\ c_1\rangle_{\mathrm{f}} & \text{by Lemma 277} \\ \longmapsto^*_{\mathrm{cek}} & \langle E_2[c_0 + E_3],\ c_2\rangle_{\mathrm{f}} & \text{by Lemma 276} \end{array}$$

Case v. $(E_0 = E_2[v_{21} + \Box])$. We have:

$$\begin{array}{ll} & \langle E_0,\ c_0\rangle_{\mathrm{b}} \\ = & \langle E_2[v_{21} + \Box],\ c_0\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{cek}} & \langle E_2,\ v_{21} + c_0\rangle_{\mathrm{r}} \end{array}$$

Then $c_1 = v_{21} + c_0$. By Lemma 276, $\langle E_2,\ v_{21} + c_0\rangle_{\mathrm{r}} \longmapsto_{\mathrm{cek}} \langle E_2,\ c_2\rangle_{\mathrm{f}}$.

□

Lemma 279. *If $v = E[c]$, then $\langle E,\ c\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{cek}} v$.*

Proof. Suppose $v = E[c]$. We know $c \in$ VALUE. We proceed by induction on the structure of the derivation of $E \in$ ECXT. □

Case 1. $(E = \Box)$. Then $v = c$. We have $\langle \Box,\ v\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mek}} v$.

Case 2. $(E = E_1[\Box\ c_{12}])$. Then $v = E_1[c\ c_{12}]$. We know $c \in$ VALUE and $c_{12} \in$ VALUE. We have:

$$\begin{array}{lll} & \langle E_1[\Box\ c_{12}],\ c\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{cek}} & \langle E_1[c\ \Box],\ c_{12}\rangle_{\mathrm{f}} & \\ \longmapsto^*_{\mathrm{cek}} & \langle E_1[c\ \Box],\ c_{12}\rangle_{\mathrm{b}} & \text{by Lemma 275} \\ \longmapsto_{\mathrm{cek}} & \langle E_1,\ c\ c_{12}\rangle_{\mathrm{f}} & \\ \longmapsto^*_{\mathrm{cek}} & \langle E_1,\ c\ c_{12}\rangle_{\mathrm{b}} & \text{by Lemma 275} \end{array}$$

Since E_1 is a component of E, by the induction hypothesis, we have $\langle E_1,\ c\ c_{12}\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{cek}} v$.

Case 3. $(E = E_1[v_{11}\ \Box])$. Then $v = E_1[v_{11}\ c]$. We know $c \in \text{VALUE}^i$. We have:

$$\begin{array}{lll} & \langle E_1[v_{11}\ \Box],\ c\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{cek}} & \langle E_1,\ v_{11}\ c\rangle_{\mathrm{f}} & \\ \longmapsto^*_{\mathrm{cek}} & \langle E_1,\ v_{11}\ c\rangle_{\mathrm{b}} & \text{by Lemma 275} \end{array}$$

Since E_1 is a component of E, by the induction hypothesis, we have $\langle E_1,\ v_{11}\ c\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{cek}} v$.

Case 4. $(E = E_1[\Box + c_{12}])$. Then $v = E_1[c + c_{12}]$. We know $c \in$ VALUE and $c_{12} \in$ VALUE. We have:

$$\begin{array}{lll} & \langle E_1[\Box + c_{12}],\ c\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{cek}} & \langle E_1[c + \Box],\ c_{12}\rangle_{\mathrm{f}} & \\ \longmapsto^*_{\mathrm{cek}} & \langle E_1[c + \Box],\ c_{12}\rangle_{\mathrm{b}} & \text{by Lemma 275} \\ \longmapsto_{\mathrm{cek}} & \langle E_1,\ c + c_{12}\rangle_{\mathrm{f}} & \\ \longmapsto^*_{\mathrm{cek}} & \langle E_1,\ c + c_{12}\rangle_{\mathrm{b}} & \text{by Lemma 275} \end{array}$$

Since E_1 is a component of E, by the induction hypothesis, we have $\langle E_1,\ c + c_{12}\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{cek}} v$.

Case 5. ($E = E_1[v_{11} + \Box]$). Then $v = E_1[v_{11} + c]$. We know $c \in \textsc{Value}^i$. We have:

$$\begin{array}{ll} & \langle E_1[v_{11} + \Box],\ c\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{cek}} & \langle E_1,\ v_{11} + c\rangle_{\mathrm{f}} \\ \longmapsto^*_{\mathrm{cek}} & \langle E_1,\ v_{11} + c\rangle_{\mathrm{b}} \qquad \text{by Lemma 275} \end{array}$$

Since E_1 is a component of E, by the induction hypothesis, we have $\langle E_1,\ v_{11} + c\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{cek}} v$.

Lemma 280. *If $v = E[c]$, then $\langle E,\ c\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} v$.*

Proof. Suppose $v = E[c]$. Then $c \in \textsc{Value}$. By Lemma 275, $\langle E,\ c\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} \langle E,\ c\rangle_{\mathrm{b}}$. By Lemma 279, $\langle E,\ c\rangle_{\mathrm{b}} \longmapsto_{\mathrm{cek}} v$. Hence $\langle E,\ c\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} v$. □

Lemma 281. *If $E[c_1] \longmapsto^* v_2$, then $\langle E,\ c_1\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} v_2$.*

Proof. Suppose $E[c_1] \longmapsto^{(n)}_{\mathrm{cek}} v_2$. We proceed by induction on n.

Case 1. When $n = 0$, $v_2 = E[c_1]$. By Lemma 280, $\langle E,\ c_1\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} v_2$.

Case 2. Let $E[c_1] \longmapsto E_1[c_2] \longmapsto^{(n)} v_2$ where $E[c_1] = E_1[c_{11}]$ and $c_{11}\ \mathscr{R}\ c_2$. By Lemma 278, $\langle E,\ c_1\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} \langle E_1,\ c_2\rangle_{\mathrm{f}}$. Given $E_1[c_2] \longmapsto^{(n)} v_2$, by the induction hypothesis, $\langle E_1,\ c_2\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} v_2$. Hence we have $\langle E,\ c_1\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} v_2$.

□

We demonstrate the soundness of the CEK machine with respect to the reduction semantics of Environmental ISWIM.

Theorem 282 (Soundness of CEK Machine w.r.t. Reduction Semantics of Environmental ISWIM)**.** *For any $t_1 \in \textsc{Prgm}_{\mathrm{ISWIM}}$, if $\langle t_1,\ \emptyset\rangle \longmapsto^* v_2$, then $\langle \Box,\ \langle t_1,\ \emptyset\rangle\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} v_2$.*

Proof. Suppose $\Box[\langle t_1,\ \emptyset\rangle] \longmapsto^* v_2$, by Lemma 281, $\langle \Box,\ \langle t_1,\ \emptyset\rangle\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{cek}} v_2$. □

Any machine configuration in the CEK machine can be translated to its corresponding representation as a configuration in Environmental ISWIM.

Definition 283 (Translator)**.** Define the translator $\mathscr{T}_{\mathrm{cek}\to\mathrm{env}}$ to be a total function from the set of machine configurations CFG to the set of configurations CONF.

$$\mathscr{T}_{\mathrm{cek}\to\mathrm{env}} : \textsc{Cfg} \to \textsc{Conf}$$

$$\begin{array}{rcl} \mathscr{T}_{\mathrm{cek}\to\mathrm{env}}(v) & = & v \\ \mathscr{T}_{\mathrm{cek}\to\mathrm{env}}(\langle E,\ c\rangle_{\mathrm{r}}) & = & E[c] \\ \mathscr{T}_{\mathrm{cek}\to\mathrm{env}}(\langle E,\ c\rangle_{\mathrm{f}}) & = & E[c] \\ \mathscr{T}_{\mathrm{cek}\to\mathrm{env}}(\langle E,\ c\rangle_{\mathrm{b}}) & = & E[c] \end{array}$$

Lemma 284. *If* $C_1 \longmapsto_{\text{cek}} C_2$, *then* $\mathscr{T}_{\text{cek}\to\text{env}}(C_1) \longmapsto^* \mathscr{T}_{\text{cek}\to\text{env}}(C_2)$.

Proof. We proceed by cases on $C_1 \longmapsto_{\text{cek}} C_2$. □

Case 1. Reduce rules: Let $C_1 = \langle E,\ c_1\rangle_{\text{r}}$ and $C_2 = \langle E,\ c_2\rangle_{\text{f}}$. Then $E[c_1] \longmapsto E[c_2]$ where $c_1\ \mathscr{R}\ c_2$. Hence $\mathscr{T}_{\text{cek}\to\text{env}}(C_1) \longmapsto^* \mathscr{T}_{\text{cek}\to\text{env}}(C_2)$.

Case 2. Focus rules: Let $C_1 = \langle E_1,\ c_1\rangle_{\text{f}}$ and $C_2 = \langle E_2,\ c_2\rangle_{?}$. Then $E_1[c_1] = E_2[c_2]$. Hence $\mathscr{T}_{\text{cek}\to\text{env}}(C_1) \longmapsto^* \mathscr{T}_{\text{cek}\to\text{env}}(C_2)$.

Case 3. Build rules:

Case i. (b-val). Let $C_1 = \langle \Box,\ v\rangle_{\text{b}}$ and $C_2 = v$. Then $\Box[v] = v$. Hence $\mathscr{T}_{\text{cek}\to\text{env}}(C_1) \longmapsto^* \mathscr{T}_{\text{cek}\to\text{env}}(C_2)$.

Case ii. (other rules). Let $C_1 = \langle E_1,\ c_1\rangle_{\text{b}}$ and $C_2 = \langle E_2,\ c_2\rangle_{?}$. Then $E_1[c_1] = E_2[c_2]$. Hence $\mathscr{T}_{\text{cek}\to\text{env}}(C_1) \longmapsto^* \mathscr{T}_{\text{cek}\to\text{env}}(C_2)$.

Lemma 285. *If* $C_1 \longmapsto^*_{\text{cek}} C_2$, *then* $\mathscr{T}_{\text{cek}\to\text{env}}(C_1) \longmapsto^* \mathscr{T}_{\text{cek}\to\text{env}}(C_2)$.

Proof. Suppose $C_1 \longmapsto^{(n)}_{\text{cek}} C_2$. We proceed by induction on n.

Case 1. (0). Then $C_1 = C_2$. Then $\mathscr{T}_{\text{cek}\to\text{env}}(C_1) = \mathscr{T}_{\text{cek}\to\text{env}}(C_2)$. We have $\mathscr{T}_{\text{cek}\to\text{env}}(C_1) \longmapsto^* \mathscr{T}_{\text{cek}\to\text{env}}(C_2)$ immediately.

Case 2. $(n+1)$. Let $C_1 \longmapsto_{\text{cek}} C_3 \longmapsto^{(n)}_{\text{cek}} C_2$.
Given $C_1 \longmapsto_{\text{cek}} C_3$, by Lemma 284, $\mathscr{T}_{\text{cek}\to\text{env}}(C_1) \longmapsto^* \mathscr{T}_{\text{cek}\to\text{env}}(C_3)$.
Given $C_3 \longmapsto^{(n)}_{\text{cek}} C_2$, by the induction hypothesis, $\mathscr{T}_{\text{cek}\to\text{env}}(C_3) \longmapsto^* \mathscr{T}_{\text{cek}\to\text{env}}(C_2)$.
Hence $\mathscr{T}_{\text{cek}\to\text{env}}(C_1) \longmapsto^* \mathscr{T}_{\text{cek}\to\text{env}}(C_2)$.

□

We demonstrate the completeness of the CEK machine with respect to the reduction semantics of Environmental ISWIM.

Theorem 286 (Completeness of CEK Machine w.r.t. Reduction Semantics of Environmental ISWIM)**.** *For any* $t_1 \in \text{PRGM}_{\text{ISWIM}}$, *if* $\langle \Box,\ \langle t,\ \emptyset\rangle\rangle_{\text{f}} \longmapsto^*_{\text{cek}} v_2$, *then* $\langle t,\ \emptyset\rangle \longmapsto^* v_2$.

Proof. If $\langle \Box,\ \langle t,\ \emptyset\rangle\rangle_{\text{f}} \longmapsto^*_{\text{cek}} v_2$, by Lemma 285, $\mathscr{T}_{\text{cek}\to\text{env}}(\langle \Box,\ \langle t,\ \emptyset\rangle\rangle_{\text{f}}) \longmapsto^* \mathscr{T}_{\text{cek}\to\text{env}}(v_2)$. We have $\langle t,\ \emptyset\rangle \longmapsto^* v_2$. □

We prove the Kleene equality of evaluators $eval_{\text{ISWIM:EnvRed}}(t)$ and $eval_{\text{ISWIM:CEK}}(t)$.

Theorem 287 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{ISWIM}}$, $eval_{\text{ISWIM:EnvRed}}(t)$ *is Kleene equal to* $eval_{\text{ISWIM:CEK}}(t)$.

Proof. We first show if $eval_{\text{ISWIM:EnvRed}}(t) = a$ where $a \in \text{ANS}_{\text{ISWIM}}$, then $eval_{\text{ISWIM:CEK}}(t) = a$.

Case 1. If $eval_{\text{ISWIM:EnvRed}}(t) = \texttt{function}$, then $\langle t, \emptyset\rangle \longmapsto^* \triangleleft\lambda x.t', \rho\triangleright$. By Theorem 282, $\langle\Box, \langle t, \emptyset\rangle\rangle_{\text{f}} \longmapsto^*_{\text{cek}} \triangleleft\lambda x.t', \rho\triangleright$. We have $eval_{\text{ISWIM:CEK}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{ISWIM:EnvRed}}(t) = n$, then $\langle t, \emptyset\rangle \longmapsto^* n$. By Theorem 282, $\langle\Box, \langle t, \emptyset\rangle\rangle_{\text{f}} \longmapsto^*_{\text{cek}} n$. We have $eval_{\text{ISWIM:CEK}}(t) = n$.

We then show if $eval_{\text{ISWIM:CEK}}(t) = a$ where $a \in \text{ANS}_{\text{ISWIM}}$, then $eval_{\text{ISWIM:EnvRed}}(t) = a$.

Case 1. If $eval_{\text{ISWIM:CEK}}(t) = \texttt{function}$, then $\langle\Box, \langle t, \emptyset\rangle\rangle_{\text{f}} \longmapsto^*_{\text{cek}} \triangleleft\lambda x.t', \rho\triangleright$. By Theorem 286, $\langle t, \emptyset\rangle \longmapsto^* \triangleleft\lambda x.t', \rho\triangleright$. We have $eval_{\text{ISWIM:EnvRed}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{ISWIM:CEK}}(t) = n$, then $\langle\Box, \langle t, \emptyset\rangle\rangle_{\text{f}} \longmapsto^*_{\text{cek}} n$. By Theorem 286, $\langle t, \emptyset\rangle \longmapsto^* n$. We have $eval_{\text{ISWIM:EnvRed}}(t) = n$.

We observe that $eval_{\text{ISWIM:EnvRed}}(t)$ is undefined if and only if $eval_{\text{ISWIM:CEK}}(t)$ is undefined. Therefore, $eval_{\text{ISWIM:EnvRed}}(t)$ is Kleene equal to $eval_{\text{ISWIM:CEK}}(t)$. □

Appendix C

Proofs of Chapter 4

C.1 Equivalence of Substitutional Structural Operational Semantics and Substitutional Reduction Semantics of MetaML

We demonstrate the equivalence of the substitutional structural operational semantics of MetaML and the substitutional reduction semantics of Environmental MetaML.

Lemma 288. *If* $t_1^i \longrightarrow^i t_2^i$ *and* $E \in \text{EXCT}^{i\multimap j}$*, then* $E^{i\multimap j}[t_1^i] \longmapsto^j E^{i\multimap j}[t_2^i]$.

Proof. Suppose that $t_1^i \longrightarrow^i t_2^i$ and $E \in \text{EXCT}^{i\multimap j}$. We show there exists some $E_0 \in \text{ECXT}^{k\multimap j}$ and $t_{01}, t_{02} \in \text{TERM}^k$ such that $E^{i\multimap j}[t_1^i] = E_0^{k\multimap j}[t_{01}^k]$ and $E^{i\multimap j}[t_2^i] = E_0^{k\multimap j}[t_{02}^k]$ where $t_{01}^k \longrightarrow^k t_{02}^k$. We proceed by induction on the structure of the derivation of $t_1^i \longrightarrow^i t_2^i$.

Case 1. (lambda-(i+1)). Let $t_1^{i+1} = \lambda x.t_{11}^{i+1}$, $t_2^{i+1} = \lambda x.t_{21}^{i+1}$ and $t_{11}^{i+1} \longrightarrow^{i+1} t_{21}^{i+1}$. Let $E_0^{(i+1)\multimap j} = E^{(i+1)\multimap j}[\lambda x.\Box]$. By the induction hypothesis, $E_0^{(i+1)\multimap j}[t_{11}^{i+1}] \longmapsto^j E_0^{(i+1)\multimap j}[t_{21}^{i+1}]$. Thus $E^{(i+1)\multimap j}[\lambda x.t_{11}^{i+1}] \longmapsto^j E^{(i+1)\multimap j}[\lambda x.t_{21}^{i+1}]$.

Case 2. (appL-i). Let $t_1^i = t_{11}^i\ t_{12}^i$, $t_2^i = t_{21}^i\ t_{12}^i$ and $t_{11}^i \longrightarrow^i t_{21}^i$. Let $E_0^{i\multimap j} = E^{i\multimap j}[\Box\ t_{12}^i]$. By the induction hypothesis, $E_0^{i\multimap j}[t_{11}^i] \longmapsto^j E_0^{i\multimap j}[t_{21}^i]$. Thus $E^{i\multimap j}[t_{11}^i\ t_{12}^i] \longmapsto^j E^{i\multimap j}[t_{21}^i\ t_{12}^i]$.

Case 3. (appR-i). Let $t_1^i = v_{11}^i\ t_{12}^i$, $t_2^i = v_{11}^i\ t_{22}^i$ and $t_{11}^i \longrightarrow^i t_{22}^i$. Let $E_0^{i\multimap j} = E^{i\multimap j}[v_{11}^i\ \Box]$. By the induction hypothesis, $E_0^{i\multimap j}[t_{12}^i] \longmapsto^j E_0^{i\multimap j}[t_{22}^i]$. Thus $E^{i\multimap j}[v_{11}^i\ t_{12}^i] \longmapsto^j E^{i\multimap j}[v_{11}^i\ t_{22}^i]$.

Case 4. (app-0). Let $t_1^0 = (\lambda x.t^0)\ v^0$ and $t_2^0 = t^0[v^0/x]$. We have $E^{0\multimap j}[t_1^0] \longmapsto^j E^{0\multimap j}[t_2^0]$ because $t_1^0\ \mathscr{R}^0\ t_2^0$.

Case 5. (run-0). Let $t_1^0 = !\langle v^1\rangle$ and $t_2^0 = v^1$. We have $E^{0\multimap j}[t_1^0] \longmapsto^j E^{0\multimap j}[t_2^0]$ because $t_1^0\ \mathscr{R}^0\ t_2^0$.

Case 6. (run-i). Let $t_1^i = !t_{11}^i$, $t_2^i = !t_{21}^i$ and $t_{11}^i \longrightarrow^i t_{21}^i$. Let $E_0^{i\multimap j} = E^{i\multimap j}[!\Box]$. By the induction hypothesis, $E_0^{i\multimap j}[t_{11}^i] \longmapsto^j E_0^{i\multimap j}[t_{21}^i]$. Thus $E^{i\multimap j}[!t_{11}^i] \longmapsto^j E^{i\multimap j}[!t_{21}^i]$.

Case 7. (code-i). Let $t_1^i = \langle t_{11}^{i+1}\rangle$, $t_2^i = \langle t_{21}^{i+1}\rangle$ and $t_{11}^{i+1} \longrightarrow^{i+1} t_{21}^{i+1}$. Let $E_0^{(i+1)\multimap j} = E^{i\multimap j}[\langle\Box\rangle]$. By the induction hypothesis, $E_0^{(i+1)\multimap j}[t_{11}^{i+1}] \longmapsto^j E_0^{(i+1)\multimap j}[t_{21}^{i+1}]$. Thus $E^{i\multimap j}[\langle t_{11}^{i+1}\rangle] \longmapsto^j E^{i\multimap j}[\langle t_{21}^{i+1}\rangle]$.

Case 8. (splice-1). Let $t_1^1 = {\sim}\langle v^1\rangle$ and $t_2^1 = v^1$. We have $E^{1\multimap j}[t_1^1] \longmapsto^j E^{1\multimap j}[t_2^1]$ because $t_1^1\ \mathscr{R}^1\ t_2^1$.

Case 9. (splice-(i+1)). Let $t_1^{i+1} = {\sim}t_{11}^i$, $t_2^{i+1} = {\sim}t_{21}^i$ and $t_{11}^i \longrightarrow^i t_{21}^i$. Let $E_0^{i\multimap j} = E^{(i+1)\multimap j}[{\sim}\Box]$. By the induction hypothesis, $E_0^{i\multimap j}[t_{11}^i] \longmapsto^j E_0^{i\multimap j}[t_{21}^i]$. Thus $E^{(i+1)\multimap j}[{\sim}t_{11}^i] \longmapsto^j E^{(i+1)\multimap j}[{\sim}t_{21}^i]$.

Case 10. (plusL-i). Let $t_1^i = t_{11}^i + t_{12}^i$, $t_2^i = t_{21}^i + t_{12}^i$ and $t_{11}^i \longrightarrow^i t_{21}^i$. Let $E_0^{i-\circ j} = E^{i-\circ j}[\Box + t_{12}^i]$. By the induction hypothesis, $E_0^{i-\circ j}[t_{11}^i] \longmapsto^j E_0^{i-\circ j}[t_{21}^i]$. Thus $E^{i-\circ j}[t_{11}^i + t_{12}^i] \longmapsto^j E^{i-\circ j}[t_{21}^i + t_{12}^i]$.

Case 11. (plusR-i). Let $t_1^i = v_{11}^i + t_{12}^i$, $t_2^i = v_{11}^i + t_{22}^i$ and $t_{11}^i \longrightarrow^i t_{22}^i$. Let $E_0^{i-\circ j} = E^{i-\circ j}[v_{11}^i + \Box]$. By the induction hypothesis, $E_0^{i-\circ j}[t_{12}^i] \longmapsto^j E_0^{i-\circ j}[t_{22}^i]$. Thus $E^{i-\circ j}[v_{11}^i + t_{12}^i] \longmapsto^j E^{i-\circ j}[v_{11}^i + t_{22}^i]$.

Case 12. (plus-0). Let $t_1^0 = n_1 + n_2$ and $t_2^0 = n$ where $n = n_1 + n_2$. We have $E^{0-\circ j}[t_1^0] \longmapsto^j E^{0-\circ j}[t_2^0]$ because $t_1^0 \mathscr{R}^0 t_2^0$.

Therefore, if $t_1^i \longrightarrow^i t_2^i$ and $E \in \text{EXCT}^{i-\circ j}$, then $E^{i-\circ j}[t_1^i] \longmapsto^j E^{i-\circ j}[t_2^i]$. □

Corollary 289. *If $t_1^i \longrightarrow^i t_2^i$, then $t_1^i \longmapsto^i t_2^i$.*

Proof. Suppose $t_1^i \longrightarrow^i t_2^i$. Let $E = \Box^{i-\circ i}$ in Lemma 288. We get $\Box^{i-\circ i}[t_1^i] \longmapsto^i \Box^{i-\circ i}[t_2^i]$. Hence $t_1^i \longmapsto^i t_2^i$. □

Lemma 290. *If $t_1^i \longrightarrow^i t_2^i$ and $E \in \text{ECXT}^{i-\circ j}$, then $E^{i-\circ j}[t_1^i] \longrightarrow^j E^{i-\circ j}[t_2^i]$.*

Proof. Suppose $t_1^i \longrightarrow^i t_2^i$ and $E \in \text{ECXT}^{i-\circ j}$. We proceed by induction on the structure of the derivation $E \in \text{ECXT}^{i-\circ j}$.

Case 1. ($E = \Box$). Observe that $t_1^i = \Box^{i-\circ i}[t_1^i]$ and $t_2^i = \Box^{i-\circ i}[t_2^i]$. We have $\Box^{i-\circ i}[t_1^i] \longrightarrow^i \Box^i[t_1^i]$

Case 2. ($E = E_0^{i-\circ j}[\Box\ t_0^i]$). By (appL-i), $t_1^i\ t_0^i \longrightarrow^i t_2^i\ t_0^i$. By the induction hypothesis, $E_0^{i-\circ j}[t_1^i\ t_0^i] \longrightarrow^j E_0^{i-\circ j}[t_2^i\ t_0^i]$. Thus $E_0^{i-\circ j}[\Box\ t_0^i][t_1^i] \longrightarrow^j E_0^{i-\circ j}[\Box\ t_0^i][t_2^i]$.

Case 3. ($E = E_0^{i-\circ j}[v_0^i\ \Box]$). By (appR-i), $v_0^i\ t_1^i \longrightarrow^i v_0^i\ t_2^i$. By the induction hypothesis, $E_0^{i-\circ j}[v_0^i\ t_1^i] \longrightarrow^j E_0^{i-\circ j}[v_0^i\ t_2^i]$. Thus $E_0^{i-\circ j}[v_0^i\ \Box][t_1^i] \longrightarrow^j E_0^{i-\circ j}[v_0^i\ \Box][t_2^i]$.

Case 4. ($E = E_0^{(i+1)-\circ j}[\lambda x.\Box]$). By (lambda-(i+1)), $\lambda x.t_1^{i+1} \longrightarrow^{i+1} \lambda x.t_2^{i+1}$.
By the induction hypothesis, $E_0^{(i+1)-\circ j}[\lambda x.t_1^{i+1}] \longrightarrow^j E_0^{(i+1)-\circ j}[\lambda x.t_2^{i+1}]$. Thus $E_0^{(i+1)-\circ j}[\lambda x.\Box][t_1^{i+1}] \longrightarrow^j E_0^{(i+1)-\circ j}[\lambda x.\Box][t_2^{i+1}]$.

Case 5. ($E = E_0^{i-\circ j}[\langle\Box\rangle]$). By (code-i), $\langle t_1^{i+1}\rangle \longrightarrow^i \langle t_2^{i+1}\rangle$. By the induction hypothesis, $E_0^{i-\circ j}[\langle t_1^{i+1}\rangle] \longrightarrow^j E_0^{i-\circ j}[\langle t_2^{i+1}\rangle]$. Thus $E_0^{i-\circ j}[\langle\Box\rangle][t_1^{i+1}] \longrightarrow^j E_0^{i-\circ j}[\langle\Box\rangle][t_2^{i+1}]$.

Case 6. ($E = E_0^{(i+1)-\circ j}[\sim\Box]$). By (splice-(i+1)), $\sim t_1^i \longrightarrow^{i+1} \sim t_2^i$. By the induction hypothesis, $E_0^{(i+1)-\circ j}[\sim t_1^i] \longrightarrow^j E_0^{(i+1)-\circ j}[\sim t_2^i]$. Thus $E_0^{(i+1)-\circ j}[\sim\Box][t_1^i] \longrightarrow^j E_0^{(i+1)-\circ j}[\sim\Box][t_2^i]$.

Case 7. ($E = E_0^{i-\circ j}[!\Box]$). By (run-i), $!t_1^i \longrightarrow^i !t_2^i$. By the induction hypothesis, $E_0^{i-\circ j}[!t_1^i] \longrightarrow^j E_0^{i-\circ j}[!t_2^i]$. Thus $E_0^{i-\circ j}[!\Box][t_1^i] \longrightarrow^j E_0^{i-\circ j}[!\Box][t_2^i]$.

Case 8. ($E = E_0^{i-\circ j}[\Box + t_0^i]$). By (plusL-i), $t_1^i + t_0^i \longrightarrow^i t_2^i + t_0^i$. By the induction hypothesis, $E_0^{i-\circ j}[t_1^i + t_0^i] \longrightarrow^j E_0^{i-\circ j}[t_2^i + t_0^i]$. Thus $E_0^{i-\circ j}[\Box + t_0^i][t_1^i] \longrightarrow^j E_0^{i-\circ j}[\Box + t_0^i][t_2^i]$.

Case 9. ($E = E_0^{i-\circ j}[v_0^i + \Box]$). By (plusR-i), $v_0^i + t_1^i \longrightarrow^i v_0^i + t_2^i$. By the induction hypothesis, $E_0^{i-\circ j}[v_0^i + t_1^i] \longrightarrow^j E_0^{i-\circ j}[v_0^i + t_2^i]$. Thus $E_0^{i-\circ j}[v_0^i + \Box][t_1^i] \longrightarrow^j E_0^{i-\circ j}[v_0^i + \Box][t_2^i]$.

Therefore, if $t_1^i \longrightarrow^i t_2^i$ and $E \in \text{ECXT}^{i-\circ j}$, then $E^{i-\circ j}[t_1^i] \longrightarrow^j E^{i-\circ j}[t_2^i]$. □

Corollary 291. *If $t_1^i \longmapsto^i t_2^i$, then $t_1^i \longrightarrow^i t_2^i$.*

Proof. Suppose $t_1^i \longmapsto^i t_2^i$. Let $t_1^i = E_0^{j-\circ i}[t_{01}^j]$, $t_2^i = E_0^{j-\circ i}[t_{02}^j]$ and $t_{01}^j \mathscr{R}^j t_{02}^j$. Observe that $t_{01}^j \mathscr{R}^j t_{02}^j$ implies $t_{01}^j \longrightarrow^j t_{02}^j$. Let $E = E_0^{j-\circ i}$ in Lemma 290. We get $E_0^{j-\circ i}[t_{01}^j] \longrightarrow^i E_0^{j-\circ i}[t_{02}^j]$. Hence $t_1^i \longrightarrow^i t_2^i$. □

Theorem 292. *$t_1^i \longrightarrow^i t_2^i$ if and only if $t_1^i \longmapsto^i t_2^i$.*

Proof. This theorem follows Corollaries 289 and 291 directly. □

Theorem 293. *$t_1^i \longrightarrow^{i*} t_2^i$ if and only if $t_1^i \longmapsto^{i*} t_2^i$.*

Proof. We first show that if $t_1^i \longrightarrow^{i*} t_2^i$ then $t_1^i \longmapsto^{i*} t_2^i$. Suppose $t_1^i \longrightarrow^{i(n)} t_2^i$. We proceed by induction on n.

Case 1. When $n = 0$, $t_1^i = t_2^i$. We have $t_1^i \longmapsto^{i*} t_2^i$ immediately.

Case 2. Let $t_1^i \longrightarrow^i t_3^i \longrightarrow^{i(n)} t_2^i$.
Given $t_1^i \longrightarrow^i t_3^i$, by Corollary 289, $t_1^i \longmapsto^i t_3^i$.
Given $t_3^i \longrightarrow^{i(n)} t_2^i$, by the induction hypothesis, $t_3^i \longmapsto^{i*} t_2^i$.
We get $t_1^i \longmapsto^i t_3^i \longmapsto^{i*} t_2^i$. Hence $t_1^i \longmapsto^{i*} t_2^i$.

Now we show that if $t_1^i \longmapsto^{i*} t_2^i$ then $t_1^i \longrightarrow^{i*} t_2^i$. Suppose $t_1^i \longmapsto^{i(n)} t_2^i$. We proceed by induction on n.

Case 1. When $n = 0$, $t_1^i = t_2^i$. We have $t_1^i \longrightarrow^{i*} t_2^i$ immediately.

Case 2. Let $t_1^i \longmapsto^i t_3^i \longmapsto^{i(n)} t_2^i$.
Given $t_1^i \longmapsto^i t_3^i$, by Corollary 291, $t_1^i \longrightarrow^i t_3^i$.
Given $t_3^i \longmapsto^{i(n)} t_2^i$, by the induction hypothesis, $t_3^i \longrightarrow^{i*} t_2^i$.
We get $t_1^i \longrightarrow^i t_3^i \longrightarrow^{i*} t_2^i$. Hence $t_1^i \longrightarrow^{i*} t_2^i$.

Therefore, $t_1^i \longrightarrow^{i*} t_2^i$ if and only if $t_1^i \longmapsto^{i*} t_2^i$. □

Theorem 294 (Kleene Equality of Evaluators)**.** *For any $t \in \text{PRGM}_{\text{MetaML}}$, $eval_{\text{MetaML:SubSOS}}(t)$ is Kleene equal to $eval_{\text{MetaML:SubRed}}(t)$.*

Proof. We first show if $eval_{\text{MetaML:SubSOS}}(t) = a$ where $a \in \text{ANS}_{\text{MetaML}}$, then $eval_{\text{MetaML:SubRed}}(t) = a$.

Case 1. If $eval_{\text{MetaML:SubSOS}}(t) = \texttt{function}$, then $t \longrightarrow^{0*} \lambda x.t'^0$. By Theorem 293, $t \longmapsto^{0*} \lambda x.t'^0$.
We have $eval_{\text{MetaML:SubRed}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:SubSOS}}(t) = \texttt{code}$, then $t \longrightarrow^{0*} \langle v^1 \rangle$. By Theorem 293, $t \longmapsto^{0*} \langle v^1 \rangle$.
We have $eval_{\text{MetaML:SubRed}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:SubSOS}}(t) = n$, then $t \longrightarrow^{0*} n$. By Theorem 293, $t \longmapsto^{0*} n$.
We have $eval_{\text{MetaML:SubRed}}(t) = n$.

We then show if $eval_{\text{MetaML:SubRed}}(t) = a$ where $a \in \text{ANS}_{\text{MetaML}}$, then $eval_{\text{MetaML:SubSOS}}(t) = a$.

Case 1. If $eval_{\text{MetaML:SubRed}}(t) = \texttt{function}$, then $t \longmapsto^{0*} \lambda x.t'^{0}$. By Theorem 293, $t \longrightarrow^{0*} \lambda x.t'^{0}$.
We have $eval_{\text{MetaML:SubSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:SubRed}}(t) = \texttt{code}$, then $t \longmapsto^{0*} \langle v^1 \rangle$. By Theorem 293, $t \longrightarrow^{0*} \langle v^1 \rangle$.
We have $eval_{\text{MetaML:SubSOS}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:SubRed}}(t) = n$, then $t \longmapsto^{0*} n$. By Theorem 293, $t \longrightarrow^{0*} n$.
We have $eval_{\text{MetaML:SubSOS}}(t) = n$.

We observe that $eval_{\text{MetaML:SubSOS}}(t)$ is undefined if and only if $eval_{\text{MetaML:SubRed}}(t)$ is undefined. Therefore, $eval_{\text{MetaML:SubSOS}}(t)$ is Kleene equal to $eval_{\text{MetaML:SubRed}}(t)$. □

C.2 Equivalence of Substitutional Reduction Semantics and Substitutional Abstract Machine (MK Machine) of MetaML

We demonstrate the equivalence of the substitutional reduction semantics of MetaML and the substitutional abstract machine (the MK machine) of Environmental MetaML.

Lemma 295. $\langle i,\, E^{i-\circ 0},\, v^i \rangle_{\text{f}} \longmapsto^*_{\text{mk}} \langle i,\, E^{i-\circ 0},\, v^i \rangle_{\text{b}}$.

Proof. We proceed by induction on the structure of the derivation of $v^i \in \text{VALUE}^i$.

Case 1. ($x \in \text{VALUE}^{i+1}$). We immediately have $\langle i+1,\, E^{(i+1)-\circ 0},\, x \rangle_{\text{f}} \longmapsto_{\text{mk}} \langle i,\, E^{(i+1)-\circ 0},\, x \rangle_{\text{b}}$.

Case 2. ($v_1^{i+1}\ v_2^{i+1} \in \text{VALUE}^{i+1}$). We have:

$$
\begin{array}{lll}
 & \langle i+1,\, E^{(i+1)-\circ 0},\, v_1^{i+1}\ v_2^{i+1} \rangle_{\text{f}} & \\
\longmapsto_{\text{mk}} & \langle i+1,\, E^{(i+1)-\circ 0}[\Box\ v_2^{i+1}],\, v_1^{i+1} \rangle_{\text{f}} & \\
\longmapsto^*_{\text{mk}} & \langle i+1,\, E^{(i+1)-\circ 0}[\Box\ v_2^{i+1}],\, v_1^{i+1} \rangle_{\text{b}} & \text{by the induction hypothesis} \\
\longmapsto_{\text{mk}} & \langle i+1,\, E^{(i+1)-\circ 0}[v_1^{i+1}\ \Box],\, v_2^{i+1} \rangle_{\text{f}} & \\
\longmapsto^*_{\text{mk}} & \langle i+1,\, E^{(i+1)-\circ 0}[v_1^{i+1}\ \Box],\, v_2^{i+1} \rangle_{\text{b}} & \text{by the induction hypothesis} \\
\longmapsto_{\text{mk}} & \langle i+1,\, E^{(i+1)-\circ 0},\, v_1^{i+1}\ v_2^{i+1} \rangle_{\text{b}} &
\end{array}
$$

Case 3. ($\lambda x.t_1^0 \in \text{VALUE}^0$). We immediately have $\langle 0,\, E^{0-\circ 0},\, \lambda x.t_1^0 \rangle_{\text{f}} \longmapsto_{\text{mk}} \langle 0,\, E^{0-\circ 0},\, \lambda x.t_1^0 \rangle_{\text{b}}$.

Case 4. ($\lambda x.v_1^{i+1} \in \text{VALUE}^{i+1}$). We have:

$$
\begin{array}{lll}
 & \langle i+1,\, E^{(i+1)-\circ 0},\, \lambda x.v_1^{i+1} \rangle_{\text{f}} & \\
\longmapsto_{\text{mk}} & \langle i+1,\, E^{(i+1)-\circ 0}[\lambda x.\Box],\, v_1^{i+1} \rangle_{\text{f}} & \\
\longmapsto^*_{\text{mk}} & \langle i+1,\, E^{(i+1)-\circ 0}[\lambda x.\Box],\, v_1^{i+1} \rangle_{\text{b}} & \text{by the induction hypothesis} \\
\longmapsto_{\text{mk}} & \langle i+1,\, E^{(i+1)-\circ 0},\, \lambda x.v_1^{i+1} \rangle_{\text{b}} &
\end{array}
$$

Case 5. ($\langle v_1^{i+1} \rangle \in \text{VALUE}^i$). We have:

$$\begin{array}{rll} & \langle i,\, E^{i-\circ 0},\, \langle v_1^{i+1} \rangle \rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E^{i-\circ 0}[\langle \Box \rangle],\, v_1^{i+1} \rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}}^{*} & \langle i+1,\, E^{i-\circ 0}[\langle \Box \rangle],\, v_1^{i+1} \rangle_{\mathrm{b}} & \text{by the induction hypothesis} \\ \longmapsto_{\mathrm{mk}} & \langle i,\, E^{i-\circ 0},\, \langle v_1^{i+1} \rangle \rangle_{\mathrm{b}} & \end{array}$$

Case 6. ($\sim v_1^{i+1} \in \text{VALUE}^{i+2}$). We have:

$$\begin{array}{rll} & \langle i+2,\, E^{(i+2)-\circ 0},\, \sim v_1^{i+1} \rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E^{(i+2)-\circ 0}[\sim \Box],\, v_1^{i+1} \rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}}^{*} & \langle i+1,\, E^{(i+2)-\circ 0}[\sim \Box],\, v_1^{i+1} \rangle_{\mathrm{b}} & \text{by the induction hypothesis} \\ \longmapsto_{\mathrm{mk}} & \langle i+2,\, E^{(i+2)-\circ 0},\, \sim v_1^{i+1} \rangle_{\mathrm{b}} & \end{array}$$

Case 7. ($!v_1^{i+1} \in \text{VALUE}^{i+1}$). We have:

$$\begin{array}{rll} & \langle i+1,\, E^{(i+1)-\circ 0},\, !v_1^{i+1} \rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E^{(i+1)-\circ 0}[!\Box],\, v_1^{i+1} \rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}}^{*} & \langle i+1,\, E^{(i+1)-\circ 0}[!\Box],\, v_1^{i+1} \rangle_{\mathrm{b}} & \text{by the induction hypothesis} \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E^{(i+1)-\circ 0},\, !v_1^{i+1} \rangle_{\mathrm{b}} & \end{array}$$

Case 8. ($n \in \text{VALUE}^i$). We immediately have $\langle i,\, E^{i-\circ 0},\, n \rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}} \langle i,\, E^{i-\circ 0},\, n \rangle_{\mathrm{b}}$.

□

Lemma 296. *If* $t_1^i \mathrel{\mathscr{R}^i} t_2^i$, *then* $\langle i,\, E^{i-\circ 0},\, t_1^i \rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^{*} \langle i,\, E^{i-\circ 0},\, t_1^i \rangle_{\mathrm{r}} \longmapsto_{\mathrm{mk}} \langle i,\, E^{i-\circ 0},\, t_2^i \rangle_{\mathrm{f}}$.

Proof. We proceed by cases on $t_1^i \mathrel{\mathscr{R}^i} t_2^i$.

Case 1. (app-0). Let $i = 0$, $t_1^0 = (\lambda x.t_{11}^0)\ v_{12}^0$ and $t_2^0 = t_{11}^0[v_{12}^0/x]$. We have:

$$\begin{array}{rll} & \langle 0,\, E^{0-\circ 0},\, (\lambda x.t_{11}^0)\ v_{12}^0 \rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0}[\Box\ v_{12}^0],\, \lambda x.t_{11}^0 \rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0}[\Box\ v_{12}^0],\, \lambda x.t_{11}^0 \rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0}[(\lambda x.t_{11}^0)\ \Box],\, v_{12}^0 \rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}}^{*} & \langle 0,\, E^{0-\circ 0}[(\lambda x.t_{11}^0)\ \Box],\, v_{12}^0 \rangle_{\mathrm{b}} & \text{by Lemma 295} \\ \longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0},\, (\lambda x.t_{11}^0)\ v_{12}^0 \rangle_{\mathrm{r}} & \\ \longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0},\, t_{11}^0[v_{12}^0/x] \rangle_{\mathrm{f}} & \end{array}$$

Case 2. (run-0). Let $i = 0$, $t_1^0 = !\langle v_{11}^1 \rangle$ and $t_2^0 = v_{11}^1$. We have:

$$
\begin{array}{ll}
 & \langle 0,\, E^{0-\circ 0},\, !\langle v_{11}^1\rangle\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0}[!\Box],\, \langle v_{11}^1\rangle\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mk}} & \langle 1,\, E^{0-\circ 0}[!\Box][\langle\Box\rangle],\, v_{11}^1\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mk}}^{*} & \langle 1,\, E^{0-\circ 0}[!\Box][\langle\Box\rangle],\, v_{11}^1\rangle_{\mathrm{b}} \quad \text{by Lemma 295} \\
\longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0}[!\Box],\, \langle v_{11}^1\rangle\rangle_{\mathrm{b}} \\
\longmapsto_{\mathrm{mk}}^{*} & \langle 0,\, E^{0-\circ 0},\, !\langle v_{11}^1\rangle\rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0},\, v_{11}^1\rangle_{\mathrm{f}}
\end{array}
$$

Case 3. (splice-1). Let $i = 1$, $t_1^1 = {\sim}\langle v_{11}^1\rangle$ and $t_2^1 = v_{11}^1$. We have:

$$
\begin{array}{ll}
 & \langle 1,\, E^{0-\circ 0},\, {\sim}\langle v_{11}^1\rangle\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0}[{\sim}\Box],\, \langle v_{11}^1\rangle\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mk}} & \langle 1,\, E^{0-\circ 0}[{\sim}\Box][\langle\Box\rangle],\, v_{11}^1\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mk}}^{*} & \langle 1,\, E^{0-\circ 0}[{\sim}\Box][\langle\Box\rangle],\, v_{11}^1\rangle_{\mathrm{b}} \quad \text{by Lemma 295} \\
\longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0}[{\sim}\Box],\, \langle v_{11}^1\rangle\rangle_{\mathrm{b}} \\
\longmapsto_{\mathrm{mk}}^{*} & \langle 1,\, E^{0-\circ 0},\, {\sim}\langle v_{11}^1\rangle\rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{mk}} & \langle 1,\, E^{0-\circ 0},\, v_{11}^1\rangle_{\mathrm{f}}
\end{array}
$$

Case 4. (plus-0). Let $i = 0$, $t_1^0 = n_1 + n_2$ and $t_2^0 = n$ where $n = n_1 + n_2$. We have:

$$
\begin{array}{lll}
 & \langle 0,\, E^{0-\circ 0},\, n_1 + n_2\rangle_{\mathrm{f}} & \\
\longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0}[\Box + n_2],\, n_1\rangle_{\mathrm{f}} & \\
\longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0}[\Box + n_2],\, n_1\rangle_{\mathrm{b}} & \\
\longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0}[n_1 + \Box],\, n_2\rangle_{\mathrm{f}} & \\
\longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0}[n_1 + \Box],\, n_2\rangle_{\mathrm{b}} & \\
\longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0},\, n_1 + n_2\rangle_{\mathrm{r}} & \\
\longmapsto_{\mathrm{mk}} & \langle 0,\, E^{0-\circ 0},\, n\rangle_{\mathrm{f}} & \text{where } n = n_1 + n_2
\end{array}
$$

□

Lemma 297. *If* $t^i = E_1^{j-\circ i}[t_1^j]$ *and* $t_1^j \mathrel{\mathscr{R}^j} t_2^j$, *then* $\langle i,\, E^{i-\circ 0},\, t^i\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^{*} \langle j,\, E^{i-\circ 0}E_1^{j-\circ i},\, t_1^j\rangle_{\mathrm{f}}$.

Proof. Suppose $t^i = E_1^{j-\circ i}[t_1^j]$ and $t_1^j \mathrel{\mathscr{R}^j} t_2^j$. We want to show $\langle i,\, E^{i-\circ 0},\, E_1^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^{*} \langle j,\, E^{i-\circ 0}E_1^{j-\circ i},\, t_1^j\rangle$. We proceed by induction on the structure of the derivation of $E_1^{j-\circ i}$.

Case 1. ($E_1^{i-\circ i} = \Box^{i-\circ i}$). We know $\langle i,\, E^{i-\circ 0},\, t^i\rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, t_1^i\rangle_{\mathrm{f}}$ and $\langle i,\, E^{i-\circ 0},\, t_1^i\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^{*} \langle i,\, E^{i-\circ 0},\, t_1^i\rangle_{\mathrm{f}}$.

Case 2. ($E_1^{j-\circ i} = E_{11}^{j-\circ i}\, t_{11}^i$). We know $\langle i,\, E^{i-\circ 0},\, t^i\rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, E_1^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, (E_{11}^{j-\circ i}\, t_{11}^i)[t_1^j]\rangle_{\mathrm{f}}$. We have:

$$
\begin{array}{ll}
 & \langle i,\, E^{i-\circ 0},\, (E_{11}^{j-\circ i}\, t_{11}^i)[t_1^j]\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mk}} & \langle i,\, E^{i-\circ 0}[\Box\, t_{11}^i],\, E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}}
\end{array}
$$

Since $E_{11}^{j-\circ i}$ is a component of $E_1^{j-\circ i}$, by the induction hypothesis, we have $\langle i,\, E^{i-\circ 0}[\Box\, t_{11}^i],\, E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^* \langle i,\, E^{i-\circ 0}[E_{11}^{j-\circ i}\, t_{11}^i],\, t_1^j\rangle_{\mathrm{f}}$.

Case 3. ($E_1^{j-\circ i} = v_{11}^i\, E_{11}^{j-\circ i}$). We know $\langle i,\, E^{i-\circ 0},\, t^i\rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, E_1^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, (v_{11}^i\, E_{11}^{j-\circ i})[t_1^j]\rangle_{\mathrm{f}}$. We have:

$$\begin{array}{rl} & \langle i,\, E^{i-\circ 0},\, (v_{11}^i\, E_{11}^{j-\circ i})[t_1^j]\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mk}} & \langle i,\, E^{i-\circ 0}[\Box\, E_{11}^{j-\circ i}[t_1^j]],\, v_{11}^i\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mk}}^* & \langle i,\, E^{i-\circ 0}[\Box\, E_{11}^{j-\circ i}[t_1^j]],\, v_{11}^i\rangle_{\mathrm{b}} \quad \text{by Lemma 295} \\ \longmapsto_{\mathrm{mk}} & \langle i,\, E^{i-\circ 0}[v_{11}^i\, \Box],\, E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \end{array}$$

Since $E_{11}^{j-\circ i}$ is a component of $E_1^{j-\circ i}$, by the induction hypothesis, we have $\langle i,\, E^{i-\circ 0}[v_{11}^i\, \Box],\, E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^* \langle i,\, E^{i-\circ 0}[v_{11}^i\, E_{11}^{j-\circ i}],\, t_1^j\rangle_{\mathrm{f}}$.

Case 4. ($E_1^{j-\circ(i+1)} = \lambda x.E_{11}^{j-\circ(i+1)}$). We know $\langle i+1,\, E^{(i+1)-\circ 0},\, t^{i+1}\rangle_{\mathrm{f}} = \langle i+1,\, E^{i+1-\circ 0},\, E_1^{j-\circ(i+1)}[t_1^j]\rangle_{\mathrm{f}} = \langle i+1,\, E^{(i+1)-\circ 0},\, (\lambda x.E_{11}^{j-\circ(i+1)})[t_1^j]\rangle_{\mathrm{f}}$. We have:

$$\begin{array}{rl} & \langle i+1,\, E^{(i+1)-\circ 0},\, (\lambda x.E_{11}^{j-\circ(i+1)})[t_1^j]\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E^{(i+1)-\circ 0}[\lambda x.\Box],\, E_{11}^{j-\circ(i+1)}[t_1^j]\rangle_{\mathrm{f}} \end{array}$$

Since $E_{11}^{j-\circ(i+1)}$ is a component of $E_1^{j-\circ(i+1)}$, by the induction hypothesis, we have $\langle i+1,\, E^{(i+1)-\circ 0}[\lambda x.\Box],\, E_{11}^{j-\circ(i+1)}[t_1^j]\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^* \langle i+1,\, E^{(i+1)-\circ 0}[\lambda x.E_{11}^{j-\circ(i+1)}],\, t_1^j\rangle_{\mathrm{f}}$.

Case 5. ($E_1^{j-\circ i} = \langle E_{11}^{j-\circ(i+1)}\rangle$). We know $\langle i,\, E^{i-\circ 0},\, t^i\rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, E_1^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, \langle E_{11}^{j-\circ(i+1)}\rangle[t_1^j]\rangle_{\mathrm{f}}$. We have:

$$\begin{array}{rl} & \langle i,\, E^{i-\circ 0},\, \langle E_{11}^{j-\circ(i+1)}\rangle[t_1^j]\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mk}} & \langle i,\, E^{i-\circ 0}[\langle\Box\rangle],\, E_{11}^{j-\circ(i+1)}[t_1^j]\rangle_{\mathrm{f}} \end{array}$$

Since $E_{11}^{j-\circ(i+1)}$ is a component of $E_1^{j-\circ i}$, by the induction hypothesis, we have $\langle i,\, E^{i-\circ 0}[\langle\Box\rangle],\, E_{11}^{j-\circ(i+1)}[t_1^j]\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^* \langle i,\, E^{i-\circ 0}[\langle E_{11}^{j-\circ(i+1)}\rangle],\, t_1^j\rangle_{\mathrm{f}}$.

Case 6. ($E_1^{j-\circ(i+1)} = {\sim}E_{11}^{j-\circ i}$). We know:

$$\begin{array}{rl} & \langle i+1,\, E^{(i+1)-\circ 0},\, t^{i+1}\rangle_{\mathrm{f}} \\ = & \langle i+1,\, E^{(i+1)-\circ 0},\, E_1^{j-\circ(i+1)}[t_1^j]\rangle_{\mathrm{f}} \\ = & \langle i+1,\, E^{(i+1)-\circ 0},\, {\sim}E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \end{array}$$

We have:

$$\begin{array}{rl} & \langle i+1,\, E^{(i+1)-\circ 0},\, {\sim}E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E^{(i+1)-\circ 0}[{\sim}\Box],\, E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \end{array}$$

Since $E_{11}^{j-\circ i}$ is a component of $E_1^{j-\circ(i+1)}$, by the induction hypothesis,
we have $\langle i+1,\ E^{(i+1)-\circ 0}[\sim\Box],\ E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^* \langle i+1,\ E^{(i+1)-\circ 0}[\sim E_{11}^{j-\circ i}],\ t_1^j\rangle_{\mathrm{f}}$.

Case 7. ($E_1^{j-\circ i} = !E_{11}^{j-\circ i}$). We know $\langle i,\ E^{i-\circ 0},\ t^i\rangle_{\mathrm{f}} = \langle i,\ E^{i-\circ 0},\ E_1^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} = \langle i,\ E^{i-\circ 0},\ !E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}}$. We have:

$$\begin{array}{rl} & \langle i,\ E^{i-\circ 0},\ !E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mk}} & \langle i,\ E^{i-\circ 0}[!\Box],\ E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \end{array}$$

Since $E_{11}^{j-\circ i}$ is a component of $E_1^{j-\circ i}$, by the induction hypothesis, we have $\langle i,\ E^{i-\circ 0}[!\Box],\ E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^*$ $\langle i,\ E^{i-\circ 0}[!E_{11}^{j-\circ i}],\ t_1^j\rangle_{\mathrm{f}}$.

Case 8. ($E_1^{j-\circ i} = E_{11}^{j-\circ i} + t_{11}^i$). We know $\langle i,\ E^{i-\circ 0},\ t^i\rangle_{\mathrm{f}} = \langle i,\ E^{i-\circ 0},\ E_1^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} = \langle i,\ E^{i-\circ 0},\ (E_{11}^{j-\circ i} + t_{11}^i)[t_1^j]\rangle_{\mathrm{f}}$.
We have:

$$\begin{array}{rl} & \langle i,\ E^{i-\circ 0},\ (E_{11}^{j-\circ i} + t_{11}^i)[t_1^j]\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mk}} & \langle i,\ E^{i-\circ 0}[\Box + t_{11}^i],\ E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \end{array}$$

Since $E_{11}^{j-\circ i}$ is a component of $E_1^{j-\circ i}$, by the induction hypothesis, we have $\langle i,\ E^{i-\circ 0}[\Box + t_{11}^i],\ E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^*$ $\langle i,\ E^{i-\circ 0}[E_{11}^{j-\circ i} + t_{11}^i],\ t_1^j\rangle_{\mathrm{f}}$.

Case 9. ($E_1^{j-\circ i} = v_{11}^i + E_{11}^{j-\circ i}$). We know $\langle i,\ E^{i-\circ 0},\ t^i\rangle_{\mathrm{f}} = \langle i,\ E^{i-\circ 0},\ E_1^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} = \langle i,\ E^{i-\circ 0},\ (v_{11}^i + E_{11}^{j-\circ i})[t_1^j]\rangle_{\mathrm{f}}$.
We have:

$$\begin{array}{rll} & \langle i,\ E^{i-\circ 0},\ (v_{11}^i + E_{11}^{j-\circ i})[t_1^j]\rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}} & \langle i,\ E^{i-\circ 0}[\Box + E_{11}^{j-\circ i}[t_1^j]],\ v_{11}^i\rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}}^* & \langle i,\ E^{i-\circ 0}[\Box + E_{11}^{j-\circ i}[t_1^j]],\ v_{11}^i\rangle_{\mathrm{b}} & \text{by Lemma 295} \\ \longmapsto_{\mathrm{mk}} & \langle i,\ E^{i-\circ 0}[v_{11}^i + \Box],\ E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} & \end{array}$$

Since $E_{11}^{j-\circ i}$ is a component of $E_1^{j-\circ i}$, by the induction hypothesis, we have $\langle i,\ E^{i-\circ 0}[v_{11}^i + \Box],\ E_{11}^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^*$ $\langle i,\ E^{i-\circ 0}[v_{11}^i + E_{11}^{j-\circ i}],\ t_1^j\rangle_{\mathrm{f}}$.

□

Lemma 298. *If* $E_0^{i-\circ 0}[t_0^i] = E_1^{j-\circ 0}[t_1^j]$ *and* $E_1^{j-\circ 0}[t_1^j] \longmapsto E_1^{j-\circ 0}[t_2^j]$ *where* $t_1^j \mathrel{\mathscr{R}^j} t_2^j$, *then* $\langle i,\ E_0^{i-\circ 0},\ t_0^i\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^*$ $\langle j,\ E_1^{j-\circ 0},\ t_2^j\rangle_{\mathrm{f}}$.

Proof. If t_1^j is inside t_0^i (or the same as t_0^i), $E_1^{j-\circ 0}$ extends $E_0^{i-\circ 0}$ (or is the same as $E_0^{i-\circ 0}$). Otherwise, because t_0^i is not reduced, it must be a value.

Case 1. Suppose t_1^j is inside t_0^i (or the same as t_0^i). Let $t_0^i = E_2^{j-\circ i}[t_1^j]$. Then $E_1^{j-\circ 0} = E_0^{i-\circ 0}E_2^{j-\circ i}$. We have $\langle i,\ E_0^{i-\circ 0},\ t_0^i\rangle_{\mathrm{f}} = \langle i,\ E_0^{i-\circ 0},\ E_2^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}}$. By Lemma 297, $\langle i,\ E_0^{i-\circ 0},\ E_2^{j-\circ i}[t_1^j]\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^* \langle j,\ E_0^{i-\circ 0}E_2^{j-\circ i},\ t_1^j\rangle_{\mathrm{f}}$.
By Lemma 296, $\langle j,\ E_0^{i-\circ 0}E_2^{j-\circ i},\ t_1^j\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^* \langle j,\ E_0^{i-\circ 0}E_2^{j-\circ i},\ t_1^j\rangle_{\mathrm{r}} \longmapsto_{\mathrm{mk}} \langle j,\ E_0^{i-\circ 0}E_2^{j-\circ i},\ t_2^j\rangle_{\mathrm{f}}$.

Case 2. Otherwise, $t_0^i \in \text{VALUE}^i$. By Lemma 295, $\langle i,\ E_0^{i-\circ 0},\ t_0^i\rangle_{\text{f}} \longmapsto_{\text{mk}}^* \langle i,\ E_0^{i-\circ 0},\ t_0^i\rangle_{\text{b}}$. We prove the following statement by induction on the structure of the derivation of $E_0^{i-\circ 0} \in \text{ECXT}^{i-\circ 0}$.

Statement: If $E_0^{i-\circ 0}[t_0^i] = E_1^{j-\circ 0}[t_1^j]$ where $t_0^i \in \text{VALUE}^i$ and $E_1^{j-\circ 0}[t_1^j] \longmapsto E_1^{j-\circ 0}[t_2^j]$ where $t_1^j \mathrel{\mathscr{R}^j} t_2^j$, then $\langle i,\ E_0^{i-\circ 0},\ t_0^i\rangle_{\text{b}} \longmapsto_{\text{mk}}^* \langle j,\ E_1^{j-\circ 0},\ t_2^j\rangle_{\text{f}}$.

Case i. ($E_0^{0-\circ 0} = \Box^{0-\circ 0}$). This case is vacuous.

Case ii. ($E_0^{i-\circ 0} = E_2^{i-\circ 0}[\Box\ t_{22}^i]$). We have:

$$\begin{array}{ll} & \langle i,\ E_0^{i-\circ 0},\ t_0^i\rangle_{\text{b}} \\ = & \langle i,\ E_2^{i-\circ 0}[\Box\ t_{22}^i],\ t_0^i\rangle_{\text{b}} \\ \longmapsto_{\text{mk}} & \langle i,\ E_2^{i-\circ 0}[t_0^i\ \Box],\ t_{22}^i\rangle_{\text{f}} \end{array}$$

Case a. If $t_{22}^i \mathrel{\mathscr{R}^i} t_{23}^i$, then $t_1^i = t_{22}^i$ and $t_2^i = t_{23}^i$.
By Lemma 296, $\langle i,\ E_2^{i-\circ 0}[t_0^i\ \Box],\ t_{22}^i\rangle_{\text{f}} \longmapsto_{\text{mk}}^* \langle i,\ E_2^{i-\circ 0}[t_0^i\ \Box],\ t_2^i\rangle_{\text{f}}$.

Case b. If $t_{22}^i \not\mathrel{\mathscr{R}^i}$ and $t_{22}^i \in \text{VALUE}^i$.

Case 1. If $t_{22} \in \text{VALUE}^0$. We have:

$$\begin{array}{ll} & \langle 0,\ E_2^{0-\circ 0}[t_0^0\ \Box],\ t_{22}^0\rangle_{\text{f}} \\ \longmapsto_{\text{mk}} & \langle 0,\ E_2^{0-\circ 0}[t_0^0\ \Box],\ t_{22}^0\rangle_{\text{b}} \\ \longmapsto_{\text{mk}} & \langle 0,\ E_2^{0-\circ 0},\ t_0^0\ t_{22}^0\rangle_{\text{r}} \end{array}$$

Then $t_1^0 = t_0^0\ t_{22}^0$. By Lemma 296, $\langle 0,\ E_2^{0-\circ 0},\ t_0^0\ t_{22}^0\rangle_{\text{r}} \longmapsto_{\text{mk}} \langle 0,\ E_2^{0-\circ 0},\ t_2^0\rangle_{\text{f}}$.

Case 2. If $t_{22} \in \text{VALUE}^{i+1}$. We have:

$$\begin{array}{ll} & \langle i+1,\ E_2^{(i+1)-\circ 0}[t_0^{i+1}\ \Box],\ t_{22}^{i+1}\rangle_{\text{f}} \\ \longmapsto_{\text{mk}}^* & \langle i+1,\ E_2^{(i+1)-\circ 0}[t_0^{i+1}\ \Box],\ t_{22}^{i+1}\rangle_{\text{b}} \quad \text{by Lemma 295} \\ \longmapsto_{\text{mk}} & \langle i+1,\ E_2^{(i+1)-\circ 0},\ t_0^{i+1}\ t_{22}^{i+1}\rangle_{\text{b}} \end{array}$$

We know $t_0^{i+1}\ t_{22}^{i+1} \in \text{VALUE}^{i+1}$. Since $E_2^{(i+1)-\circ 0}$ is a component of $E_0^{(i+1)-\circ 0}$, by the induction hypothesis, $\langle i+1,\ E_2^{(i+1)-\circ 0},\ t_0^{i+1}\ t_{22}^{i+1}\rangle_{\text{b}} \longmapsto_{\text{mk}}^* \langle j,\ E_1^{j-\circ 0},\ t_2^j\rangle_{\text{f}}$.

Case c. If $t_{22}^i \not\mathrel{\mathscr{R}^i}$ and $t_{22}^i \notin \text{VALUE}^i$, then $t_{22}^i = E_3^{j-\circ i}[t_1^j]$. Hence $E_1^{j-\circ 0} = E_2^{i-\circ 0}[t_0^i\ E_3^{j-\circ i}]$.
We have:

$$\begin{array}{lll} & \langle i+1,\ E_2^{(i+1)-\circ 0}[t_0^{i+1}\ \Box],\ t_{22}^{i+1}\rangle_{\text{f}} & \\ & \langle i,\ E_2^{i-\circ 0}[t_0^i\ \Box],\ t_{22}^i\rangle_{\text{f}} & \\ = & \langle i,\ E_2^{i-\circ 0}[t_0^i\ \Box],\ E_3^{j-\circ i}[t_1^j]\rangle_{\text{f}} & \\ \longmapsto_{\text{mk}}^* & \langle j,\ E_2^{i-\circ 0}[t_0^i\ E_3^{j-\circ i}],\ t_1^j\rangle_{\text{f}} & \text{by Lemma 297} \\ \longmapsto_{\text{mk}}^* & \langle j,\ E_2^{i-\circ 0}[t_0^i\ E_3^{j-\circ i}],\ t_2^j\rangle_{\text{f}} & \text{by Lemma 296} \end{array}$$

Case iii. ($E_0^{i-\circ 0} = E_2^{i-\circ 0}[v_{21}^i\ \Box]$).

Case a. If $t_0 \in \text{VALUE}^0$. We have:

$$\begin{array}{ll} & \langle 0,\ E_0^{0-\circ 0},\ t_0^0\rangle_{\text{b}} \\ = & \langle 0,\ E_2^{0-\circ 0}[v_{21}^0\ \Box],\ t_0^0\rangle_{\text{b}} \\ \longmapsto_{\text{mk}} & \langle 0,\ E_2^{0-\circ 0},\ v_{21}^0\ t_0^0\rangle_{\text{r}} \end{array}$$

Then $t_1^0 = v_{21}^0\, t_0^0$. By Lemma 296, $\langle 0,\, E_2^{0-\circ 0},\, v_{21}^0\, t_0^0 \rangle_{\mathrm{r}} \longmapsto_{\mathrm{mk}} \langle 0,\, E_2^{0-\circ 0},\, t_2^0 \rangle_{\mathrm{f}}$.

Case b. If $t_0 \in \textsc{Value}^{i+1}$. We have:

$$\begin{array}{rl} & \langle i+1,\, E_0^{(i+1)-\circ 0},\, t_0^{i+1} \rangle_{\mathrm{b}} \\ = & \langle i+1,\, E_2^{(i+1)-\circ 0}[v_{21}^{i+1}\, \Box],\, t_0^{i+1} \rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E_2^{(i+1)-\circ 0},\, v_{21}^{i+1}\, t_0^{i+1} \rangle_{\mathrm{b}} \end{array}$$

We know $v_{21}^{i+1}\, t_0^{i+1} \in \textsc{Value}^{i+1}$. Since $E_2^{(i+1)-\circ 0}$ is a component of $E_0^{(i+1)-\circ 0}$, by the induction hypothesis, $\langle i+1,\, E_2^{(i+1)-\circ 0},\, v_{21}^{i+1}\, t_0^{i+1} \rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}}^{*} \langle j,\, E_1^{j-\circ 0},\, t_2^j \rangle_{\mathrm{f}}$.

Case iv. $(E_0^{(i+1)-\circ 0} = E_2^{(i+1)-\circ 0}[\lambda x.\Box])$. We have:

$$\begin{array}{rl} & \langle i+1,\, E_0^{(i+1)-\circ 0},\, t_0^{i+1} \rangle_{\mathrm{b}} \\ = & \langle i+1,\, E_2^{(i+1)-\circ 0}[\lambda x.\Box],\, t_0^{i+1} \rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E_2^{(i+1)-\circ 0},\, \lambda x.t_0^{i+1} \rangle_{\mathrm{b}} \end{array}$$

We know $\lambda x.t_0^{i+1} \in \textsc{Value}^{i+1}$. Since $E_2^{(i+1)-\circ 0}$ is a component of $E_0^{(i+1)-\circ 0}$, by the induction hypothesis, $\langle i+1,\, E_2^{(i+1)-\circ 0},\, \lambda x.t_0^{i+1} \rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}}^{*} \langle j,\, E_1^{j-\circ 0},\, t_2^j \rangle_{\mathrm{f}}$.

Case v. $(E_0^{(i+1)-\circ 0} = E_2^{i-\circ 0}[\langle \Box \rangle])$. We have:

$$\begin{array}{rl} & \langle i+1,\, E_0^{(i+1)-\circ 0},\, t_0^{i+1} \rangle_{\mathrm{b}} \\ = & \langle i+1,\, E_2^{i-\circ 0}[\langle \Box \rangle],\, t_0^{i+1} \rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle i,\, E_2^{i-\circ 0},\, \langle t_0^{i+1} \rangle \rangle_{\mathrm{b}} \end{array}$$

We know $\langle t_0^{i+1} \rangle \in \textsc{Value}^{i}$. Since $E_2^{i-\circ 0}$ is a component of $E_0^{(i+1)-\circ 0}$, by the induction hypothesis, $\langle i,\, E_2^{i-\circ 0},\, \langle t_0^{i+1} \rangle \rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}}^{*} \langle j,\, E_1^{j-\circ 0},\, t_2^j \rangle_{\mathrm{f}}$.

Case vi. $(E_0^{i-\circ 0} = E_2^{(i+1)-\circ 0}[\sim \Box])$.

Case a. If $t_0 \in \textsc{Value}^0$. We have:

$$\begin{array}{rl} & \langle 0,\, E_0^{0-\circ 0},\, t_0^0 \rangle_{\mathrm{b}} \\ = & \langle 0,\, E_2^{1-\circ 0}[\sim \Box],\, t_0^0 \rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle 1,\, E_2^{1-\circ 0},\, \sim t_0^0 \rangle_{\mathrm{r}} \end{array}$$

Then $t_1^1 = \sim t_0^0$. By Lemma 296, $\langle 1,\, E_2^{1-\circ 0},\, \sim t_0^0 \rangle_{\mathrm{r}} \longmapsto_{\mathrm{mk}} \langle 1,\, E_2^{1-\circ 0},\, t_2^1 \rangle_{\mathrm{f}}$.

Case b. If $t_0 \in \textsc{Value}^{i+1}$. We have:

$$\begin{array}{rl} & \langle i+1,\, E_0^{(i+1)-\circ 0},\, t_0^{i+1} \rangle_{\mathrm{b}} \\ = & \langle i+1,\, E_2^{(i+2)-\circ 0}[\sim \Box],\, t_0^{i+1} \rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle i+2,\, E_2^{(i+2)-\circ 0},\, \sim t_0^{i+1} \rangle_{\mathrm{b}} \end{array}$$

We know $\sim t_0^{i+1} \in \textsc{Value}^{i+2}$. Since $E_2^{(i+2)-\circ 0}$ is a component of $E_0^{(i+1)-\circ 0}$, by the induction hypothesis, $\langle i+2,\, E_2^{(i+2)-\circ 0},\, \sim t_0^{i+1} \rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}}^{*} \langle j,\, E_1^{j-\circ 0},\, t_2^j \rangle_{\mathrm{f}}$.

Case vii. $(E_0^{i-\circ 0} = E_2^{i-\circ 0}[!\Box])$.

Case a. If $t_0 \in \textsc{Value}^0$. We have:

$$\begin{array}{rl} & \langle 0,\, E_0^{0-\circ 0},\, t_0^0 \rangle_{\mathrm{b}} \\ = & \langle 0,\, E_2^{0-\circ 0}[!\Box],\, t_0^0 \rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle 0,\, E_2^{0-\circ 0},\, !t_0^0 \rangle_{\mathrm{r}} \end{array}$$

Then $t_1^0 = !t_0^0$. By Lemma 296, $\langle 0,\, E_2^{0\multimap 0},\, !t_0^0\rangle_{\mathrm{r}} \longmapsto_{\mathrm{mk}} \langle 0,\, E_2^{0\multimap 0},\, t_2^0\rangle_{\mathrm{f}}$.

Case b. If $t_0 \in \text{VALUE}^{i+1}$. We have:

$$\begin{array}{ll} & \langle i+1,\, E_0^{(i+1)\multimap 0},\, t_0^{i+1}\rangle_{\mathrm{b}} \\ = & \langle i+1,\, E_2^{(i+1)\multimap 0}[!\Box],\, t_0^{i+1}\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E_2^{(i+1)\multimap 0},\, !t_0^{i+1}\rangle_{\mathrm{b}} \end{array}$$

We know $!t_0^{i+1} \in \text{VALUE}^{i+1}$. Since $E_2^{(i+1)\multimap 0}$ is a component of $E_0^{(i+1)\multimap 0}$, by the induction hypothesis, $\langle i+1,\, E_2^{(i+1)\multimap 0},\, !t_0^{i+1}\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}}^{*} \langle j,\, E_1^{j\multimap 0},\, t_2^j\rangle_{\mathrm{f}}$.

Case viii. $(E_0^{i\multimap 0} = E_2^{i\multimap 0}[\Box + t_{22}^i])$. We have:

$$\begin{array}{ll} & \langle i,\, E_0^{i\multimap 0},\, t_0^i\rangle_{\mathrm{b}} \\ = & \langle i,\, E_2^{i\multimap 0}[\Box + t_{22}^i],\, t_0^i\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle i,\, E_2^{i\multimap 0}[t_0^i + \Box],\, t_{22}^i\rangle_{\mathrm{f}} \end{array}$$

Case a. If $t_{22}^i \mathrel{\mathscr{R}^i} t_{23}^i$, then $t_1^i = t_{22}^i$ and $t_2^i = t_{23}^i$.
By Lemma 296, $\langle i,\, E_2^{i\multimap 0}[t_0^i + \Box],\, t_{22}^i\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mk}}^{*} \langle i,\, E_2^{i\multimap 0}[t_0^i + \Box],\, t_2^i\rangle_{\mathrm{f}}$.

Case b. If $t_{22}^i \not\mathrel{\mathscr{R}^i}$ and $t_{22}^i \in \text{VALUE}^i$.

Case 1. If $t_{22} \in \text{VALUE}^0$. We have:

$$\begin{array}{ll} & \langle 0,\, E_2^{0\multimap 0}[t_0^0 + \Box],\, t_{22}^0\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mk}} & \langle 0,\, E_2^{0\multimap 0}[t_0^0 + \Box],\, t_{22}^0\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle 0,\, E_2^{0\multimap 0},\, t_0^0 + t_{22}^0\rangle_{\mathrm{r}} \end{array}$$

Then $t_1^0 = t_0^0 + t_{22}^0$. By Lemma 296, $\langle 0,\, E_2^{0\multimap 0},\, t_0^0 + t_{22}^0\rangle_{\mathrm{r}} \longmapsto_{\mathrm{mk}} \langle 0,\, E_2^{0\multimap 0},\, t_2^0\rangle_{\mathrm{f}}$.

Case 2. If $t_{22} \in \text{VALUE}^{i+1}$. We have:

$$\begin{array}{lll} & \langle i+1,\, E_2^{(i+1)\multimap 0}[t_0^{i+1} + \Box],\, t_{22}^{i+1}\rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}}^{*} & \langle i+1,\, E_2^{(i+1)\multimap 0}[t_0^{i+1} + \Box],\, t_{22}^{i+1}\rangle_{\mathrm{b}} & \text{by Lemma 295} \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E_2^{(i+1)\multimap 0},\, t_0^{i+1} + t_{22}^{i+1}\rangle_{\mathrm{b}} & \end{array}$$

We know $t_0^{i+1} + t_{22}^{i+1} \in \text{VALUE}^{i+1}$. Since $E_2^{(i+1)\multimap 0}$ is a component of $E_0^{(i+1)\multimap 0}$, by the induction hypothesis, $\langle i+1,\, E_2^{(i+1)\multimap 0},\, t_0^{i+1} + t_{22}^{i+1}\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}}^{*} \langle j,\, E_1^{j\multimap 0},\, t_2^j\rangle_{\mathrm{f}}$.

Case c. If $t_{22}^i \not\mathrel{\mathscr{R}^i}$ and $t_{22}^i \notin \text{VALUE}^i$, then $t_{22}^i = E_3^{j\multimap i}[t_1^j]$. Hence $E_1^{j\multimap 0} = E_2^{i\multimap 0}[t_0^i + E_3^{j\multimap i}]$. We have:

$$\begin{array}{lll} & \langle i+1,\, E_2^{(i+1)\multimap 0}[t_0^{i+1} + \Box],\, t_{22}^{i+1}\rangle_{\mathrm{f}} & \\ & \langle i,\, E_2^{i\multimap 0}[t_0^i + \Box],\, t_{22}^i\rangle_{\mathrm{f}} & \\ = & \langle i,\, E_2^{i\multimap 0}[t_0^i + \Box],\, E_3^{j\multimap i}[t_1^j]\rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}}^{*} & \langle j,\, E_2^{i\multimap 0}[t_0^i + E_3^{j\multimap i}],\, t_1^j\rangle_{\mathrm{f}} & \text{by Lemma 297} \\ \longmapsto_{\mathrm{mk}}^{*} & \langle j,\, E_2^{i\multimap 0}[t_0^i + E_3^{j\multimap i}],\, t_2^j\rangle_{\mathrm{f}} & \text{by Lemma 296} \end{array}$$

Case ix. $(E_0^{i\multimap 0} = E_2^{i\multimap 0}[v_{21}^i + \Box])$.

Case a. If $t_0 \in \text{VALUE}^0$. We have:

$$\begin{array}{lll} & \langle 0,\, E_0^{0-\circ 0},\, t_0^0\rangle_{\mathrm{b}} \\ = & \langle 0,\, E_2^{0-\circ 0}[v_{21}^0+\Box],\, t_0^0\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle 0,\, E_2^{0-\circ 0},\, v_{21}^0+t_0^0\rangle_{\mathrm{r}} \end{array}$$

Then $t_1^0 = v_{21}^0 + t_0^0$. By Lemma 296, $\langle 0,\, E_2^{0-\circ 0},\, v_{21}^0+t_0^0\rangle_{\mathrm{r}} \longmapsto_{\mathrm{mk}} \langle 0,\, E_2^{0-\circ 0},\, t_2^0\rangle_{\mathrm{f}}$.

Case b. If $t_0 \in \mathrm{VALUE}^{i+1}$. We have:

$$\begin{array}{lll} & \langle i+1,\, E_0^{(i+1)-\circ 0},\, t_0^{i+1}\rangle_{\mathrm{b}} \\ = & \langle i+1,\, E_2^{(i+1)-\circ 0}[v_{21}^{i+1}+\Box],\, t_0^{i+1}\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E_2^{(i+1)-\circ 0},\, v_{21}^{i+1}+t_0^{i+1}\rangle_{\mathrm{b}} \end{array}$$

We know $v_{21}^{i+1}+t_0^{i+1} \in \mathrm{VALUE}^{i+1}$. Since $E_2^{(i+1)-\circ 0}$ is a component of $E_0^{(i+1)-\circ 0}$, by the induction hypothesis, $\langle i+1,\, E_2^{(i+1)-\circ 0},\, v_{21}^{i+1}+t_0^{i+1}\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}}^* \langle j,\, E_1^{j-\circ 0},\, t_2^j\rangle_{\mathrm{f}}$.

□

Lemma 299. *If* $v^0 = E^{i-\circ 0}[t^i]$, *then* $\langle i,\, E^{i-\circ 0},\, t^i\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}}^* v^0$.

Proof. Suppose $v^0 = E^{i-\circ 0}[t^i]$. We know $t^i \in \mathrm{VALUE}^i$. We proceed by induction on the structure of the derivation of $E^{i-\circ 0} \in \mathrm{ECXT}^{i-\circ 0}$. □

Case 1. ($E^{0-\circ 0} = \Box^{0-\circ 0}$). Then $v^0 = t^0$. We have $\langle 0,\, \Box^{0-\circ 0},\, v^0\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}} v^0$.

Case 2. ($E^{i-\circ 0} = E_1^{i-\circ 0}[\Box\ t_{12}^i]$). Then $v^0 = E_1^{i-\circ 0}[t^i\ t_{12}^i]$. We know $t^i \in \mathrm{VALUE}^i$, $t_{12}^i \in \mathrm{VALUE}^i$ and $i \geq 1$. Let's use $i+1$ instead of i. We have:

$$\begin{array}{lll} & \langle i+1,\, E_1^{(i+1)-\circ 0}[\Box\ t_{12}^{i+1}],\, t^{i+1}\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E_1^{(i+1)-\circ 0}[t^{i+1}\ \Box],\, t_{12}^{i+1}\rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}}^* & \langle i+1,\, E_1^{(i+1)-\circ 0}[t^{i+1}\ \Box],\, t_{12}^{i+1}\rangle_{\mathrm{b}} & \text{by Lemma 295} \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E_1^{(i+1)-\circ 0},\, t^{i+1}\ t_{12}^{i+1}\rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}}^* & \langle i+1,\, E_1^{(i+1)-\circ 0},\, t^{i+1}\ t_{12}^{i+1}\rangle_{\mathrm{b}} & \text{by Lemma 295} \end{array}$$

Since $E_1^{(i+1)-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i+1,\, E_1^{(i+1)-\circ 0},\, t^{i+1}\ t_{12}^{i+1}\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}}^* v^0$.

Case 3. ($E^{i-\circ 0} = E_1^{i-\circ 0}[v_{11}^i\ \Box]$). Then $v^0 = E_1^{i-\circ 0}[v_{11}^i\ t^i]$. We know $t^i \in \mathrm{VALUE}^i$ and $i \geq 1$. Let's use $i+1$ instead of i. We have:

$$\begin{array}{lll} & \langle i+1,\, E_1^{(i+1)-\circ 0}[v_{11}^{i+1}\ \Box],\, t^{i+1}\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E_1^{(i+1)-\circ 0},\, v_{11}^{i+1}\ t^{i+1}\rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}}^* & \langle i+1,\, E_1^{(i+1)-\circ 0},\, v_{11}^{i+1}\ t^{i+1}\rangle_{\mathrm{b}} & \text{by Lemma 295} \end{array}$$

Since $E_1^{(i+1)-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i+1,\, E_1^{(i+1)-\circ 0},\, v_{11}^{i+1}\ t^{i+1}\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}}^* v^0$.

Case 4. $(E^{(i+1)-\circ 0} = E_1^{(i+1)-\circ 0}[\lambda x.\Box])$. Then $v^0 = E_1^{(i+1)-\circ 0}[\lambda x.t^{i+1}]$. We know $t^{i+1} \in \text{VALUE}^{i+1}$. We have:

$$\begin{array}{rl} & \langle i+1,\, E_1^{(i+1)-\circ 0}[\lambda x.\Box],\, t^{i+1}\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E_1^{(i+1)-\circ 0},\, \lambda x.t^{i+1}\rangle_{\mathrm{b}} \end{array}$$

Since $E_1^{(i+1)-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i+1,\, E_1^{(i+1)-\circ 0},\, \lambda x.t^{i+1}\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}}^* v^0$.

Case 5. $(E^{(i+1)-\circ 0} = E_1^{i-\circ 0}[\langle\Box\rangle])$. Then $v^0 = E_1^{i-\circ 0}[\langle t^{i+1}\rangle]$. We know $t^{i+1} \in \text{VALUE}^{i+1}$. We have:

$$\begin{array}{rl} & \langle i+1,\, E_1^{i-\circ 0}[\langle\Box\rangle],\, t^{i+1}\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle i,\, E_1^{i-\circ 0},\, \langle t^{i+1}\rangle\rangle_{\mathrm{b}} \end{array}$$

Since $E_1^{i-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i,\, E_1^{i-\circ 0},\, \langle t^{i+1}\rangle\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}}^* v^0$.

Case 6. $(E^{i-\circ 0} = E_1^{(i+1)-\circ 0}[\sim\Box])$. Then $v^0 = E_1^{(i+1)-\circ 0}[\sim t^i]$. We know $t^i \in \text{VALUE}^i$ and $i \geq 1$. Let use $i+1$ instead of i and $i+2$ instead of $i+1$. We have:

$$\begin{array}{rl} & \langle i+1,\, E_1^{(i+2)-\circ 0}[\sim\Box],\, t^{i+1}\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle i+2,\, E_1^{(i+2)-\circ 0},\, \sim t^{i+1}\rangle_{\mathrm{b}} \end{array}$$

Since $E_1^{(i+2)-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i+2,\, E_1^{(i+2)-\circ 0},\, \sim t^{i+1}\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}}^* v^0$.

Case 7. $(E^{i-\circ 0} = E_1^{i-\circ 0}[!\Box])$. Then $v^0 = E_1^{i-\circ 0}[!t^i]$. We know $t^i \in \text{VALUE}^i$ and $i \geq 1$. Let use $i+1$ instead of i. We have:

$$\begin{array}{rl} & \langle i+1,\, E_1^{(i+1)-\circ 0}[!\Box],\, t^{i+1}\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E_1^{(i+1)-\circ 0},\, !t^{i+1}\rangle_{\mathrm{b}} \end{array}$$

Since $E_1^{(i+1)-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i+1,\, E_1^{(i+1)-\circ 0},\, !t^{i+1}\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}}^* v^0$.

Case 8. $(E^{i-\circ 0} = E_1^{i-\circ 0}[\Box + t_{12}^i])$. Then $v^0 = E_1^{i-\circ 0}[t^i + t_{12}^i]$. We know $t^i \in \text{VALUE}^i$, $t_{12}^i \in \text{VALUE}^i$ and $i \geq 1$. Let's use $i+1$ instead of i. We have:

$$\begin{array}{rll} & \langle i+1,\, E_1^{(i+1)-\circ 0}[\Box + t_{12}^{i+1}],\, t^{i+1}\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E_1^{(i+1)-\circ 0}[t^{i+1} + \Box],\, t_{12}^{i+1}\rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}}^* & \langle i+1,\, E_1^{(i+1)-\circ 0}[t^{i+1} + \Box],\, t_{12}^{i+1}\rangle_{\mathrm{b}} & \text{by Lemma 295} \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E_1^{(i+1)-\circ 0},\, t^{i+1} + t_{12}^{i+1}\rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mk}}^* & \langle i+1,\, E_1^{(i+1)-\circ 0},\, t^{i+1} + t_{12}^{i+1}\rangle_{\mathrm{b}} & \text{by Lemma 295} \end{array}$$

Since $E_1^{(i+1)-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i+1,\, E_1^{(i+1)-\circ 0},\, t^{i+1} + t_{12}^{i+1}\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{mk}} v^0$.

Case 9. ($E^{i-\circ 0} = E_1^{i-\circ 0}[v_{11}^i + \Box]$). Then $v^0 = E_1^{i-\circ 0}[v_{11}^i + t^i]$. We know $t^i \in \textsc{Value}^i$ and $i \geq 1$. Let's use $i+1$ instead of i. We have:

$$\begin{array}{ll} & \langle i+1,\, E_1^{(i+1)-\circ 0}[v_{11}^{i+1} + \Box],\, t^{i+1}\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mk}} & \langle i+1,\, E_1^{(i+1)-\circ 0},\, v_{11}^{i+1} + t^{i+1}\rangle_{\mathrm{f}} \\ \longmapsto^*_{\mathrm{mk}} & \langle i+1,\, E_1^{(i+1)-\circ 0},\, v_{11}^{i+1} + t^{i+1}\rangle_{\mathrm{b}} \qquad \text{by Lemma 295} \end{array}$$

Since $E_1^{(i+1)-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i+1,\, E_1^{(i+1)-\circ 0},\, v_{11}^{i+1} + t^{i+1}\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{mk}} v^0$.

Lemma 300. *If* $v^0 = E^{i-\circ 0}[t^i]$, *then* $\langle i,\, E^{i-\circ 0},\, t^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} v^0$.

Proof. Suppose $v^0 = E^{i-\circ 0}[t^i]$. Then $t^i \in \textsc{Value}^i$. By Lemma 295, $\langle i,\, E^{i-\circ 0},\, t^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} \langle i,\, E^{i-\circ 0},\, t^i\rangle_{\mathrm{b}}$. By Lemma 299, $\langle i,\, E^{i-\circ 0},\, t^i\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mk}} v^0$. Hence $\langle i,\, E^{i-\circ 0},\, t^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} v^0$. □

Lemma 301. *If* $E^{i-\circ 0}[t_1^i] \longmapsto^* v_2^0$, *then* $\langle i,\, E^{i-\circ 0},\, t_1^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} v_2^0$.

Proof. Suppose $E^{i-\circ 0}[t_1^i] \longmapsto^{(n)} v_2^0$. We proceed by induction on n.

Case 1. When $n = 0$, $v_2^0 = E^{i-\circ 0}[t_1^i]$. By Lemma 300, $\langle i,\, E^{i-\circ 0},\, t_1^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} v_2^0$.

Case 2. Let $E^{i-\circ 0}[t_1^i] \longmapsto E_1^{j-\circ 0}[t_2^j] \longmapsto^{(n)} v_2^0$, where $E^{i-\circ 0}[t_1^i] = E_1^{j-\circ 0}[t_{11}^j]$ and $t_{11}^j \mathrel{\mathscr{R}^j} t_2^j$.
By Lemma 298, $\langle i,\, E^{i-\circ 0},\, t_1^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} \langle j,\, E_1^{j-\circ 0},\, t_2^j\rangle_{\mathrm{f}}$. Given $E_1^{j-\circ 0}[t_2^j] \longmapsto^{(n)} v_2^0$, by the induction hypothesis, $\langle j,\, E_1^{j-\circ 0},\, t_2^j\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} v_2^0$. Hence we have $\langle i,\, E^{i-\circ 0},\, t_1^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} v_2^0$.

□

Theorem 302 (Soundness of Substitutional Abstract Machine w.r.t. Substitutional Reduction Semantics)**.** *For any* $t_1 \in \textsc{Prgm}_{\mathrm{MetaML}}$, *if* $t_1^0 \longmapsto^* v_2^0$, *then* $\langle 0,\, \Box^{0-\circ 0},\, t_1^0\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} v_2^0$.

Proof. Suppose $\Box^{0-\circ 0}[t_1^0] \longmapsto^* v_2^0$, by Lemma 301 , $\langle 0,\, \Box^{0-\circ 0},\, t_1^0\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} v_2^0$. □

Any machine configuration in the MK machine can be translated to its corresponding representation as a term in (Substitutional) MetaML at level 0.

Definition 303 (Translator)**.** Define the translator $\mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}$ to be a total function from the set of machine configurations $\textsc{Cfg}$ to the set of level 0 terms $\textsc{Term}^0$.

$$\mathscr{T}_{\mathrm{mk}\to\mathrm{sub}} : \textsc{Cfg} \to \textsc{Term}^0$$

$$\begin{array}{rcl} \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(\langle i,\, E^{i-\circ 0},\, t^i\rangle_{\mathrm{f}}) & = & E^{i-\circ 0}[t^i] \\ \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(\langle i,\, E^{i-\circ 0},\, v^i\rangle_{\mathrm{b}}) & = & E^{i-\circ 0}[v^i] \\ \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(\langle i,\, E^{i-\circ 0},\, t^i\rangle_{\mathrm{r}}) & = & E^{i-\circ 0}[t^i] \\ \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(v^0) & = & v^0 \end{array}$$

Lemma 304. *If* $C_1 \longmapsto_{\mathrm{mk}} C_2$, *then* $\mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_1) \longmapsto^{0*} \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_2)$.

Proof. We proceed by cases on $C_1 \longmapsto_{\mathrm{mk}} C_2$. □

Case 1. Reduce rules: Let $C_1 = \langle i,\, E^{i\multimap 0},\, t_1^i\rangle_{\mathrm{r}}$ and $C_2 = \langle i,\, E^{i\multimap 0},\, t_2^i\rangle_{\mathrm{f}}$. Then $E^{i\multimap 0}[t_1] \longmapsto^0 E^{i\multimap 0}[t_2]$ where $t_1^i \mathscr{R}^i t_2^i$. Hence $\mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_1) \longmapsto^0 \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_2)$.

Case 2. Focus rules: Let $C_1 = \langle i,\, E^{i\multimap 0},\, t_1^i\rangle_{\mathrm{f}}$ and $C_2 = \langle i,\, E^{i\multimap 0},\, t_2^i\rangle_{?}$. Then $E^{i\multimap 0}[t_1] = E^{i\multimap 0}[t_2]$.
Hence $\mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_1) \longmapsto^{0*} \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_2)$.

Case 3. Build rules:

Case i. (b-value-0). Let $C_1 = \langle 0,\, \square,\, v^0\rangle_{\mathrm{b}}$ and $C_2 = v^0$. Then $\square[v^0] = v^0$. Hence $\mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_1) \longmapsto^{0*} \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_2)$.

Case ii. (other rules). Let $C_1 = \langle i,\, E^{i\multimap 0},\, t_1^i\rangle_{\mathrm{b}}$ and $C_2 = \langle i,\, E^{i\multimap 0},\, t_2^i\rangle_{?}$. Then $E^{i\multimap 0}[t_1] = E^{i\multimap 0}[t_2]$. Hence $\mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_1) \longmapsto^{0*} \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_2)$.

Lemma 305. *If* $C_1 \longmapsto^*_{\mathrm{mk}} C_2$, *then* $\mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_1) \longmapsto^{0*} \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_2)$.

Proof. Suppose $C_1 \longmapsto^{(n)}_{\mathrm{mk}} C_2$. We proceed by induction on n.

Case 1. When $n = 0$, $C_1 = C_2$. Then $\mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_1) = \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_2)$. We have $\mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_1) \longmapsto^{0*} \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_2)$ immediately.

Case 2. Let $C_1 \longmapsto_{\mathrm{mk}} C_3 \longmapsto^{(n)}_{\mathrm{mk}} C_2$.
Given $C_1 \longmapsto_{\mathrm{mk}} C_3$, by Lemma 304, $\mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_1) \longmapsto^{0*} \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_3)$.
Given $C_3 \longmapsto^{(n)}_{\mathrm{mk}} C_2$, by the induction hypothesis, $\mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_3) \longmapsto^{0*} \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_2)$.
Hence $\mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_1) \longmapsto^{0*} \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(C_2)$.

□

Theorem 306 (Completeness of Substitutional Abstract Machine w.r.t. Substitutional Reduction Semantics)**.** *For any* $t_1^0 \in \mathrm{PRGM}_{\mathrm{MetaML}}$, *if* $\langle 0,\, \square,\, t_1^0\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} v_2^0$, *then* $t_1^0 \longmapsto^{0*} v_2^0$.

Proof. If $\langle 0,\, \square,\, t_1^0\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} v_2^0$, by Lemma 305, $\mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(\langle 0,\, \square,\, t_1^0\rangle_{\mathrm{f}}) \longmapsto^{0*} \mathscr{T}_{\mathrm{mk}\to\mathrm{sub}}(v_2^0)$. We have $t_1^0 \longmapsto^{0*} v_2^0$. □

Theorem 307 (Kleene Equality of Evaluators)**.** *For any* $t \in \mathrm{PRGM}_{\mathrm{MetaML}}$, $eval_{\mathrm{MetaML:SubRed}}(t)$ *is Kleene equal to* $eval_{\mathrm{MetaML:MK}}(t)$.

Proof. We first show if $eval_{\mathrm{MetaML:SubRed}}(t) = a$ where $a \in \mathrm{ANS}_{\mathrm{MetaML}}$, then $eval_{\mathrm{MetaML:MK}}(t) = a$.

Case 1. If $eval_{\mathrm{MetaML:SubRed}}(t) = \texttt{function}$, then $t \longmapsto^{0*} \lambda x.t'^0$. By Theorem 302, $\langle 0,\, \square,\, t\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mk}} \lambda x.t'^0$. We have $eval_{\mathrm{MetaML:MK}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:SubRed}}(t) = \texttt{code}$, then $t \longmapsto^{0*} \langle v^1 \rangle$. By Theorem 302, $\langle 0, \Box, t \rangle_{\text{f}} \longmapsto^{*}_{\text{mk}} \langle v^1 \rangle$. We have $eval_{\text{MetaML:MK}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:SubRed}}(t) = n$, then $t \longmapsto^{0*} n$. By Theorem 302, $\langle 0, \Box, t \rangle_{\text{f}} \longmapsto^{*}_{\text{mk}} n$. We have $eval_{\text{MetaML:MK}}(t) = n$.

We then show if $eval_{\text{MetaML:MK}}(t) = a$ where $a \in \text{ANS}_{\text{MetaML}}$, then $eval_{\text{MetaML:SubRed}}(t) = a$.

Case 1. If $eval_{\text{MetaML:MK}}(t) = \texttt{function}$, then $\langle 0, \Box, t \rangle_{\text{f}} \longmapsto^{*}_{\text{mk}} \lambda x.t'^{0}$. By Theorem 306, $t \longmapsto^{0*} \lambda x.t'^{0}$. We have $eval_{\text{MetaML:SubRed}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:MK}}(t) = \texttt{code}$, then $\langle 0, \Box, t \rangle_{\text{f}} \longmapsto^{*}_{\text{mk}} \langle v^1 \rangle$. By Theorem 306, $t \longmapsto^{0*} \langle v^1 \rangle$. We have $eval_{\text{MetaML:SubRed}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:MK}}(t) = n$, then $\langle 0, \Box, t \rangle_{\text{f}} \longmapsto^{*}_{\text{mk}} n$. By Theorem 306, $t \longmapsto^{0*} n$. We have $eval_{\text{MetaML:SubRed}}(t) = n$.

We observe that $eval_{\text{MetaML:SubRed}}(t)$ is undefined if and only if $eval_{\text{MetaML:MK}}(t)$ is undefined. Therefore, $eval_{\text{MetaML:SubRed}}(t)$ is Kleene equal to $eval_{\text{MetaML:MK}}(t)$. □

Appendix D

Proofs of Chapter 5

D.1 Equivalence of MetaML and Explicit MetaML

We demonstrate the equivalence of the substitutional structural operational semantics of MetaML and the structural operational semantics of Explicit MetaML. We use subscripts "$_{\text{sub}}$" and "$_{\text{exp}}$" to differentiate the syntax of (Substitutional) MetaML from the syntax of Explicit MetaML.

D.1.1 Bisimulation Relation

Definition 308 (Bisimulation Relation)**.** Define the bisimulation relation $\simeq$ to be a binary relation up to alpha equivalence on the set of terms in (Substitutional) MetaML and the set of runtime terms in Explicit MetaML.

$$\simeq \subseteq \text{TERM}^i_{\text{sub}} \times \text{RTERM}^i_{\text{exp}}$$

$$\frac{}{x \simeq x}\ \text{(var-sim)} \qquad \frac{t^i_{a_1} \simeq t^i_{b_1} \quad t^i_{a_2} \simeq t^i_{b_2}}{t^i_{a_1}\ t^i_{a_2} \simeq t^i_{b_1}\ t^i_{b_2}}\ \text{(app-sim)} \qquad \frac{t^i_a \simeq t^i_b}{(\lambda x.t^i_a) \simeq (\lambda x.t^i_b)}\ \text{(lam-sim)}$$

$$\frac{t^0_a \simeq t^0_b}{(\lambda x.t^0_a) \simeq (\underline{\lambda} x.t^0_b)}\ \text{(lamu-sim)} \qquad \frac{t^{i+1}_a \simeq t^{i+1}_b}{\langle t^{i+1}_a \rangle \simeq \langle t^{i+1}_b \rangle}\ \text{(code-sim)} \qquad \frac{t^i_a \simeq t^i_b}{{\sim}t^i_a \simeq {\sim}t^i_b}\ \text{(splice-sim)}$$

$$\frac{t^i_a \simeq t^i_b}{!t^i_a \simeq !t^i_b}\ \text{(run-sim)} \qquad \frac{}{\overline{n} \simeq \overline{n}}\ \text{(num-sim)} \qquad \frac{t^i_{a_1} \simeq t^i_{b_1} \quad t^i_{a_2} \simeq t^i_{b_2}}{t^i_{a_1} + t^i_{a_2} \simeq t^i_{b_1} + t^i_{b_2}}\ \text{(plus-sim)}$$

$$\frac{t^i_a \simeq t^i_b \quad w_a \simeq w_b}{t^i_a[w_a/x] \simeq t^i_b[x := w_b]}\ \text{(subst-sim)}$$

Remark 309. The bisimulation relation $\sim$ is up to alpha equivalence. We immediately have: (1) if $t_{a_1} \simeq t_b$ and $t_{a_1} \sim_\alpha t_{a_2}$ then $t_{a_2} \simeq t_b$, and (2) if $t_a \simeq t_{b_1}$ and $t_{b_1} \sim_\alpha t_{b_2}$ then $t_a \simeq t_{b_2}$.

D.1.2 Unload Function

Definition 310 (Unload Function)**.** Let $i \in \mathbb{N}$. Define the unload function U to be a total function from the set of Explicit MetaML runtime terms $\text{RTERM}^i_{\text{exp}}$ to the set of Substitutional MetaML terms $\text{TERM}^i_{\text{sub}}$.

$$U \;:\; \text{RTERM}^i_{\text{exp}} \to \text{TERM}^i_{\text{sub}}$$

$$\begin{array}{rcl}
U(x) & = & x \\
U(t_1\, t_2) & = & U(t_1)\, U(t_2) \\
U(\underline{\lambda} x.t) & = & \lambda x.U(t) \\
U(\lambda x.t) & = & \lambda x.U(t) \\
U(\langle t \rangle) & = & \langle U(t) \rangle \\
U(\sim t) & = & \sim U(t) \\
U(!t) & = & !U(t) \\
U(n) & = & n \\
U(t_1 + t_2) & = & U(t_1) + U(t_2) \\
U(t[x := w]) & = & U(t)[U(w)/x]
\end{array}$$

Lemma 311 (Equality of Related Terms w.r.t. Unload Function)**.** *If $t^i_a \simeq t^i_b$, then $t^i_a = U(t^i_b)$.*

Proof. We proceed by structural induction on $t^i_a \simeq t^i_b$.

Case 1. (var-sim). Let $t^i_a = t^i_b = x$. We immediately get $x = U(x)$.

Case 2. (app-sim). Let $t^i_a = t^i_{a_1}\, t^i_{a_2}$ and $t^i_b = t^i_{b_1}\, t^i_{b_2}$ where $t^i_{a_1} \simeq t^i_{b_2}$ and $t^i_{a_2} \simeq t^i_{b_2}$. By the induction hypothesis, $t^i_{a_1} = U(t^i_{b_1})$ and $t^i_{a_2} = U(t^i_{b_2})$. Hence $U(t^i_{b_1}\, t^i_{b_2}) = U(t^i_{b_1})\, U(t^i_{b_2}) = t^i_{a_1}\, t^i_{a_2}$.

Case 3. (lam-sim). Let $t^i_a = \lambda x.t^i_{a_1}$ and $t^i_b = \lambda x.t^i_{b_1}$ where $t^i_{a_1} \simeq t^i_{b_1}$. By the induction hypothesis, $t^i_{a_1} = U(t^i_{b_1})$. Hence $U(\lambda x.t^i_{b_1}) = \lambda x.U(t^i_{b_1}) = \lambda x.t^i_{a_1}$.

Case 4. (lamu-sim). Let $t^i_a = \lambda x.t^0_{a_1}$ and $t^i_b = \underline{\lambda} x.t^0_{b_1}$ where $t^0_{a_1} \simeq t^0_{b_1}$. By the induction hypothesis, $t^0_{a_1} = U(t^0_{b_1})$. Hence $U(\underline{\lambda} x.t^0_{b_1}) = \lambda x.U(t^0_{b_1}) = \lambda x.t^0_{a_1}$.

Case 5. (code-sim). Let $t^i_a = \langle t^{i+1}_{a_1} \rangle$ and $t^i_b = \langle t^{i+1}_{b_1} \rangle$ where $t^{i+1}_{a_1} \simeq t^{i+1}_{b_1}$. By the induction hypothesis, $t^{i+1}_{a_1} = U(t^{i+1}_{b_1})$. Hence, $U(\langle t^{i+1}_{b_1} \rangle) = \langle U(t^{i+1}_{b_1}) \rangle = \langle t^{i+1}_{a1} \rangle$.

Case 6. (splice-sim). Let $t^{i+1}_a = \sim t^i_{a_1}$ and $t^{i+1}_b = \sim t^i_{b_1}$ where $t^i_{a_1} \simeq t^i_{b_1}$. By the induction hypothesis, $t^i_{a_1} = U(t^i_{b_1})$. Hence, $U(\sim t^i_{b_1}) = \sim U(t^i_{b_1}) = \sim t^i_{a_1}$.

Case 7. (run-sim). Let $t^i_a = !t^i_{a_1}$ and $t^i_b = !t^i_{b_1}$ where $t^i_{a_1} \simeq t^i_{b_1}$. By the induction hypothesis, $t^i_{a_1} = U(t^i_{b_1})$. Hence, $U(!t^i_{b1}) = !U(t^i_{b1}) = !t^i_{a1}$.

Case 8. (num-sim). Let $t^i_a = t^i_b = n$. We immediately get $U(n) = n$.

Case 9. (plus-sim). Let $t^i_a = t^i_{a_1} + t^i_{a_2}$ and $t^i_b = t^i_{b_1} + t^i_{b_2}$ where $t^i_{a_1} \simeq t^i_{b_2}$ and $t^i_{a_2} \simeq t^i_{b_2}$. By the induction hypothesis, $t^i_{a_1} = U(t^i_{b_1})$ and $t^i_{a_2} = U(t^i_{b_2})$. Hence $U(t^i_{b_1} + t^i_{b_2}) = U(t^i_{b_1}) + U(t^i_{b_2}) = t^i_{a_1} + t^i_{a_2}$.

Case 10. (subst-sim). Let $t_a = t^i_{a_1}[w_{a_1}/x]$ and $t_b = t^i_{b_1}[x := w_{b_1}]$ where $t^i_{a_1} \simeq t^i_{b_1}$ and $w_{a_1} \simeq w_{b_1}$. By the induction hypothesis, $t^i_{a_1} = U(t^i_{b_1})$ and $w_{a_1} = U(w_{b_1})$. Hence $U(t^i_{b_1}[x := w_{b_1}]) = U(t^i_{b_1})[U(w_{b_1})/x] = t^i_{a_1}[w_{a_1}/x]$.

□

D.1.3 Substitution Normal Form

Definition 312 (Substitution Normal Form)**.** A term $t^i \in \text{RTERM}^i_{\text{exp}}$ is in *substitution normal form* if and only if $t^i \not\longrightarrow^{\text{x}i}$.

Remark 313. We use s with or without any subscript or superscript as a metavariable to range over the runtime terms of Explicit MetaML in substitution normal form.

Remark 314. An Explicit MetaML runtime term in substitution normal is not necessarily in the normal form with respect to the single-step relation $\longrightarrow$. For example, $(\lambda x.t)\ v$ is in substitution normal form but is not in the normal form with respect to the single-step relation $\longrightarrow$.

Lemma 315. *If* $t^i_a \simeq t^i_{b_1}[x := w_{b_1}]$, *then* $t^i_{b_1} \longrightarrow^{\text{x}i*} s^i_{b_1}$, $s^i_{b_1}[x := w_{b_1}] \longrightarrow^{\text{x}i*} s^i_{b_2}$, *and* $t^i_a \simeq s^i_{b_2}$.

Proof. We proceed by structural induction on $t^i_a \simeq t^i_{b_1}[x := w_{b_1}]$. Only (subst-sim) applies. Let $t^i_a = t^i_{a_1}[w_{a_1}/x]$ and we have

$$\frac{t^i_{a_1} \simeq t^i_{b_1} \quad w_{a_1} \simeq w_{b_1}}{t^i_{a_1}[w_{a_1}/x] \simeq t^i_{b_1}[x := w_{b_1}].}$$

We proceed by cases on $t^i_{a_1} \in \text{TERM}^i_{\text{sub}}$.

Case 1. $(t^i_{a_1} = x)$. We have

$$\frac{x \simeq t^i_{b_1} \quad w_{a_1} \simeq w_{b_1}}{x[w_{a_1}/x] \simeq t^i_{b_1}[x := w_{b_1}].}$$

Then, $x[w_{a_1}/x] = w_{a_1}$. We proceed by cases on $x \simeq t^i_{b_1}$.

Case i. (var-sim). Let $t^i_{b_1} = x$. Then, $x \longrightarrow^{\text{x}i*} x$, $x[x := w_{b_1}] \longrightarrow^{\text{x}i} w_{b_1}$ and $w_{a_1} \simeq w_{b_1}$.

Case ii. (subst-sim). Let $t^i_{b_1} = t^i_{b_{11}}[x_1 := w_{b_{11}}]$. Given $x \simeq t^i_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we get $t^i_{b_{11}} \longrightarrow^{\text{x}i*} s^i_{b_{11}}$, $s^i_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{\text{x}i*} s^i_{b_{12}}$ and $x \simeq s^i_{b_{12}}$. We proceed by cases on $x \simeq s^i_{b_{12}}$. The only case is (var-sim), so let $s^i_{b_{12}} = x$. Then, $t^i_{b_1} \longrightarrow^{\text{x}i*} x$, $x[x := w_{b_1}] \longrightarrow^{\text{x}i} w_{b_1}$ and $w_{a_1} \simeq w_{b_1}$.

Case 2. $(t^i_{a_1} = x_0$ and $x_0 \not\equiv x)$. We have

$$\frac{x_0 \simeq t^i_{b_1} \quad w_{a_1} \simeq w_{b_1}}{x_0[w_{a_1}/x] \simeq t^i_{b_1}[x := w_{b_1}].}$$

Then, $x_0[w_{a_1}/x] = x_0$. We proceed by cases on $x_0 \simeq t^i_{b_1}$.

Case i. (var-sim). Let $t^i_{b_1} = x_0$. Then, $x_0 \longrightarrow^{\text{x}i*} x_0$, $x_0[x := w_{b_1}] \longrightarrow^{\text{x}i} x_0$ and $x_0 \simeq x_0$.

Case ii. (subst-sim). Let $t^i_{b_1} = t^i_{b_{11}}[x_1 := w_{b_{11}}]$. Given $x_0 \simeq t^i_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t^i_{b_{11}} \longrightarrow^{\mathrm{x}i*} s^i_{b_{11}}$, $s^i_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i*} s^i_{b_{12}}$ and $x_0 \simeq s^i_{b_{12}}$. We proceed by cases on $x_0 \simeq s^i_{b_{12}}$. The only case is (var-sim), so let $s^i_{b_{12}} = x_0$. Then, $t^i_{b_1} \longrightarrow^{\mathrm{x}i*} x_0$, $x_0[x := w_{b_1}] \longrightarrow^{\mathrm{x}i} x_0$ and by (var-sim) $x_0 \simeq x_0$.

Case 3. $(t^i_{a_1} = (t^i_{a_{11}}\ t^i_{a_{12}}))$. We have

$$\frac{(t^i_{a_{11}}\ t^i_{a_{12}}) \simeq t^i_{b_1} \quad w_{a_1} \simeq w_{b_1}}{(t^i_{a_{11}}\ t^i_{a_{12}})[w_{a_1}/x] \simeq t^i_{b_1}[x := w_{b_1}].}$$

Then, $(t^i_{a_{11}}\ t^i_{a_{12}})[w_{a_1}/x] = (t^i_{a_{11}}[w_{a_1}/x])\ (t^i_{a_{12}}[w_{a_1}/x])$. We proceed by cases on $(t^i_{a_{11}}\ t^i_{a_{12}}) \simeq t^i_{b_1}$.

Case i. (app-sim). Let $t^i_{b_1} = (t^i_{b_{11}}\ t^i_{b_{12}})$ where $t^i_{a_{11}} \simeq t^i_{b_{11}}$ and $t^i_{a_{12}} \simeq t^i_{b_{12}}$. We have $(t^i_{b_{11}}\ t^i_{b_{12}}) \longrightarrow^{\mathrm{x}i*} (t^i_{b_{11}}\ t^i_{b_{12}})$ and $(t^i_{b_{11}}\ t^i_{b_{12}})[x := w_{b1}] \longrightarrow^{\mathrm{x}i} (t^i_{b_{11}}[x := w_{b_1}])\ (t^i_{b_{12}}[x := w_{b_1}])$. By (subst-sim) and (app-sim), we get $(t^i_{a_{11}}[w_{a_1}/x])\ (t^i_{a_{12}}[w_{a_1}/x]) \simeq (t^i_{b_{11}}[x := w_{b_1}])\ (t^i_{b_{12}}[x := w_{b_1}])$.

Case ii. (subst-sim). Let $t^i_{b_1} = t^i_{b_{11}}[x_1 := w_{b_{11}}]$. Given $(t^i_{a_{11}}\ t^i_{a_{12}}) \simeq t^i_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t^i_{b_{11}} \longrightarrow^{\mathrm{x}i*} s^i_{b_{11}}$, $s^i_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i*} s^i_{b_{12}}$, and $(t^i_{a_{11}}\ t^i_{a_{12}}) \simeq s^i_{b_{12}}$. We proceed by cases on $(t^i_{a_{11}}\ t^i_{a_{12}}) \simeq s^i_{b_{12}}$. The only case is (app-sim), so let $s^i_{b_{12}} = (t^i_{b_{121}}\ t^i_{b_{122}})$ where $t^i_{a_{11}} \simeq t^i_{b_{121}}$ and $t^i_{a_{12}} \simeq t^i_{b_{122}}$. Then, $t^i_{b_1} \longrightarrow^{\mathrm{x}i*} (t^i_{b_{121}}\ t^i_{b_{122}})$ and $(t^i_{b_{121}}\ t^i_{b_{122}})[x := w_{b_1}] \longrightarrow^{\mathrm{x}i} (t^i_{b_{121}}[x := w_{b_1}])\ (t^i_{b_{122}}[x := w_{b_1}])$. By (subst-sim) and (app-sim), we get $(t^i_{a_{11}}[w_{a_1}/x])\ (t^i_{a_{12}}[w_{a_1}/x]) \simeq (t^i_{b_{121}}[x := w_{b_1}])\ (t^i_{b_{122}}[x := w_{b_1}])$.

Case 4. $(t^i_{a_1} = \lambda x_0.t^i_{a_{11}})$. We have

$$\frac{(\lambda x_0.t^i_{a_{11}}) \simeq t^i_{b_1} \quad w_{a_1} \simeq w_{b_1}}{(\lambda x_0.t^i_{a_{11}})[w_{a_1}/x] \simeq t^i_{b_1}[x := w_{b_1}].}$$

Then, $(\lambda x_0.t^i_{a_{11}})[w_{a_1}/x] = \lambda x_1.t^i_{a_{11}}[x_1/x_0][w_{a_1}/x]$ where $x_1 \notin FV(\lambda x_0.t^i_{a_{11}}) \cup FV(w_{a_1}) \cup \{x\}$. We proceed by cases on $(\lambda x_0.t^i_{a_{11}}) \simeq t^i_{b_1}$.

Case i. (lam-sim). Let $t^i_{b_1} = \lambda x_0.t^i_{b_{11}}$ where $t^i_{a_{11}} \simeq t^i_{b_{11}}$. We have $\lambda x_0.t^i_{b_{11}} \longrightarrow^{\mathrm{x}i*} \lambda x_0.t^i_{b_{11}}$ and $(\lambda x_0.t^i_{b_{11}})[x := w_{b_1}] \longrightarrow^{\mathrm{x}i} \lambda x_2.t^i_{b_{11}}[x_0 := x_2][x := w_{b_1}]$ where $x_2 \notin FV(\lambda x_0.t^i_{b_{11}}) \cup FV(w_{b_1}) \cup \{x\}$.

Let $x_3 \notin FV(\lambda x_0.t^i_{a_{11}}) \cup FV(w_{a_1}) \cup FV(\lambda x_0.t^i_{b_{11}}) \cup FV(w_{b_1}) \cup \{x\}$, then by the definition of α-equivalence, we get $\lambda x_1.t^i_{a_{11}}[x_1/x_0][w_{a_1}/x] \sim_\alpha \lambda x_3.t^i_{a_{11}}[x_3/x_0][w_{a_1}/x]$ and $\lambda x_2.t^i_{b_{11}}[x_0 := x_2][x := w_{b_1}] \sim_\alpha \lambda x_3.t^i_{b_{11}}[x_0 := x_3][x := w_{b_1}]$.

By (lam-sim) and (subst-sim), we get $\lambda x_3.t^i_{a_{11}}[x_3/x_0][w_{a_1}/x] \simeq \lambda x_3.t^i_{b_{11}}[x_0 := x_3][x := w_{b_1}]$. Hence we have $\lambda x_1.t^i_{a_{11}}[x_1/x_0][w_{a_1}/x] \simeq \lambda x_2.t^i_{b_{11}}[x_0 := x_2][x := w_{b_1}]$.

Case ii. (lamu-sim). Let $t^i_{a_1} = \lambda x_0.t^0_{a_{11}}$ and $t^i_{b_1} = \underline{\lambda} x_0.t^0_{b_{11}}$ where $t^0_{a_{11}} \simeq t^0_{b_{11}}$. We have $\underline{\lambda} x_0.t^0_{b_{11}} \longrightarrow^{\mathrm{x}i*} \underline{\lambda} x_0.t^0_{b_{11}}$ and $(\underline{\lambda} x_0.t^0_{b_{11}})[x := w_{b1}] \longrightarrow^{\mathrm{x}i} \underline{\lambda} x_2.t^0_{b_{11}}[x_0 := x_2][x := w_{b_1}]$ where $x_2 \notin FV(\underline{\lambda} x_0.t^0_{b_{11}}) \cup FV(w_{b_1}) \cup \{x\}$.

Let $x_3 \notin FV(\lambda x_0.t^0_{a_{11}}) \cup FV(w_{a_1}) \cup FV(\underline{\lambda} x_0.t^0_{b_{11}}) \cup FV(w_{b_1}) \cup \{x\}$, then by the definition of α-equivalence, we get $\lambda x_1.t^0_{a_{11}}[x_1/x_0][w_{a_1}/x] \sim_\alpha \lambda x_3.t^0_{a_{11}}[x_3/x_0][w_{a_1}/x]$ and $\underline{\lambda} x_2.t^0_{b_{11}}[x_0 := x_2][x := w_{b_1}] \sim_\alpha \underline{\lambda} x_3.t^0_{b_{11}}[x_0 := x_3][x := w_{b_1}]$.
By (lam-sim) and (subst-sim), we get $\lambda x_3.t^0_{a_{11}}[x_3/x_0][w_{a_1}/x] \simeq \underline{\lambda} x_3.t^0_{b_{11}}[x_0 := x_3][x := w_{b_1}]$. Hence we have $\lambda x_1.t^0_{a_{11}}[x_1/x_0][w_{a_1}/x] \simeq \underline{\lambda} x_2.t^0_{b_{11}}[x_0 := x_2][x := w_{b_1}]$.

Case iii. (subst-sim). Let $t^i_{b_1} = t^i_{b_{11}}[x_1 := w_{b_{11}}]$. Given $(\lambda x_0.t^i_{a_{11}}) \simeq t^i_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t^i_{b_{11}} \longrightarrow^{\mathrm{x}i*} s^i_{b_{11}}$, $s^i_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i*} s^i_{b_{12}}$ and $(\lambda x_0.t^i_{a_{11}}) \simeq s^i_{b_{12}}$. We proceed by cases on $(\lambda x_0.t^i_{a_{11}}) \simeq s^i_{b_{12}}$.

Case a. (lam-sim). Let $s^i_{b12} = \lambda x_0.t^i_{b_{121}}$ where $t^i_{a_{11}} \simeq t^i_{b_{121}}$. We have $t^i_{b_1} \longrightarrow^{\mathrm{x}i*} \lambda x_0.t^i_{b_{121}}$ and $(\lambda x_0.t^i_{b_{121}})[x := w_{b_1}] \longrightarrow^{\mathrm{x}i} \lambda x_2.t^i_{b_{121}}[x_0 := x_2][x := w_{b_1}]$ where $x_2 \notin FV(\lambda x_0.t^i_{b_{121}}) \cup FV(w_{b_1}) \cup \{x\}$.
Let $x_3 \notin FV(\lambda x_0.t^i_{a_{11}}) \cup FV(w_{a_1}) \cup FV(\lambda x_0.t^i_{b_{121}}) \cup FV(w_{b_1}) \cup \{x\}$, then by the definition of α-equivalence, $\lambda x_1.t^i_{a_{11}}[x_1/x_0][w_{a_1}/x] \sim_\alpha \lambda x_3.t^i_{a_{11}}[x_3/x_0][w_{a_1}/x]$ and $\lambda x_2.t^i_{b_{121}}[x_0 := x_2][x := w_{b_1}] \sim_\alpha \lambda x_3.t^i_{b_{121}}[x_0 := x_3][x := w_{b_1}]$. By (lam-sim) and (subst-sim), we get $\lambda x_3.t^i_{a_{11}}[x_3/x_0][w_{a_1}/x] \simeq \lambda x_3.t^i_{b_{121}}[x_0 := x_3][x := w_{b_1}]$. Hence we have $\lambda x_1.t^i_{a_{11}}[x_1/x_0][w_{a_1}/x] \simeq \lambda x_2.t^i_{b_{121}}[x_0 := x_2][x := w_{b_1}]$.

Case b. (lamu-sim). Let $t^i_{a_1} = \lambda x_0.t^0_{a_{11}}$ and $s^i_{b_{12}} = \underline{\lambda} x_0.t^0_{b_{121}}$ where $t^0_{a_{11}} \simeq t^0_{b_{121}}$. We have $t^i_{b_1} \longrightarrow^{\mathrm{x}i*} \underline{\lambda} x_0.t^0_{b_{121}}$ and $(\underline{\lambda} x_0.t^0_{b_{121}})[x := w_{b_1}] \longrightarrow^{\mathrm{x}i} \underline{\lambda} x_2.t^0_{b_{121}}[x_0 := x_2][x := w_{b_1}]$ where $x_2 \notin FV(\underline{\lambda} x_0.t^0_{b_{121}}) \cup FV(w_{b_1}) \cup \{x\}$.
Let $x_3 \notin FV(\lambda x_0.t^0_{a_{11}}) \cup FV(w_{a_1}) \cup FV(\underline{\lambda} x_0.t^0_{b_{121}}) \cup FV(w_{b_1}) \cup \{x\}$, then by the definition of α-equivalence, $\lambda x_1.t^0_{a_{11}}[x_1/x_0][w_{a_1}/x] \sim_\alpha \lambda x_3.t^0_{a_{11}}[x_3/x_0][w_{a_1}/x]$ and $\underline{\lambda} x_2.t^0_{b_{121}}[x_0 := x_2][x := w_{b_1}] \sim_\alpha \underline{\lambda} x_3.t^0_{b_{121}}[x_0 := x_3][x := w_{b_1}]$. By (lamu-sim) and (subst-sim), we get $\lambda x_3.t^0_{a_{11}}[x_3/x_0][w_{a_1}/x] \simeq \underline{\lambda} x_3.t^0_{b_{121}}[x_0 := x_3][x := w_{b_1}]$. Hence we have $\lambda x_1.t^0_{a_{11}}[x_1/x_0][w_{a_1}/x] \simeq \underline{\lambda} x_2.t^0_{b_{121}}[x_0 := x_2][x := w_{b_1}]$.

Case 5. $(t^i_{a_1} = \langle t^{i+1}_{a_{11}} \rangle)$. We have

$$\frac{\langle t^{i+1}_{a_{11}} \rangle \simeq t^i_{b_1} \quad w_{a_1} \simeq w_{b_1}}{\langle t^{i+1}_{a_{11}} \rangle [w_{a_1}/x] \simeq t^i_{b_1}[x := w_{b_1}]}\ .$$

Then, $\langle t^{i+1}_{a_{11}} \rangle [w_{a_1}/x] = \langle t^{i+1}_{a_{11}}[w_{a_1}/x] \rangle$. We proceed by cases on $\langle t^{i+1}_{a_{11}} \rangle \simeq t^i_{b1}$.

Case i. (code-sim). Let $t^i_{b_1} = \langle t^{i+1}_{b_{11}} \rangle$ where $t^{i+1}_{a_{11}} \simeq t^{i+1}_{b_{11}}$. We have $\langle t^{i+1}_{b_{11}} \rangle \longrightarrow^{\mathrm{x}i*} \langle t^{i+1}_{b_{11}} \rangle$, $\langle t^{i+1}_{b_{11}} \rangle [x := w_{b_1}] \longrightarrow^{\mathrm{x}i} \langle t^{i+1}_{b_{11}}[x := w_{b_1}] \rangle$ and $\langle t^{i+1}_{a_{11}}[w_{a_1}/x] \rangle \simeq \langle t^{i+1}_{b_{11}}[x := w_{b_1}] \rangle$.

Case ii. (subst-sim). Let $t^i_{b_1} = t^i_{b_{11}}[x_1 := w_{b_{11}}]$. Given $\langle t^{i+1}_{a_{11}} \rangle \simeq t^i_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t^i_{b_{11}} \longrightarrow^{\mathrm{x}i*} s^i_{b_{11}}$, $s^i_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i*} s^i_{b_{12}}$ and $\langle t^{i+1}_{a_{11}} \rangle \simeq s^i_{b_{12}}$. We proceed by cases on $\langle t^{i+1}_{a_{11}} \rangle \simeq s^i_{b_{12}}$. The only case is (code-sim), so let $s^i_{b_{12}} = \langle t^{i+1}_{b_{121}} \rangle$ where $t^{i+1}_{a_{11}} \simeq t^{i+1}_{b_{121}}$. Then, $t^i_{b_1} \longrightarrow^{\mathrm{x}i*} \langle t^{i+1}_{b_{121}} \rangle$ and $\langle t^{i+1}_{b_{121}} \rangle [x := w_{b_1}] \longrightarrow^{\mathrm{x}i} \langle t^{i+1}_{b_{121}}[x := w_{b_1}] \rangle$. By (subst-sim) and (code-sim), we get $\langle t^{i+1}_{a_{11}}[w_{a_1}/x] \rangle \simeq \langle t^{i+1}_{b_{121}}[x := w_{b_1}] \rangle$.

Case 6. $(t^{i+1}_{a_1} = {\sim}t^{i}_{a_{11}})$. We have

$$\frac{{\sim}\, t^{i}_{a_{11}} \simeq t^{i+1}_{b_1} \quad w_{a_1} \simeq w_{b_1}}{({\sim}\, t^{i}_{a_{11}})[w_{a_1}/x] \simeq t^{i+1}_{b_1}[x := w_{b_1}].}$$

Then, $({\sim}t^{i}_{a_{11}})[w_{a_1}/x] = {\sim}(t^{i}_{a_{11}}[w_{a_1}/x])$. We proceed by cases on ${\sim}t^{i}_{a_{11}} \simeq t^{i+1}_{b_1}$.

Case i. (splice-sim). Let $t^{i+1}_{b_1} = {\sim}t^{i}_{b_{11}}$ where $t^{i}_{a_{11}} \simeq t^{i}_{b_{11}}$. We have ${\sim}t^{i}_{b_{11}} \longrightarrow^{\mathrm{x}(i+1)*} {\sim}t^{i}_{b_{11}}$ and $({\sim}t^{i}_{b_{11}})[x := w_{b_1}] \longrightarrow^{\mathrm{x}(i+1)} {\sim}(t^{i}_{b_{11}}[x := w_{b_1}])$ and ${\sim}(t^{i}_{a_{11}}[w_{a_1}/x]) \simeq {\sim}(t^{i}_{b_{11}}[x := w_{b_1}])$.

Case ii. (subst-sim). Let $t^{i+1}_{b_1} = t^{i+1}_{b_{11}}[x_1 := w_{b_{11}}]$. Given ${\sim}t^{i}_{a_{11}} \simeq t^{i+1}_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t^{i+1}_{b_{11}} \longrightarrow^{\mathrm{x}(i+1)*} s^{i+1}_{b_{11}}$, $s^{i+1}_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}(i+1)*} s^{i+1}_{b_{12}}$ and ${\sim}t^{i}_{a_{11}} \simeq s^{i+1}_{b_{12}}$. We proceed by cases on ${\sim}t^{i}_{a_{11}} \simeq s^{i+1}_{b_{12}}$. The only case is (splice-sim), so let $s^{i+1}_{b_{12}} = {\sim}t^{i}_{b_{121}}$ where $t^{i}_{a_{11}} \simeq t^{i}_{b_{121}}$. Then $t^{i+1}_{b_1} \longrightarrow^{\mathrm{x}(i+1)*} {\sim}\, t^{i}_{b_{121}}$ and $({\sim}t^{i}_{b_{121}})[x := w_{b_1}] \longrightarrow^{\mathrm{x}(i+1)} {\sim}(t^{i}_{b_{121}}[x := w_{b_1}])$. By (subst-sim) and (splice-sim), we get ${\sim}(t^{i}_{a_{11}}[w_{a_1}/x]) \simeq {\sim}(t^{i}_{b_{121}}[x := w_{b_1}])$.

Case 7. $(t^{i}_{a_1} = !t^{i}_{a_{11}})$. We have

$$\frac{!t^{i}_{a_{11}} \simeq t^{i}_{b_1} \quad w_{a_1} \simeq w_{b_1}}{(!t^{i}_{a_{11}})[w_{a_1}/x] \simeq t^{i}_{b_1}[x := w_{b_1}].}$$

Then, $(!t^{i}_{a_{11}})[w_{a_1}/x] = !(t^{i}_{a_{11}}[w_{a_1}/x])$. We proceed by cases on $!t^{i}_{a_{11}} \simeq t^{i}_{b_1}$.

Case i. (run-sim). Let $t^{i}_{b_1} = !t^{i}_{b_{11}}$ where $t^{i}_{a_{11}} \simeq t^{i}_{b_{11}}$. We have $!t^{i}_{b_{11}} \longrightarrow^{\mathrm{x}i*} !t^{i}_{b_{11}}$ and $(!t^{i}_{b_{11}})[x := w_{b_1}] \longrightarrow^{\mathrm{x}i} !(t^{i}_{b_{11}}[x := w_{b_1}])$ and $!(t^{i}_{a_{11}}[w_{a_1}/x]) \simeq !(t^{i}_{b_{11}}[x := w_{b_1}])$.

Case ii. (subst-sim). Let $t^{i}_{b_1} = t^{i}_{b_{11}}[x_1 := w_{b_{11}}]$. Given $!t^{i}_{a_{11}} \simeq t^{i}_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t^{i}_{b_{11}} \longrightarrow^{\mathrm{x}i*} s^{i}_{b_{11}}$, $s^{i}_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i*} s^{i}_{b_{12}}$ and $!t^{i}_{a_{11}} \simeq s^{i}_{b_{12}}$. We proceed by cases on $!t^{i}_{a_{11}} \simeq s^{i}_{b_{12}}$. The only case is (splice-sim), so let $s^{i}_{b_{12}} = !t^{i}_{b_{121}}$ where $t^{i}_{a_{11}} \simeq t^{i}_{b_{121}}$. Then $t^{i}_{b_1} \longrightarrow^{\mathrm{x}i*} !t^{i}_{b_{121}}$ and $(!t^{i}_{b_{121}})[x := w_{b_1}] \longrightarrow^{\mathrm{x}i} !(t^{i}_{b_{121}}[x := w_{b_1}])$. By (subst-sim) and (run-sim), we get $!(t^{i}_{a_{11}}[w_{a_1}/x]) \simeq !(t^{i}_{b_{121}}[x := w_{b_1}])$.

Case 8. $(t^{i}_{a_1} = n)$. We have

$$\frac{n \simeq t^{i}_{b_1} \quad w_{a_1} \simeq w_{b_1}}{n[w_{a_1}/x] \simeq t^{i}_{b_1}[x := w_{b_1}].}$$

Then, $n[w_{a_1}/x] = n$. We proceed by cases on $n \simeq t^{i}_{b_1}$.

Case i. (num-sim). Let $t^{i}_{b_1} = n$. We have $n \longrightarrow^{\mathrm{x}i*} n$, $n[x := w_{b_1}] \longrightarrow^{\mathrm{x}i} n$ and $n \simeq n$.

Case ii. (subst-sim). Then $t^i_{b_1} = t^i_{b_{11}}[x_1 := w_{b_{11}}]$. Given $n \simeq t^i_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t^i_{b_{11}} \longrightarrow^{\mathrm{x}i*} s^i_{b_{11}}$, $s^i_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i*} s^i_{b_{12}}$ and $n \simeq s^i_{b_{12}}$. We proceed by cases on $n \simeq s^i_{b_{12}}$. The only case is (num-sim), so let $s^i_{b_{12}} = n$. Then $t^i_{b_1} \longrightarrow^{\mathrm{x}i*} n$, $n[x := w_{b_1}] \longrightarrow^{\mathrm{x}i} n$ and $n \simeq n$.

Case 9. $(t^i_{a_1} = (t^i_{a_{11}} + t^i_{a_{12}}))$. We have

$$\frac{(t^i_{a_{11}} + t^i_{a_{12}}) \simeq t^i_{b_1} \quad w_{a_1} \simeq w_{b_1}}{(t^i_{a_{11}} + t^i_{a_{12}})[w_{a_1}/x] \simeq t^i_{b_1}[x := w_{b_1}].}$$

Then, $(t^i_{a_{11}} + t^i_{a_{12}})[w_{a_1}/x] = (t^i_{a_{11}}[w_{a_1}/x]) + (t^i_{a_{12}}[w_{a_1}/x])$. We proceed by cases on $(t^i_{a_{11}} + t^i_{a_{12}}) \simeq t^i_{b_1}$.

Case i. (plus-sim). Let $t^i_{b_1} = (t^i_{b_{11}} + t^i_{b_{12}})$ where $t^i_{a_{11}} \simeq t^i_{b_{11}}$ and $t^i_{a_{12}} \simeq t^i_{b_{12}}$. We have $(t^i_{b_{11}} + t^i_{b_{12}}) \longrightarrow^{\mathrm{x}i*} (t^i_{b_{11}} + t^i_{b_{12}})$ and $(t^i_{b_{11}} + t^i_{b_{12}})[x := w_{b1}] \longrightarrow^{\mathrm{x}i} (t^i_{b_{11}}[x := w_{b_1}]) + (t^i_{b_{12}}[x := w_{b_1}])$. By (subst-sim) and (plus-sim), we get $(t^i_{a_{11}}[w_{a_1}/x]) + (t^i_{a_{12}}[w_{a_1}/x]) \simeq (t^i_{b_{11}}[x := w_{b_1}]) + (t^i_{b_{12}}[x := w_{b_1}])$.

Case ii. (subst-sim). Let $t^i_{b_1} = t^i_{b_{11}}[x_1 := w_{b_{11}}]$. Given $(t^i_{a_{11}} + t^i_{a_{12}}) \simeq t^i_{b_{11}}[x_1 := w_{b_{11}}]$, by the induction hypothesis, we have $t^i_{b_{11}} \longrightarrow^{\mathrm{x}i*} s^i_{b_{11}}$, $s^i_{b_{11}}[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i*} s^i_{b_{12}}$, and $(t^i_{a_{11}} + t^i_{a_{12}}) \simeq s^i_{b_{12}}$. We proceed by cases on $(t^i_{a_{11}} + t^i_{a_{12}}) \simeq s^i_{b_{12}}$. The only case is (plus-sim), so let $s^i_{b_{12}} = (t^i_{b_{121}} + t^i_{b_{122}})$ where $t^i_{a_{11}} \simeq t^i_{b_{121}}$ and $t^i_{a_{12}} \simeq t^i_{b_{122}}$. Then, $t^i_{b_1} \longrightarrow^{\mathrm{x}i*} (t^i_{b_{121}} + t^i_{b_{122}})$ and $(t^i_{b_{121}} + t^i_{b_{122}})[x := w_{b_1}] \longrightarrow^{\mathrm{x}i} (t^i_{b_{121}}[x := w_{b_1}]) + (t^i_{b_{122}}[x := w_{b_1}])$. By (subst-sim) and (plus-sim), we get $(t^i_{a_{11}}[w_{a_1}/x]) + (t^i_{a_{12}}[w_{a_1}/x]) \simeq (t^i_{b_{121}}[x := w_{b_1}]) + (t^i_{b_{122}}[x := w_{b_1}])$.

□

D.1.4 Canonisation

Lemma 316 (Canonisation of (Substitutional) MetaML)**.** *If* $t^i_{a_1} \simeq v^i_{b_1}$*, then* $t^i_{a_1} \in \mathrm{VALUE}^i_{\mathrm{sub}}$.

Proof. We proceed by structural induction on $t^i_{a_1} \simeq v^i_{b_1}$.

Case 1. (var-sim). Let $t^{i+1}_{a_1} = v^{i+1}_{b_1} = x$. We have $x \in \mathrm{VALUE}^{i+1}_{\mathrm{sub}}$.

Case 2. (app-sim). Let $t^{i+1}_{a_1} = t^{i+1}_{a_{11}}\ t^{i+1}_{a_{12}}$ and $v^{i+1}_{b_1} = v^{i+1}_{b_{11}}\ v^{i+1}_{b_{12}}$ where $t^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{11}}$ and $t^{i+1}_{a_{12}} \simeq v^{i+1}_{b_{12}}$. By the induction hypothesis, $t^{i+1}_{a_{11}} \in \mathrm{VALUE}^{i+1}_{\mathrm{sub}}$ and $t^{i+1}_{a_{12}} \in \mathrm{VALUE}^{i+1}_{\mathrm{sub}}$. Then $t^{i+1}_{a_{11}}\ t^{i+1}_{a_{12}} \in \mathrm{VALUE}^{i+1}_{\mathrm{sub}}$.

Case 3. (lam-sim). We proceed by cases on i.

Case i. (0). Let $t^0_{a_1} = \lambda x.t^0_{a_{11}}$ and $v^0_{b_1} = \lambda x.t^0_{b_{11}}$ where $t^0_{a_{11}} \simeq t^0_{b_{11}}$. This case is vacuous because $\lambda x.t^0_{b_{11}} \notin \mathrm{VALUE}^0_{\mathrm{exp}}$.

Case ii. $(i+1)$. Let $t^{i+1}_{a_1} = \lambda x.t^{i+1}_{a_{11}}$ and $v^{i+1}_{b_1} = \lambda x.v^{i+1}_{b_{11}}$ where $t^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{11}}$. By the induction hypothesis, $t^{i+1}_{a_{11}} \in \mathrm{VALUE}^{i+1}_{\mathrm{sub}}$. Then $\lambda x.t^{i+1}_{a_{11}} \in \mathrm{VALUE}^{i+1}_{\mathrm{sub}}$.

Case 4. (lamu-sim). Let $t^i_{a_1} = \lambda x.t^0_{a_{11}}$ and $v^i_{b_1} = \underline{\lambda} x.t^0_{b_{11}}$ where $t^0_{a_{11}} \simeq t^0_{b_{11}}$. Then $\lambda x.t^0_{a_{11}} \in \text{VALUE}^0_{\text{sub}} \subseteq \text{VALUE}^i_{\text{sub}}$.

Case 5. (code-sim). Let $t^i_{a_1} = \langle t^{i+1}_{a_{11}} \rangle$ and $v^i_{b_1} = \langle v^{i+1}_{b_{11}} \rangle$ where $t^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{11}}$. By the induction hypothesis, $t^{i+1}_{a_{11}} \in \text{VALUE}^{i+1}_{\text{sub}}$. Then $\langle t^{i+1}_{a_{11}} \rangle \in \text{VALUE}^i_{\text{sub}}$.

Case 6. (splice-sim). Let $t^{i+2}_{a_1} = {\sim} t^{i+1}_{a_{11}}$ and $v^{i+2}_{b_1} = {\sim} v^{i+1}_{b_{11}}$ where $t^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{11}}$. By the induction hypothesis, $t^{i+1}_{a_{11}} \in \text{VALUE}^{i+1}_{\text{sub}}$. Then ${\sim} t^{i+1}_{a_{11}} \in \text{VALUE}^{i+2}_{\text{sub}}$.

Case 7. (run-sim). Let $t^{i+1}_{a_1} = !t^{i+1}_{a_{11}}$ and $v^{i+1}_{b_1} = !v^{i+1}_{b_{11}}$ where $t^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{11}}$. By the induction hypothesis, $t^{i+1}_{a_{11}} \in \text{VALUE}^{i+1}_{\text{sub}}$. Then $!t^{i+1}_{a_{11}} \in \text{VALUE}^{i+1}_{\text{sub}}$.

Case 8. (num-sim). Let $t^i_{a_1} = v^i_{b_1} = n$. We have $n \in \text{VALUE}^i_{\text{sub}}$.

Case 9. (plus-sim). Let $t^{i+1}_{a_1} = t^{i+1}_{a_{11}} + t^{i+1}_{a_{12}}$ and $v^{i+1}_{b_1} = v^{i+1}_{b_{11}} + v^{i+1}_{b_{12}}$ where $t^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{11}}$ and $t^{i+1}_{a_{12}} \simeq v^{i+1}_{b_{12}}$. By the induction hypothesis, $t^{i+1}_{a_{11}} \in \text{VALUE}^{i+1}_{\text{sub}}$ and $t^{i+1}_{a_{12}} \in \text{VALUE}^{i+1}_{\text{sub}}$. Then $t^{i+1}_{a_{11}} + t^{i+1}_{a_{12}} \in \text{VALUE}^{i+1}_{\text{sub}}$.

Case 10. (subst-sim). This case is vacuous.

□

Lemma 317 (Canonisation of Explicit MetaML)**.** *If* $v^i_{a_1} \simeq t^i_{b_1}$*, then* $t^i_{b_1} \longrightarrow^{i*} v^i_{b_2}$ *and* $v^i_{a_1} \simeq v^i_{b_2}$.

Proof. We proceed by structural induction on $v^i_{a_1} \simeq t^i_{b_1}$.

Case 1. (var-sim). Let $v^{i+1}_{a_1} = t^{i+1}_{b_1} = x$. Then $x \longrightarrow^{(i+1)*} x$, $x \in \text{VALUE}^{i+1}_{\text{exp}}$ and $x \simeq x$.

Case 2. (app-sim). Let $v^{i+1}_{a_1} = v^{i+1}_{a_{11}}\ v^{i+1}_{a_{12}}$ and $t^{i+1}_{b_1} = t^{i+1}_{b_{11}}\ t^{i+1}_{b_{12}}$ where $v^{i+1}_{a_{11}} \simeq t^{i+1}_{b_{11}}$ and $v^{i+1}_{a_{12}} \simeq t^{i+1}_{b_{12}}$. By the induction hypothesis, we have $t^{i+1}_{b_{11}} \longrightarrow^{(i+1)*} v^{i+1}_{b_{21}}$, $v^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{21}}$, $t^{i+1}_{b_{12}} \longrightarrow^{(i+1)*} v^{i+1}_{b_{22}}$ and $v^{i+1}_{a_{12}} \simeq v^{i+1}_{b_{22}}$. Then $t^{i+1}_{b_{11}}\ t^{i+1}_{b_{12}} \longrightarrow^{(i+1)*} v^{i+1}_{b_{21}}\ t^{i+1}_{b_{12}} \longrightarrow^{(i+1)*} v^{i+1}_{b_{21}}\ v^{i+1}_{b_{22}}$ and $v^{i+1}_{b_{21}}\ v^{i+1}_{b_{22}} \in \text{VALUE}^{i+1}_{\text{exp}}$. By (app-sim), $v^{i+1}_{a_{11}}\ v^{i+1}_{a_{12}} \simeq v^{i+1}_{b_{21}}\ v^{i+1}_{b_{22}}$.

Case 3. (lam-sim). We proceed by cases on i.

- *Case* i. (0). Let $v^0_{a_1} = \lambda x.t^0_{a_{11}}$ and $t^0_{b_1} = \lambda x.t^0_{b_{11}}$ where $t^0_{a_{11}} \simeq t^0_{b_{11}}$. Then $\lambda x.t^0_{b_{11}} \longrightarrow^0 \underline{\lambda} x.t^0_{b_{11}}$ and $\underline{\lambda} x.t^0_{b_{11}} \in \text{VALUE}^0_{\text{exp}}$. By (lamu-sim), $\lambda x.t^0_{a_{11}} \simeq \underline{\lambda} x.t^0_{b_{11}}$.
- *Case* ii. $(i+1)$. Let $v^{i+1}_{a_1} = \lambda x.v^{i+1}_{a_{11}}$ and $t^{i+1}_{b_1} = \lambda x.t^{i+1}_{b_{11}}$ where $v^{i+1}_{a_{11}} \simeq t^{i+1}_{b_{11}}$. By the induction hypothesis, we have $t^{i+1}_{b_{11}} \longrightarrow^* v^{i+1}_{b_{21}}$ and $v^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{21}}$. Then $\lambda x.t^{i+1}_{b_{11}} \longrightarrow^{(i+1)*} \lambda x.v^{i+1}_{b_{21}}$ and $\lambda x.v^{i+1}_{b_{21}} \in \text{VALUE}^{i+1}_{\text{exp}}$. By (lam-sim), $\lambda x.v^{i+1}_{a_{11}} \simeq \lambda x.v^{i+1}_{b_{21}}$.

Case 4. (lamu-sim). Let $v^i_{a_1} = \lambda x.t^0_{a_{11}}$ and $t^i_{b_1} = \underline{\lambda} x.t^0_{b_{11}}$ where $t^0_{a_{11}} \simeq t^0_{b_{11}}$. Then $\underline{\lambda} x.t^0_{b_{11}} \longrightarrow^{i*} \underline{\lambda} x.t^0_{b_{11}}$, $\underline{\lambda} x.t^0_{b_{11}} \in \text{VALUE}^i_{\text{exp}}$ and $\lambda x.t^0_{a_{11}} \simeq \underline{\lambda} x.t^0_{b_{11}}$.

Case 5. (code-sim). Let $v^i_{a_1} = \langle v^{i+1}_{a_{11}} \rangle$ and $t^i_{b_1} = \langle t^{i+1}_{b_{11}} \rangle$ where $v^{i+1}_{a_{11}} \simeq t^{i+1}_{b_{11}}$. By the induction hypothesis, we have $t^{i+1}_{b_{11}} \longrightarrow^{(i+1)*} v^{i+1}_{b_{21}}$ and $v^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{21}}$. Then $\langle t^{i+1}_{b_{11}} \rangle \longrightarrow^{i*} \langle v^{i+1}_{b_{21}} \rangle$ and $\langle v^{i+1}_{b_{21}} \rangle \in \text{VALUE}^i_{\text{exp}}$. By (code-sim), $\langle v^{i+1}_{a_{11}} \rangle \simeq \langle v^{i+1}_{b_{11}} \rangle$.

Case 6. (splice-sim). Let $v_{a_1}^{i+2} = {\sim}v_{a_{11}}^{i+1}$ and $t_{b_1}^{i+2} = {\sim}t_{b_{11}}^{i+1}$ where $v_{a_{11}}^{i+1} \simeq t_{b_{11}}^{i+1}$. By the induction hypothesis, we have $t_{b_{11}}^{i+1} \longrightarrow^{(i+1)*} v_{b_{21}}^{i+1}$ and $v_{a_{11}}^{i+1} \simeq v_{b_{21}}^{i+1}$. Then ${\sim}t_{b_{11}}^{i+1} \longrightarrow^{(i+2)*} {\sim}v_{b_{21}}^{i+1}$ and ${\sim}v_{b_{21}}^{i+1} \in \text{VALUE}_{\text{exp}}^{i+2}$. By (splice-sim), we get ${\sim}v_{a_{11}}^{i+1} \simeq {\sim}v_{b_{21}}^{i+1}$.

Case 7. (run-sim). Let $v_{a_1}^{i+1} = !v_{a_{11}}^{i+1}$ and $t_{b_1}^{i+1} = !t_{b_{11}}^{i+1}$ where $v_{a_{11}}^{i+1} \simeq t_{b_{11}}^{i+1}$. By the induction hypothesis, we have $t_{b_{11}}^{i+1} \longrightarrow^{(i+1)*} v_{b_{21}}^{i+1}$ and $v_{a_{11}}^{i+1} \simeq v_{b_{21}}^{i+1}$. Then $!t_{b_{11}}^{i+1} \longrightarrow^{(i+1)*} !v_{b_{21}}^{i+1}$ and $!v_{b_{21}}^{i+1} \in \text{VALUE}_{\text{exp}}^{i+1}$. By (run-sim), we get $!v_{a_{11}}^{i+1} \simeq !v_{b_{21}}^{i+1}$.

Case 8. (num-sim). Let $v_{a_1}^{i} = t_{b_1}^{i} = n$. Then $n \longrightarrow^{i*} n$, $n \in \text{VALUE}_{\text{exp}}^{i}$ and $n \simeq n$.

Case 9. (plus-sim). Let $v_{a_1}^{i+1} = v_{a_{11}}^{i+1} + v_{a_{12}}^{i+1}$ and $t_{b_1}^{i+1} = t_{b_{11}}^{i+1} + t_{b_{12}}^{i+1}$ where $v_{a_{11}}^{i+1} \simeq t_{b_{11}}^{i+1}$ and $v_{a_{12}}^{i+1} \simeq t_{b_{12}}^{i+1}$. By the induction hypothesis, we have $t_{b_{11}}^{i+1} \longrightarrow^{(i+1)*} v_{b_{21}}^{i+1}$, $v_{a_{11}}^{i+1} \simeq v_{b_{21}}^{i+1}$, $t_{b_{12}}^{i+1} \longrightarrow^{(i+1)*} v_{b_{22}}^{i+1}$ and $v_{a_{12}}^{i+1} \simeq v_{b_{22}}^{i+1}$. Then $t_{b_{11}}^{i+1} + t_{b_{12}}^{i+1} \longrightarrow^{(i+1)*} v_{b_{21}}^{i+1} + t_{b_{12}}^{i+1} \longrightarrow^{(i+1)*} v_{b_{21}}^{i+1} + v_{b_{22}}^{i+1}$ and $v_{b_{21}}^{i+1} + v_{b_{22}}^{i+1} \in \text{VALUE}_{\text{exp}}^{i+1}$. By (plus-sim), $v_{a_{11}}^{i+1} + v_{a_{12}}^{i+1} \simeq v_{b_{21}}^{i+1} + v_{b_{22}}^{i+1}$.

Case 10. (subst-sim). Let $v_{a_1}^{i} = t_{a_{11}}^{i}[w_{a_1}/x]$ and $t_{b_1}^{i} = t_{b_{11}}^{i}[x := w_{b_1}]$ where $t_{a_{11}}^{i} \simeq t_{b_{11}}^{i}$ and $w_{a_1} \simeq w_{b_1}$. By Lemma 315, $t_{b_{11}}^{i}[x := w_{b_1}] \longrightarrow^{\text{xi}*} s_{b_{12}}^{i}$ and $t_{a_{11}}^{i}[w_{a_1}/x] \simeq s_{b_{12}}^{i}$. We then proceed by structural induction on $t_{a_{11}}^{i}[w_{a_1}/x] \simeq s_{b_{12}}^{i}$.

Case i. (var-sim). Let $t_{a_{11}}^{i+1}[w_{a_1}/x] = s_{b_{12}}^{i+1} = x_0$. Then $x_0 \longrightarrow^{(i+1)*} x_0$, $x_0 \in \text{VALUE}_{\text{exp}}^{i+1}$ and $x_0 \simeq x_0$.

Case ii. (app-sim). Let $t_{a_{11}}^{i+1}[w_{a_1}/x] = v_{a_{111}}^{i+1}\ v_{a_{112}}^{i+1}$ and $s_{b_{12}}^{i+1} = t_{b_{121}}^{i+1}\ t_{b_{122}}^{i+1}$ where $v_{a_{111}}^{i+1} \simeq t_{b_{121}}^{i+1}$ and $v_{a_{112}}^{i+1} \simeq t_{b_{122}}^{i+1}$. By the induction hypothesis, we get $t_{b_{121}}^{i+1} \longrightarrow^{(i+1)*} v_{b_{131}}^{i+1}$, $v_{a_{111}}^{i+1} \simeq v_{b_{131}}^{i+1}$, $t_{b_{122}}^{i+1} \longrightarrow^{(i+1)*} v_{b_{132}}^{i+1}$ and $v_{a_{112}}^{i+1} \simeq v_{b_{132}}^{i+1}$. Then $t_{b_{11}}^{i+1}[x := w_{b_1}] \longrightarrow^{(i+1)*} t_{b_{121}}^{i+1}\ t_{b_{122}}^{i+1} \longrightarrow^{(i+1)*} v_{b_{131}}^{i+1}\ t_{b_{122}}^{i+1} \longrightarrow^{(i+1)*} v_{b_{131}}^{i+1}\ v_{b_{132}}^{i+1}$ and $v_{b_{131}}^{i+1}\ v_{b_{132}}^{i+1} \in \text{VALUE}_{\text{exp}}^{i+1}$. By (app-sim), $v_{a_{111}}^{i+1}\ v_{a_{112}}^{i+1} \simeq v_{a_{131}}^{i+1}\ v_{a_{132}}^{i+1}$.

Case iii. (lam-sim). We proceed by cases on i.

Case a. (0). Let $t_{a_{11}}^{0}[w_{a_1}/x] = \lambda x_0.t_{a_{111}}^{0}$ and $s_{b_{12}}^{0} = \lambda x_0.t_{b_{121}}^{0}$ where $t_{a_{111}}^{0} \simeq t_{b_{121}}^{0}$. Then $t_{b_{11}}^{0}[x := w_{b_1}] \longrightarrow^{0*} \lambda x_0.t_{b_{121}}^{0} \longrightarrow^{0} \underline{\lambda} x_0.t_{b_{121}}^{0}$ and $\underline{\lambda} x_0.t_{b_{121}}^{0} \in \text{VALUE}_{\text{exp}}^{0}$. By (lamu-sim), $\lambda x_0.t_{a_{111}}^{0} \simeq \underline{\lambda} x_0.t_{b_{121}}^{0}$.

Case b. $(i+1)$. Let $t_{a_{11}}^{i+1}[w_{a_1}/x] = \lambda x_0.v_{a_{111}}^{i+1}$ and $s_{b_{12}}^{i+1} = \lambda x_0.t_{b_{121}}^{i+1}$ where $v_{a_{111}}^{i+1} \simeq t_{b_{121}}^{i+1}$. By the induction hypothesis, we get $t_{b_{121}}^{i+1} \longrightarrow^{(i+1)*} v_{b_{131}}^{i+1}$ and $v_{a_{111}}^{i+1} \simeq v_{b_{131}}^{i+1}$. Then $t_{b_{11}}^{i+1}[x := w_{b_1}] \longrightarrow^{(i+1)*} \lambda x_0.t_{b_{121}}^{i+1} \longrightarrow^{(i+1)*} \lambda x_0.v_{b_{131}}^{i+1}$ and $\lambda x_0.v_{b_{131}}^{i+1} \in \text{VALUE}_{\text{exp}}^{i+1}$. By (lam-sim), $\lambda x_0.v_{a_{111}}^{i+1} \simeq \lambda x_0.v_{b_{131}}^{i+1}$.

Case iv. (lamu-sim). Let $t_{a_{11}}^{i}[w_{a_1}/x] = \lambda x_0.t_{a_{111}}^{0}$ and $s_{b_{12}}^{i} = \underline{\lambda} x_0.t_{b_{121}}^{0}$ where $t_{a_{111}}^{0} \simeq t_{b_{121}}^{0}$. Then $t_{b_{11}}^{i}[x := w_{b_1}] \longrightarrow^{i*} \underline{\lambda} x_0.t_{b_{121}}^{0}$, $\underline{\lambda} x_0.t_{b_{121}}^{0} \in \text{VALUE}_{\text{exp}}^{i}$ and $\lambda x_0.t_{a_{111}}^{0} \simeq \underline{\lambda} x_0.t_{b_{121}}^{0}$.

Case v. (code-sim). Let $t_{a_{11}}^{i}[w_{a_1}/x] = \langle v_{a_{111}}^{i+1} \rangle$ and $s_{b_{12}}^{i} = \langle t_{b_{121}}^{i+1} \rangle$ where $v_{a_{111}}^{i+1} \simeq t_{b_{121}}^{i+1}$. By the induction hypothesis, we get $t_{b_{121}}^{i+1} \longrightarrow^{(i+1)*} v_{b_{131}}^{i+1}$ and $v_{a_{111}}^{i+1} \simeq v_{b_{131}}^{i+1}$. Then $t_{b_{11}}^{i}[x := w_{b_1}] \longrightarrow^{i*} \langle t_{b_{121}}^{i+1} \rangle \longrightarrow^{i*} \langle v_{b_{131}}^{i+1} \rangle$ and $\langle v_{b_{131}}^{i+1} \rangle \in \text{VALUE}_{\text{exp}}^{i}$. By (code-sim), $\langle v_{a_{111}}^{i+1} \rangle \simeq \langle v_{b_{131}}^{i+1} \rangle$.

Case vi. (splice-sim). Let $t^{i+2}_{a_{11}}[w_{a_1}/x] = {\sim}v^{i+1}_{a_{111}}$ and $s^{i+2}_{b_{12}} = {\sim}t^{i+1}_{b_{121}}$ where $v^{i+1}_{a_{111}} \simeq t^{i+1}_{b_{121}}$. By the induction hypothesis, we get $t^{i+1}_{b_{121}} \longrightarrow^{(i+1)*} v^{i+1}_{b_{131}}$ and $v^{i+1}_{a_{111}} \simeq v^{i+1}_{b_{131}}$. Then $t^{i+2}_{b_{11}}[x := w_{b_1}] \longrightarrow^{(i+2)*} {\sim}t^{i+1}_{b_{121}} \longrightarrow^{(i+2)*} {\sim}v^{i+1}_{b_{131}}$ and ${\sim}v^{i+1}_{b_{131}} \in \text{VALUE}^{i+2}_{\text{exp}}$. By (splice-sim), ${\sim}v^{i+1}_{a_{111}} \simeq {\sim}v^{i+1}_{b_{131}}$.

Case vii. (run-sim). Let $t^{i+1}_{a_{11}}[w_{a_1}/x] = !v^{i+1}_{a_{111}}$ and $s^{i+1}_{b_{12}} = !t^{i+1}_{b_{121}}$ where $v^{i+1}_{a_{111}} \simeq t^{i+1}_{b_{121}}$. By the induction hypothesis, we get $t^{i+1}_{b_{121}} \longrightarrow^{(i+1)*} v^{i+1}_{b_{131}}$ and $v^{i+1}_{a_{111}} \simeq v^{i+1}_{b_{131}}$. Then $t^{i+1}_{b_{11}}[x := w_{b_1}] \longrightarrow^{(i+1)*} !t^{i+1}_{b_{121}} \longrightarrow^{(i+1)*} !v^{i+1}_{b_{131}}$ and $!v^{i+1}_{b_{131}} \in \text{VALUE}^{i+1}_{\text{exp}}$. By (run-sim), $!v^{i+1}_{a_{111}} \simeq !v^{i+1}_{b_{131}}$.

Case viii. (num-sim). Let $t^{i}_{a_{11}}[w_{a_1}/x] = s^{i}_{b_{12}} = n$. Then $n \longrightarrow^{i*} n$, $n \in \text{VALUE}^{i}_{\text{exp}}$ and $n \simeq n$.

Case ix. (plus-sim). Let $t^{i+1}_{a_{11}}[w_{a_1}/x] = v^{i+1}_{a_{111}} + v^{i+1}_{a_{112}}$ and $s^{i+1}_{b_{12}} = t^{i+1}_{b_{121}} + t^{i+1}_{b_{122}}$ where $v^{i+1}_{a_{111}} \simeq t^{i+1}_{b_{121}}$ and $v^{i+1}_{a_{112}} \simeq t^{i+1}_{b_{122}}$. By the induction hypothesis, we get $t^{i+1}_{b_{121}} \longrightarrow^{(i+1)*} v^{i+1}_{b_{131}}$, $v^{i+1}_{a_{111}} \simeq v^{i+1}_{b_{131}}$, $t^{i+1}_{b_{122}} \longrightarrow^{(i+1)*} v^{i+1}_{b_{132}}$ and $v^{i+1}_{a_{112}} \simeq v^{i+1}_{b_{132}}$. Then $t^{i+1}_{b_{11}}[x := w_{b_1}] \longrightarrow^{(i+1)*} t^{i+1}_{b_{121}} + t^{i+1}_{b_{122}} \longrightarrow^{(i+1)*} v^{i+1}_{b_{131}} + t^{i+1}_{b_{122}} \longrightarrow^{(i+1)*} v^{i+1}_{b_{131}} + v^{i+1}_{b_{132}}$ and $v^{i+1}_{b_{131}} + v^{i+1}_{b_{132}} \in \text{VALUE}^{i+1}_{\text{exp}}$. By (plus-sim), $v^{i+1}_{a_{111}} + v^{i+1}_{a_{112}} \simeq v^{i+1}_{a_{131}} + v^{i+1}_{a_{132}}$.

Case x. (subst-sim). This case is vacuous.

□

D.1.5 Explicit Substitution Descendant Relation

Definition 318 (Explicit Substitution Descendant Relation)**.** For any $t_1, t_2 \in \text{RTERM}_{\text{exp}}$, $t_1 \prec^{\text{x}} t_2$ if and only if $t_2 \longrightarrow^{\text{x}} t_1$. We call $\prec^{\text{x}}$ the explicit substitution descendant relation.

Definition 319 (Weight Function)**.** Define the weight function W to be a total function from the set of Explicit MetaML runtime terms to the set of natural numbers.

$$W : \text{RTERM}_{\text{exp}} \longrightarrow \mathbb{N}$$

$$\begin{aligned}
W(x) &= 1 \\
W(t_1\ t_2) &= W(t_1) + W(t_2) + 1 \\
W(\underline{\lambda} x.t) &= 1 \\
W(\lambda x.t) &= 1 \\
W(\langle t \rangle) &= W(t) + 1 \\
W({\sim}t) &= W(t) + 1 \\
W(!t) &= W(t) + 1 \\
W(n) &= 1 \\
W(t_1 + t_2) &= W(t_1) + W(t_2) + 1 \\
W(t[x := w]) &= W(t) \cdot (W(w) + 1)
\end{aligned}$$

Remark 320. Observe that $W(t) > 0$ for any $t \in \text{RTERM}_{\text{exp}}$.

Lemma 321. *For any $t_1, t_2 \in \text{RTERM}_{\text{exp}}$, if $t_1 \longrightarrow^{\text{x}} t_2$, then $W(t_2) < W(t_1)$.*

Proof. We proceed by structural induction on $t_1 \longrightarrow^{\mathrm{x}} t_2$.

Case 1. (var-eq-subst). We have $t_1 = x[x := w]$ and $t_2 = w$. Then, $W(x[x := w]) = W(x) \cdot (W(w) + 1) = W(w) + 1$, and $W(w) < W(w) + 1$.

Case 2. (var-dif-subst). We have $t_1 = x_1[x_2 := w]$ and $t_2 = x_1$ where $x_1 \not\equiv x_2$. Then, $W(x_1) = 1$, $W(x_1[x_2 := w]) = W(x_1) \cdot (W(w) + 1) = W(w) + 1$, and $1 < W(w) + 1$.

Case 3. (num-subst). We have $t_1 = n[x := w]$ and $t_2 = n$. Then, $W(n) = 1$, $W(n[x := w]) = W(n) \cdot (W(w) + 1) = W(w) + 1$, and $1 < W(w) + 1$.

Case 4. (app-subst). We have $t_1 = (t_1\ t_2)[x := w]$ and $t_2 = (t_1[x := w])\ (t_2[x := w])$. Then, $W((t_1[x := w])\ (t_2[x := w])) = W(t_1[x := w]) + W(t_2[x := w]) + 1 = W(t_1) \cdot (W(w) + 1) + W(t_2) \cdot (W(w) + 1) + 1 = (W(t_1) + W(t_2)) \cdot (W(w) + 1) + 1$, $W((t_1\ t_2)[x := w]) = W((t_1\ t_2)) \cdot ((W(w) + 1) = (W(t_1) + W(t_2) + 1) \cdot ((W(w) + 1) = (W(t_1) + W(t_2)) \cdot (W(w) + 1) + W(w) + 1$, and $(W(t_1) + W(t_2)) \cdot (W(w) + 1) + 1 < (W(t_1) + W(t_2)) \cdot (W(w) + 1) + W(w) + 1$.

Case 5. (plus-subst). We have $t_1 = (t_1 + t_2)[x := w]$ and $t_2 = (t_1[x := w]) + (t_2[x := w])$. Then, $W((t_1[x := w]) + (t_2[x := w])) = W(t_1[x := w]) + W(t_2[x := w]) + 1 = W(t_1) \cdot (W(w) + 1) + W(t_2) \cdot (W(w) + 1) + 1 = (W(t_1) + W(t_2)) \cdot (W(w) + 1) + 1$, $W((t_1 + t_2)[x := w]) = W((t_1 + t_2)) \cdot ((W(w) + 1) = (W(t_1) + W(t_2) + 1) \cdot ((W(w) + 1) = (W(t_1) + W(t_2)) \cdot (W(w) + 1) + W(w) + 1$, and $(W(t_1) + W(t_2)) \cdot (W(w) + 1) + 1 < (W(t_1) + W(t_2)) \cdot (W(w) + 1) + W(w) + 1$.

Case 6. (lam-eq-subst). We have $t_1 = (\lambda x.t_{11})[x := w]$ and $t_2 = \lambda x.t_{11}$. Then, $W(\lambda x.t_{11}) = 1$, $W((\lambda x.t_{11})[x := w]) = W(\lambda x.t_{11}) \cdot (W(w) + 1) = W(w) + 1$, and $1 < W(w) + 1$.

Case 7. (lamu-eq-subst). We have $t_1 = (\underline{\lambda} x.t^0_{11})[x := w]$ and $t_2 = \underline{\lambda} x.t^0_{11}$. Then, $W(\underline{\lambda} x.t^0_{11}) = 1$, $W((\underline{\lambda} x.t^0_{11})[x := w]) = W(\underline{\lambda} x.t^0_{11}) \cdot (W(w) + 1) = W(w) + 1$, and $1 < W(w) + 1$.

Case 8. (lam-df-subst). We have $t_1 = (\lambda x_1.t_{11})[x_2 := w]$ and $t_2 = \lambda x_N.t_{11}[x_1 := x_N][x_2 := w]$ where $x_1 \not\equiv x_2$ and $x_N \notin FV(\lambda x_1.t_{11}) \cup FV(w) \cup \{x_2\}$. Then, $W(\lambda x_N.t_{11}[x_1 := x_N][x_2 := w]) = 1$, $W((\lambda x_1.t_{11})[x_2 := w]) = W(\lambda x_1.t_{11}) \cdot (W(w) + 1) = W(w) + 1$, and $1 < W(w) + 1$.

Case 9. (lamu-df-subst). We have $t_1 = (\underline{\lambda} x_1.t_{11})[x_2 := w]$ and $t_2 = \underline{\lambda} x_N.t_{11}[x_1 := x_N][x_2 := w]$ where $x_1 \not\equiv x_2$ and $x_N \notin FV(\underline{\lambda} x_1.t_{11}) \cup FV(w) \cup \{x_2\}$. Then, $W(\underline{\lambda} x_N.t_{11}[x_1 := x_N][x_2 := w]) = 1$, $W((\underline{\lambda} x_1.t_{11})[x_2 := w]) = W(\underline{\lambda} x_1.t_{11}) \cdot (W(w) + 1) = W(w) + 1$, and $1 < W(w) + 1$.

Case 10. (code-subst). We have $t_1 = \langle t_{11} \rangle [x := w]$ and $t_2 = \langle t_{11}[x := w] \rangle$. Then, $W(\langle t_{11}[x := w] \rangle) = W(t_{11}[x := w]) + 1 = W(t_{11}) \cdot (W(w) + 1) + 1$, $W(\langle t_{11} \rangle [x := w]) = W(\langle t_{11} \rangle) \cdot (W(w) + 1) = (W(t_{11}) + 1) \cdot (W(w) + 1) = W(t_{11}) \cdot (W(w) + 1) + W(w) + 1$, and $W(t_{11}) \cdot (W(w) + 1) + 1 < W(t_{11}) \cdot (W(w) + 1) + W(w) + 1$.

Case 11. (run-subst). We have $t_1 = (!t_{11})[x := w]$ and $t_2 = !t_{11}[x := w]$. Then, $W(!t_{11}[x := w]) = W(t_{11}[x := w]) + 1 = W(t_{11}) \cdot (W(w) + 1) + 1$, $W((!t_{11})[x := w]) = W(!t_{11}) \cdot (W(w) + 1) = (W(t_{11}) + 1) \cdot$

$(W(w)+1) = W(t_{11}) \cdot (W(w)+1) + W(w) + 1$, and $W(t_{11}) \cdot (W(w)+1) + 1 < W(t_{11}) \cdot (W(w)+1) + W(w) + 1$.

Case 12. (splice-subst). We have $t_1 = (\sim t_{11})[x := w]$ and $t_2 = \sim t_{11}[x := w]$. Then, $W(\sim t_{11}[x := w]) = W(t_{11}[x := w]) + 1 = W(t_{11}) \cdot (W(w)+1) + 1$, $W((\sim t_{11})[x := w]) = W(\sim t_{11}) \cdot (W(w)+1) = (W(t_{11})+1) \cdot (W(w)+1) = W(t_{11}) \cdot (W(w)+1) + W(w) + 1$, and $W(t_{11}) \cdot (W(w)+1) + 1 < W(t_{11}) \cdot (W(w)+1) + W(w) + 1$.

Case 13. (subst-subst). We have $t_1 = t_{11}[x_1 := w_1][x_2 := w_2]$ and $t_2 = t_{21}[x_2 := w_2]$ where $t_{11}[x_1 := w_1] \longrightarrow^{\mathrm{x}} t_{21}$. By the induction hypothesis, $W(t_{21}) < W(t_{11}[x_1 := w_1])$. Then, $W(t_{21}[x_2 := w_2]) = W(t_{21}) \cdot (W(w_2)+1)$, $W(t_{11}[x_1 := w_1][x_2 := w_2]) = W(t_{11}[x_1 := w_1]) \cdot (W(w_2)+1)$, and $W(t_{21}) \cdot (W(w_2)+1) < W(t_{11}[x_1 := w_1]) \cdot (W(w_2)+1)$.

□

Lemma 322 (Well-foundedness of Explicit Substitution Descendant Relation)**.** *The explicit substitution descendant relation $\prec^{\mathrm{x}}$ is well-founded.*

Proof. Lemma 321 has proved that if $t_1 \longrightarrow^{\mathrm{x}} t_2$, then $W(t_2) < W(t_1)$, for any $t_1.t_2 \in \mathrm{TERM}_{\mathrm{exp}}$. For any $t \in \mathrm{TERM}_{\mathrm{exp}}$, the length of the descending chain with respect to $\prec^{\mathrm{x}}$ starting from t is bound by $W(t)$. Hence, the explicit substitution descendant relation $\prec^{\mathrm{x}}$ is well-founded. □

D.1.6 Bisimulation

Lemma 323 (Simulation: Explicit MetaML simulates (Substitutional) MetaML.)**.** *If $t^i_{a_1} \simeq t^i_{b_1}$ and $t^i_{a_1} \longrightarrow^i t^i_{a_2}$, then $t^i_{b_1} \longrightarrow^{i*} t^i_{b_2}$ and $t^i_{a_2} \simeq t^i_{b_2}$.*

Proof. We proceed by simultaneous induction on the structure of $t^i_{a_1} \simeq t^i_{b_1}$ and on the explicit substitution descendant relation $\prec^{\mathrm{x}} t^i_{b_1}$.

Case 1. (var-sim). This case is vacuous.

Case 2. (app-sim). Let $t^i_{a_1} = t^i_{a_{11}}\ t^i_{a_{12}}$ and $t^i_{b_1} = t^i_{b_{11}}\ t^i_{b_{12}}$ where $t^i_{a_{11}} \simeq t^i_{b_{11}}$ and $t^i_{a_{12}} \simeq t^i_{b_{12}}$. We proceed by cases on $t^i_{a_1} \longrightarrow^i t^i_{a_2}$.

Case i. (appL-i). Let $t^i_{a_{11}} \longrightarrow^i t^i_{a_{21}}$ and $t^i_{a_2} = t^i_{a_{21}}\ t^i_{a_{12}}$. By the induction hypothesis, $t^i_{b_{11}} \longrightarrow^{i*} t^i_{b_{21}}$ and $t^i_{a_{21}} \simeq t^i_{b_{21}}$. Then $t^i_{b_{11}}\ t^i_{b_{12}} \longrightarrow^{i*} t^i_{b_{21}}\ t^i_{b_{12}}$ and by (app-sim) $t^i_{a_{21}}\ t^i_{a_{12}} \simeq t^i_{b_{21}}\ t^i_{b_{12}}$.

Case ii. (appR-i). Let $t^i_{a_{11}} = v^i_{a_{11}}$, $t^i_{a_{12}} \longrightarrow^i t^i_{a_{22}}$ and $t^i_{a_2} = v^i_{a_{11}}\ t^i_{a_{22}}$. Given $v^i_{a_{11}} \simeq t^i_{b_{11}}$, by Lemma 317, $t^i_{b_{11}} \longrightarrow^{i*} v^i_{b_{11}}$ and $v^i_{a_{11}} \simeq v^i_{b_{11}}$. By the induction hypothesis, $t^i_{b_{12}} \longrightarrow^{i*} t^i_{b_{22}}$ and $t^i_{a_{22}} \simeq t^i_{b_{22}}$. Then $t^i_{b_{11}}\ t^i_{b_{12}} \longrightarrow^{i*} v^i_{b_{11}}\ t^i_{b_{12}} \longrightarrow^{i*} v^i_{b_{11}}\ t^i_{b_{22}}$ and by (app-sim) $v^i_{a_{11}}\ t^i_{a_{22}} \simeq v^i_{b_{11}}\ t^i_{b_{22}}$.

Case iii. (app-0). Let $t^0_{a_{11}} = \lambda x.t^0_{a_{111}}$, $t^0_{a_{12}} = v^0_{a_{12}}$ and $t^0_{a_2} = t^0_{a_{111}}[v^0_{a_{12}}/x]$. Given $\lambda x.t^0_{a_{111}} \simeq t^0_{b_{11}}$, by Lemma 317, $t^0_{b_{11}} \longrightarrow^{0*} v^0_{b_{11}}$ and $\lambda x.t^0_{a_{111}} \simeq v^0_{b_{11}}$. Given $v^0_{a_{12}} \simeq t^0_{b_{12}}$, by Lemma 317, $t^0_{b_{12}} \longrightarrow^{0*} v^0_{b_{12}}$ and $v^0_{a_{12}} \simeq v^0_{b_{12}}$. We proceed by cases on $\lambda x.t^0_{a_{111}} \simeq v^0_{b_{11}}$. The

only case is (lamu-sim), so let $v^0_{b_{11}} = \underline{\lambda}x.t^0_{b_{111}}$ and $t^0_{a_{111}} \simeq t^0_{b_{111}}$. Then $t^0_{b_{11}}\ t^0_{b_{12}} \longrightarrow^{0*} (\underline{\lambda}x.t^0_{b_{111}})\ t^0_{b_{12}} \longrightarrow^{0*} (\underline{\lambda}x.t^0_{b_{111}})\ v^0_{b_{12}} \longrightarrow^0 t^0_{b_{111}}[x := v^0_{b_{12}}]$. By (subst-sim), $t^0_{a_{111}}[v^0_{a_{12}}/x] \simeq t^0_{b_{111}}[x := v^0_{b_{12}}]$.

Case 3. (lam-sim). Let $t^{i+1}_{a_1} = \lambda x.t^{i+1}_{a_{11}}$ and $t^{i+1}_{b_1} = \lambda x.t^{i+1}_{b_{11}}$ where $t^{i+1}_{a_{11}} \simeq t^{i+1}_{b_{11}}$. We proceed by cases on $t^{i+1}_{a_1} \longrightarrow^{i+1} t^{i+1}_{a_2}$. The only case is (lambda-(i+1)). Then $t^{i+1}_{a_{11}} \longrightarrow^{i+1} t^{i+1}_{a_{21}}$ and $t^{i+1}_{a_2} = \lambda x.t^{i+1}_{a_{21}}$. By the induction hypothesis, $t^{i+1}_{b_{11}} \longrightarrow^{(i+1)*} t^{i+1}_{b_{21}}$ and $t^{i+1}_{a_{21}} \simeq t^{i+1}_{b_{21}}$. Then $\lambda x.t^{i+1}_{b_{11}} \longrightarrow^{(i+1)*} \lambda x.t^{i+1}_{b_{21}}$ and by (lam-sim) $\lambda x.t^{i+1}_{a_{21}} \simeq \lambda x.t^{i+1}_{b_{21}}$.

Case 4. (lamu-sim). We proceed by cases on i.

Case i. (0). Let $t^0_{a_1} = \lambda x.t^0_{a_{11}}$ and $t^0_{b_1} = \underline{\lambda}x.t^0_{b_{11}}$ where $t^0_{a_{11}} \simeq t^0_{b_{11}}$. This case is vacuous because $\lambda x.t^0_{a_{11}} \not\longrightarrow^0$.

Case ii. $(i+1)$. Let $t^{i+1}_{a_1} = \lambda x.t^0_{a_{11}}$ and $t^{i+1}_{b_1} = \underline{\lambda}x.t^0_{b_{11}}$ where $t^0_{a_{11}} \simeq t^0_{b_{11}}$. Given $\lambda x.t^0_{a_{11}} \in \text{VALUE}^0_{\text{sub}} \subseteq \text{VALUE}^{i+1}_{\text{sub}}$, we get $\lambda x.t^0_{a_{11}} \not\longrightarrow^{i+1}$. This case is vacuous as well.

Case 5. (code-sim). Let $t^i_{a_1} = \langle t^{i+1}_{a_{11}} \rangle$ and $t^i_{b_1} = \langle t^{i+1}_{b_{11}} \rangle$ where $t^{i+1}_{a_{11}} \simeq t^{i+1}_{b_{11}}$. We proceed by cases on $t^{i+1}_{a_1} \longrightarrow^{i+1} t^{i+1}_{a_2}$. The only case that applies is (code-i). Then $t^{i+1}_{a_{11}} \longrightarrow^{i+1} t^{i+1}_{a_{21}}$ and $t^i_{a_2} = \langle t^{i+1}_{a_{21}} \rangle$. By the induction hypothesis, $t^{i+1}_{b_{11}} \longrightarrow^{(i+1)*} t^{i+1}_{b_{21}}$ and $t^{i+1}_{a_{21}} \simeq t^{i+1}_{b_{21}}$. Then $\langle t^{i+1}_{b_{11}} \rangle \longrightarrow^{i*} \langle t^{i+1}_{b_{21}} \rangle$ and by (code-sim) $\langle t^{i+1}_{a_{21}} \rangle \simeq \langle t^{i+1}_{b_{21}} \rangle$.

Case 6. (splice-sim). Let $t^{i+1}_{a_1} = {\sim}t^i_{a_{11}}$ and $t^{i+1}_{b_1} = {\sim}t^i_{b_{11}}$ where $t^i_{a_{11}} \simeq t^i_{b_{11}}$. We proceed by cases on $t^{i+1}_{a_1} \longrightarrow^{i+1} t^{i+1}_{a_2}$.

Case i. (splice-(i+1)). Let $t^i_{a_{11}} \longrightarrow^i t^i_{a_{21}}$ and $t^{i+1}_{a_2} = {\sim}t^i_{a_{21}}$. By the induction hypothesis, $t^i_{b_{11}} \longrightarrow^{i*} t^i_{b_{21}}$ and $t^i_{a_{21}} \simeq t^i_{b_{21}}$. Then ${\sim}t^i_{b_{11}} \longrightarrow^{(i+1)*} {\sim}t^i_{b_{21}}$ and by (splice-sim) ${\sim}t^i_{a_{21}} \simeq {\sim}t^i_{b_{21}}$.

Case ii. (splice-1). Let $t^0_{a_{11}} = \langle v^1_{a_{111}} \rangle$, ${\sim}\langle v^1_{a_{111}} \rangle \longrightarrow^1 v^1_{a_{111}}$ and $t^1_{a_2} = v^1_{a_{111}}$. Given $\langle v^1_{a_{111}} \rangle \simeq t^0_{b_{11}}$ and $\langle v^1_{a_{111}} \rangle \in \text{VALUE}^0_{\text{sub}}$, by Lemma 317, $t^0_{b_{11}} \longrightarrow^{0*} v^0_{b_{21}}$ and $\langle v^1_{a_{111}} \rangle \simeq v^0_{b_{21}}$. We then proceed by cases on $\langle v^1_{a_{111}} \rangle \simeq v^0_{b_{21}}$. The only case is (code-sim), so let $v^0_{b_{21}} = \langle v^1_{b_{211}} \rangle$ where $v^1_{a_{111}} \simeq v^1_{b_{211}}$. Then ${\sim}t^0_{b_{11}} \longrightarrow^{1*} {\sim}\langle v^1_{b_{211}} \rangle \longrightarrow^1 v^1_{b_{211}}$.

Case 7. (run-sim). Let $t^i_{a_1} = !t^i_{a_{11}}$ and $t^i_{b_1} = !t^i_{b_{11}}$ where $t^i_{a_{11}} \simeq t^i_{b_{11}}$. We proceed by cases on $t^i_{a_1} \longrightarrow^i t_{a_2}$.

Case i. (run-i). Let $t^i_{a_{11}} \longrightarrow^i t^i_{a_{21}}$ and $t^i_{a_2} = !t^i_{a_{21}}$. By the induction hypothesis, $t^i_{b_{11}} \longrightarrow^{i*} t^i_{b_{21}}$ and $t^i_{a_{21}} \simeq t^i_{b_{21}}$. Then, $!t^i_{b_{11}} \longrightarrow^{i*} !t^i_{b_{21}}$ and by (run-sim) $!t^i_{a_{21}} \simeq !t^i_{b_{21}}$.

Case ii. (run-0). Let $t^0_{a_{11}} = \langle v^1_{a_{111}} \rangle$, $t^0_{a_1} = !\langle v^1_{a_{111}} \rangle$, $!\langle v^1_{a_{111}} \rangle \longrightarrow^0 v^1_{a_{111}}$ and $t^0_{a_2} = v^1_{a_{111}}$. Given $\langle v^1_{a_{111}} \rangle \simeq t^0_{b_{11}}$ and $\langle v^1_{a_{111}} \rangle \in \text{VALUE}^0_{\text{sub}}$, by Lemma 317, $t^0_{b_{11}} \longrightarrow^{0*} v^0_{b_{21}}$ and $\langle v^1_{a_{111}} \rangle \simeq v^0_{b_{21}}$. We proceed by cases on $\langle v^1_{a_{111}} \rangle \simeq v^0_{b_{21}}$. The only case that applies is (code-sim), so let $v^0_{b_{21}} = \langle v^1_{b_{211}} \rangle$ where $v^1_{a_{111}} \simeq v^1_{b_{211}}$. Then $!t^0_{b_{11}} \longrightarrow^{0*} !\langle v^1_{b_{211}} \rangle \longrightarrow^0 v^1_{b_{211}}$.

Case 8. (num-sim). This case is vacuous.

Case 9. (plus-sim). Let $t^i_{a_1} = t^i_{a_{11}} + t^i_{a_{12}}$ and $t^i_{b_1} = t^i_{b_{11}} + t^i_{b_{12}}$ where $t^i_{a_{11}} \simeq t^i_{b_{11}}$ and $t^i_{a_{12}} \simeq t^i_{b_{12}}$. We proceed by cases on $t^i_{a_1} \longrightarrow^i t^i_{a_2}$.

Case i. (plusL-i). Let $t^i_{a_{11}} \longrightarrow^i t^i_{a_{21}}$ and $t^i_{a_2} = t^i_{a_{21}} + t^i_{a_{12}}$. By the induction hypothesis, $t^i_{b_{11}} \longrightarrow^{i*} t^i_{b_{21}}$ and $t^i_{a_{21}} \simeq t^i_{b_{21}}$. Then $t^i_{b_{11}} + t^i_{b_{12}} \longrightarrow^{i*} t^i_{b_{21}} + t^i_{b_{12}}$ and by (plus-sim) $t^i_{a_{21}} + t^i_{a_{12}} \simeq t^i_{b_{21}} + t^i_{b_{12}}$.

Case ii. (plusR-i). Let $t^i_{a_{11}} = v^i_{a_{11}}, t^i_{a_{12}} \longrightarrow t^i_{a_{22}}$ and $t^i_{a_2} = v^i_{a_{11}} + t^i_{a_{22}}$. Given $v^i_{a_{11}} \simeq t^i_{b_{11}}$, by Lemma 317, $t^i_{b_{11}} \longrightarrow^{i*} v^i_{b_{11}}$ and $v^i_{a_{11}} \simeq v^i_{b_{11}}$. By the induction hypothesis, $t^i_{b_{12}} \longrightarrow^* t^i_{b_{22}}$ and $t^i_{a_{22}} \simeq t^i_{b_{22}}$. Then $t^i_{b_{11}} + t^i_{b_{12}} \longrightarrow^{i*} v^i_{b_{11}} + t^i_{b_{12}} \longrightarrow^{i*} v^i_{b_{11}} + t^i_{b_{22}}$ and by (plus-sim) $v^i_{a_{11}} + t^i_{a_{22}} \simeq v^i_{b_{11}} + t^i_{b_{22}}$.

Case iii. (plus-0). Let $t^0_{a_{11}} = n_1$, $t^0_{a_{12}} = n_2$, and $t^0_{a_2} = n$ where $n = n_1 + n_2$. Given $n_1 \simeq t^0_{b_{11}}$, by Lemma 317, $t^0_{b_{11}} \longrightarrow^{0*} v^0_{b_{11}}$ and $n_1 \simeq v^0_{b_{11}}$. Given $n_2 \simeq t^0_{b_{12}}$, by Lemma 317, $t^0_{b_{12}} \longrightarrow^{0*} v^0_{b_{12}}$ and $n_2 \simeq v^0_{b_{12}}$. We proceed by cases on $n_1 \simeq v^0_{b_{11}}$. The only case is (num-sim), so let $v^0_{b_{11}} = n_1$. We proceed by cases on $n_2 \simeq v^0_{b_{12}}$. The only case is (num-sim), so let $v^0_{b_{12}} = n_2$. Then $t^0_{b_{11}} + t^0_{b_{12}} \longrightarrow^{0*} n_1 + t^0_{b_{12}} \longrightarrow^{0*} n_1 + n_2 \longrightarrow n$ where $n = n_1 + n_2$. By (num-sim), $n \simeq n$.

Case 10. (subst-sim). Let $t^i_{a_1} = t^i_{a_{11}}[w_{a_{11}}/x]$ and $t^i_{b_1} = t^i_{b_{11}}[x := w_{b_{11}}]$ where $t^i_{a_{11}} \simeq t^i_{b_{11}}$ and $w_{a_{11}} \simeq w_{b_{11}}$. Given $t^i_{a_1} \simeq t^i_{b_{11}}[x := w_{b_{11}}]$, by Lemma 315, $t^i_{b_{11}}[x := w_{b_{11}}] \longrightarrow^{\mathrm{x}i*} s^i_{b_{21}}$ and $t^i_{a_1} \simeq s^i_{b_{21}}$. Then $s^i_{b_{21}} \prec^{\mathrm{x}} t^i_{b_1}$. If $t^i_{a_1} \longrightarrow^i t^i_{a_2}$, by the induction hypothesis, $s^i_{b_{21}} \longrightarrow^{i*} t^i_{b_2}$ and $t^i_{a_2} \simeq t^i_{b_2}$. We have $t^i_{b_1} \longrightarrow^{i*} s^i_{b_{21}} \longrightarrow^{i*} t^i_{b_2}$.

□

Remark 324. In the last case of the proof, given $t^i_{a_1} \simeq t^i_{b_1}$, $t^i_{a_1} \simeq s^i_{b_{21}}$ and $s^i_{b_{21}} \prec^{\mathrm{x}} t^i_{b_1}$, if $t^i_{a_1} \longrightarrow^i t^i_{a_2}$, by the induction hypothesis, $s^i_{b_{21}} \longrightarrow^{i*} t^i_{b_2}$ and $t^i_{a_2} \simeq t^i_{b_2}$.

Lemma 325 (Single-step explicit substitution reduction preserves bisimulation relation.)**.** *If* $t^i_{a_1} \simeq t^i_{b_1}$, $t^i_{b_1} \longrightarrow^{\mathrm{x}i} t^i_{b_2}$, *then* $t^i_{a_1} \simeq t^i_{b_2}$.

Proof. We proceed by structural induction on $t^i_{a_1} \simeq t^i_{b_1}$. Since $t^i_{b_1} \longrightarrow^{\mathrm{x}i} t^i_{b_2}$, only (subst-sim) applies. Let $t^i_{a_1} = t^i_{a_{11}}[w_{a_{11}}/x_1]$ and $t^i_{b_1} = t^i_{b_{11}}[x_1 := w_{b_{11}}]$ where $t^i_{a_{11}} \simeq t^i_{b_{11}}$ and $w_{a_{11}} \simeq w_{b_{11}}$. We proceed by cases on $t^i_{b_1} \longrightarrow^{\mathrm{x}i} t^i_{b_2}$.

Case 1. (var-eq-subst). Let $t^i_{b_{11}} = x_1$. We have $x_1[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i} w_{b_{11}}$. We proceed by cases on $t^i_{a_{11}} \simeq x_1$. The only case is (var-sim), thus we get $t^i_{a_{11}} = x_1$. Then $x_1[w_{a_{11}}/x_1] = w_{a_{11}}$ and $w_{a_{11}} \simeq w_{b_{11}}$.

Case 2. (var-df-subst). Let $t^i_{b_{11}} = x_2$ and $x_1 \not\equiv x_2$. We have $x_2[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i} x_2$. We proceed by cases on $t^i_{a_{11}} \simeq x_2$. The only case is (var-sim), thus we get $t^i_{a_{11}} = x_2$. Then $x_2[w_{a_{11}}/x_1] = x_2$ and $x_2 \simeq x_2$.

Case 3. (num-subst). Let $t^i_{b_{11}} = n$. We have $n[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i} n$. We proceed by cases on $t^i_{a_{11}} \simeq n$. The only case is (num-sim), thus we get $t^i_{a_{11}} = n$. Then $n[w_{a_{11}}/x_1] = n$ and $n \simeq n$.

Case 4. (app-subst). Let $t^i_{b_{11}} = t^i_{b_{111}}\ t^i_{b_{112}}$. We have $(t^i_{b_{111}}\ t^i_{b_{112}})[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i} t^i_{b_{111}}[x_1 := w_{b_{11}}]\ t^i_{b_{112}}[x_1 := w_{b_{11}}]$. We proceed by cases on $t^i_{a_{11}} \simeq t^i_{b_{111}}\ t^i_{b_{112}}$. The only case is (app-sim), thus we get $t^i_{a_{11}} = t^i_{a_{111}}\ t^i_{a_{112}}$ where $t^i_{a_{111}} \simeq t^i_{b_{111}}$ and $t^i_{a_{112}} \simeq t^i_{b_{112}}$. Then $(t^i_{a_{111}}\ t^i_{a_{112}})[w_{a_{11}}/x_1] = t^i_{a_{111}}[w_{a_{11}}/x_1]\ t^i_{a_{112}}[w_{a_{11}}/x_1]$ and by (subst-sim) and (app-sim) $t^i_{a_{111}}[w_{a_{11}}/x_1]\ t^i_{a_{112}}[w_{a_{11}}/x_1] \simeq t^i_{b_{111}}[x_1 := w_{b_{11}}]\ t^i_{b_{112}}[x_1 := w_{b_{11}}]$.

Case 5. (plus-subst). Let $t^i_{b_{11}} = t^i_{b_{111}} + t^i_{b_{112}}$. We have $(t^i_{b_{111}} + t^i_{b_{112}})[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i} t^i_{b_{111}}[x_1 := w_{b_{11}}] + t^i_{b_{112}}[x_1 := w_{b_{11}}]$. We proceed by cases on $t^i_{a_{11}} \simeq t^i_{b_{111}} + t^i_{b_{112}}$. The only case is (plus-sim), thus we get $t^i_{a_{11}} = t^i_{a_{111}} + t^i_{a_{112}}$ where $t^i_{a_{111}} \simeq t^i_{b_{111}}$ and $t^i_{a_{112}} \simeq t^i_{b_{112}}$. Then $(t^i_{a_{111}} + t^i_{a_{112}})[w_{a_{11}}/x_1] = t^i_{a_{111}}[w_{a_{11}}/x_1] + t^i_{a_{112}}[w_{a_{11}}/x_1]$ and by (subst-sim) and (plus-sim) $t^i_{a_{111}}[w_{a_{11}}/x_1] + t^i_{a_{112}}[w_{a_{11}}/x_1] \simeq t^i_{b_{111}}[x_1 := w_{b_{11}}] + t^i_{b_{112}}[x_1 := w_{b_{11}}]$.

Case 6. (lam-eq-subst). Let $t^i_{b_{11}} = \lambda x_1.t^i_{b_{111}}$. We have $(\lambda x_1.t^i_{b_{111}})[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i} \lambda x_1.t^i_{b_{111}}$. We proceed by cases on $t^i_{a_{11}} \simeq \lambda x_1.t^i_{b_{111}}$. The only case is (lam-subst), thus we get $t^i_{a_{11}} = \lambda x_1.t^i_{a_{111}}$ where $t^i_{a_{111}} \simeq t^i_{b_{111}}$. Then $(\lambda x_1.t^i_{a_{111}})[w_{a_{11}}/x_1] = \lambda x_1.t^i_{a_{111}}$ and $\lambda x_1.t^i_{a_{111}} \simeq \lambda x_1.t^i_{b_{111}}$.

Case 7. (lamu-eq-subst). Let $t^i_{b_{11}} = \underline{\lambda} x_1.t^0_{b_{111}}$. We have $(\underline{\lambda} x_1.t^0_{b_{111}})[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i} \underline{\lambda} x_1.t^0_{b_{111}}$. We proceed by cases on $t^i_{a_{11}} \simeq \underline{\lambda} x_1.t^0_{b_{111}}$. The only case is (lamu-sim), thus we get $t^i_{a_{11}} = \lambda x_1.t^0_{a_{111}}$ where $t^0_{a_{111}} \simeq t^0_{b_{111}}$. Then $(\lambda x_1.t^0_{a_{111}})[w_{a_{11}}/x_1] = \lambda x_1.t^0_{a_{111}}$ and $\lambda x_1.t^0_{a_{111}} \simeq \underline{\lambda} x_1.t^0_{b_{111}}$.

Case 8. (lam-df-subst). Let $t^i_{b_{11}} = \lambda x_2.t^i_{b_{111}}$ and $x_1 \not\equiv x_2$. We have $(\lambda x_2.t^i_{b_{111}})[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i} \lambda x_3.t^i_{b_{111}}[x_2 := x_3][x_1 := w_{b_{11}}]$ where $x_3 \notin FV(\lambda x_2.t^i_{b_{111}}) \cup FV(w_{b_{11}}) \cup \{x_1\}$. We proceed by cases on $t^i_{a_{11}} \simeq \lambda x_2.t^i_{b_{111}}$. The only case is (lam-sim), thus we get $t^i_{a_{11}} = \lambda x_2.t^i_{a_{111}}$ where $t^i_{a_{111}} \simeq t^i_{b_{111}}$. Then $(\lambda x_2.t^i_{a_{111}})[w_{a_{11}}/x_1] = \lambda x_4.t^i_{a_{111}}[x_4/x_2][w_{a_{11}}/x_1]$ where $x_4 \notin FV(\lambda x_2.t^i_{a_{111}}) \cup FV(w_{a_{11}}) \cup \{x_1\}$.
Let $x_5 \notin FV(\lambda x_2.t^i_{b_{111}}) \cup FV(w_{b_{11}}) \cup FV(\lambda x_2.t^i_{a_{111}}) \cup FV(w_{a_{11}}) \cup \{x_1\}$. We have $\lambda x_3.t^i_{b_{111}}[x_2 := x_3][x_1 := w_{b_{11}}] \sim_\alpha \lambda x_5.t^i_{b_{111}}[x_2 := x_5][x_1 := w_{b_{11}}]$ and $\lambda x_4.t^i_{a_{111}}[x_4/x_2][w_{a_{11}}/x_1] \sim_\alpha \lambda x_5.t^i_{a_{111}}[x_5/x_2][w_{a_{11}}/x_1]$.
By (subst-sim) and (lam-sim), we get $\lambda x_5.t^i_{a_{111}}[x_5/x_2][w_{a_{11}}/x_1] \simeq \lambda x_5.t^i_{b_{111}}[x_2 := x_5][x_1 := w_{b_{11}}]$.
Hence, $\lambda x_4.t^i_{a_{111}}[x_4/x_2][w_{a_{11}}/x_1] \simeq \lambda x_3.t^i_{b_{111}}[x_2 := x_3][x_1 := w_{b_{11}}]$.

Case 9. (lamu-df-subst). Let $t^i_{b_{11}} = \underline{\lambda} x_2.t^0_{b_{111}}$ and $x_1 \not\equiv x_2$. We have $(\underline{\lambda} x_2.t^i_{b_{111}})[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i} \underline{\lambda} x_3.t^i_{b_{111}}[x_2 := x_3][x_1 := w_{b_{11}}]$ where $x_3 \notin FV(\underline{\lambda} x_2.t^i_{b_{111}}) \cup FV(w_{b_{11}}) \cup \{x_1\}$. We proceed by cases on $t^i_{a_{11}} \simeq \underline{\lambda} x_2.t^i_{b_{111}}$. The only case is (lamu-sim), thus we get $t^i_{a_{11}} = \lambda x_2.t^i_{a_{111}}$ where $t^i_{a_{111}} \simeq t^i_{b_{111}}$. Then $(\lambda x_2.t^i_{a_{111}})[w_{a_{11}}/x_1] = \lambda x_4.t^i_{a_{111}}[x_4/x_2][w_{a_{11}}/x_1]$ where $x_4 \notin FV(\lambda x_2.t^i_{a_{111}}) \cup FV(w_{a_{11}}) \cup \{x_1\}$.
Let $x_5 \notin FV(\underline{\lambda} x_2.t^i_{b_{111}}) \cup FV(w_{b_{11}}) \cup FV(\lambda x_2.t^i_{a_{111}}) \cup FV(w_{a_{11}}) \cup \{x_1\}$. We have $\underline{\lambda} x_3.t^i_{b_{111}}[x_2 := x_3][x_1 := w_{b_{11}}] \sim_\alpha \underline{\lambda} x_5.t^i_{b_{111}}[x_2 := x_5][x_1 := w_{b_{11}}]$ and $\lambda x_4.t^i_{a_{111}}[x_4/x_2][w_{a_{11}}/x_1] \sim_\alpha \lambda x_5.t^i_{a_{111}}[x_5/x_2][w_{a_{11}}/x_1]$.
By (subst-sim) and (lamu-sim), we get $\lambda x_5.t^i_{a_{111}}[x_5/x_2][w_{a_{11}}/x_1] \simeq \underline{\lambda} x_5.t^i_{b_{111}}[x_2 := x_5][x_1 := w_{b_{11}}]$.
Hence, $\lambda x_4.t^i_{a_{111}}[x_4/x_2][w_{a_{11}}/x_1] \simeq \underline{\lambda} x_3.t^i_{b_{111}}[x_2 := x_3][x_1 := w_{b_{11}}]$.

Case 10. (code-subst). Let $t^i_{b_{11}} = \langle t^{i+1}_{b_{111}} \rangle$. We have $\langle t^{i+1}_{b_{111}} \rangle[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i} \langle t^{i+1}_{b_{111}}[x_1 := w_{b_{11}}] \rangle$. We proceed by cases on $t^i_{a_{11}} \simeq \langle t^{i+1}_{b_{111}} \rangle$. The only case is (code-sim), thus we get $t^i_{a_{11}} = \langle t^{i+1}_{a_{111}} \rangle$ where $t^{i+1}_{a_{111}} \simeq t^{i+1}_{b_{111}}$. Then $\langle t^{i+1}_{a_{111}} \rangle[w_{a_{11}}/x_1] = \langle t^{i+1}_{a_{111}}[w_{a_{11}}/x_1] \rangle$ and by (subst-sim) and (code-sim) $\langle t^{i+1}_{a_{111}}[w_{a_{11}}/x_1] \rangle \simeq \langle t^{i+1}_{b_{111}}[x_1 := w_{b_{11}}] \rangle$.

Case 11. (run-subst). Let $t^i_{b_{11}} = !t^i_{b_{111}}$. We have $(!t^i_{b_{111}})[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i} !(t^i_{b_{111}}[x_1 := w_{b_{11}}])$. We proceed by cases on $t^i_{a_{11}} \simeq !t^i_{b_{111}}$. The only case is (run-sim), thus we get $t^i_{a_{11}} = !t^i_{a_{111}}$ where $t^i_{a_{111}} \simeq t^i_{b_{111}}$. Then $(!t^i_{a_{111}})[w_{a_{11}}/x_1] = !(t^i_{a_{111}}[w_{a_{11}}/x_1])$ and by (subst-sim) and (run-sim) $!(t^i_{a_{111}}[w_{a_{11}}/x_1]\rangle \simeq !(t^i_{b_{111}}[x_1 := w_{b_{11}}])$.

Case 12. (splice-subst). Let $t^{i+1}_{b_{11}} = {\sim}t^i_{b_{111}}$. We have $({\sim}t^i_{b_{111}})[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}(i+1)} {\sim}(t^i_{b_{111}}[x_1 := w_{b_{11}}])$. We proceed by cases on $t^{i+1}_{a_{11}} \simeq {\sim}t^i_{b_{111}}$. The only case is (splice-sim), thus we get $t^{i+1}_{a_{11}} = {\sim}t^i_{a_{111}}$ where $t^i_{a_{111}} \simeq t^i_{b_{111}}$. Then $({\sim}t^i_{a_{111}})[w_{a_{11}}/x_1] = {\sim}(t^i_{a_{111}}[w_{a_{11}}/x_1])$ and by (subst-sim) and (splice-sim) ${\sim}(t^i_{a_{111}}[w_{a_{11}}/x_1]\rangle \simeq {\sim}(t^i_{b_{111}}[x_1 := w_{b_{11}}])$.

Case 13. (subst-subst). Let $t^i_{b_{11}} = t^i_{b_{111}}[x_2 := w_{b_{12}}]$. We have $(t^i_{b_{111}}[x_2 := w_{b_{12}}])[x_1 := w_{b_{11}}] \longrightarrow^{\mathrm{x}i} t^i_{b_{121}}[x_1 := w_{b_{11}}]$ where $t^i_{b_{111}}[x_2 := w_{b_{12}}] \longrightarrow^{\mathrm{x}i} t^i_{b_{121}}$. By the induction hypothesis, $t^i_{a_{11}} \simeq t^i_{b_{121}}$. Then by (subst-sim) $t^i_{a_{11}}[w_{a_{11}}/x_1] \simeq t^i_{b_{121}}[x_1 := w_{b_{11}}]$.

□

Lemma 326 (Simulation: (Substitutional) MetaML simulates Explicit MetaML.)**.** *If $t^i_{a_1} \simeq t^i_{b_1}$ and $t^i_{b_1} \longrightarrow^i t^i_{b_2}$, then $t^i_{a_1} \longrightarrow^{i*} t^i_{a_2}$ and $t^i_{a_2} \simeq t^i_{b_2}$.*

Proof. We proceed by induction on the structure of $t^i_{a_1} \simeq t^i_{b_1}$.

Case 1. (var-sim). This case is vacuous.

Case 2. (app-sim). Let $t^i_{a_1} = t^i_{a_{11}}\ t^i_{a_{12}}$ and $t^i_{b_1} = t^i_{b_{11}}\ t^i_{b_{12}}$ where $t^i_{a_{11}} \simeq t^i_{b_{11}}$ and $t^i_{a_{12}} \simeq t^i_{b_{12}}$. We proceed by cases on $t^i_{b_1} \longrightarrow^i t^i_{b_2}$.

Case i. (appL-i). Let $t^i_{b_{11}} \longrightarrow^i t^i_{b_{21}}$ and $t^i_{b_2} = t^i_{b_{21}}\ t^i_{b_{12}}$. By the induction hypothesis, $t^i_{a_{11}} \longrightarrow^{i*} t^i_{a_{21}}$ and $t^i_{a_{21}} \simeq t^i_{b_{21}}$. Then $t^i_{a_{11}}\ t^i_{a_{12}} \longrightarrow^{i*} t^i_{a_{21}}\ t^i_{a_{12}}$ and by (app-sim) $t^i_{a_{21}}\ t^i_{a_{12}} \simeq t^i_{b_{21}}\ t^i_{b_{12}}$.

Case ii. (appR-i). Let $t^i_{b_{11}} = v^i_{b_{11}}$, $t^i_{b_{12}} \longrightarrow^i t^i_{b_{22}}$ and $t^i_{b_2} = v^i_{b_{11}}\ t^i_{b_{22}}$. Given $t^i_{a_{11}} \simeq v^i_{b_{11}}$, by Lemma 316, $t^i_{a_{11}} \longrightarrow^{i*} v^i_{a_{11}}$ and $v^i_{a_{11}} \simeq v^i_{b_{11}}$. By the induction hypothesis, $t^i_{a_{12}} \longrightarrow^{i*} t^i_{a_{22}}$ and $t^i_{a_{22}} \simeq t^i_{b_{22}}$. Then $t^i_{a_{11}}\ t^i_{a_{12}} \longrightarrow^{i*} v^i_{a_{11}}\ t^i_{a_{12}} \longrightarrow^{i*} v^i_{a_{11}}\ t^i_{a_{22}}$ and by (app-sim) $v^i_{a_{11}}\ t^i_{a_{22}} \simeq v^i_{b_{11}}\ t^i_{b_{22}}$.

Case iii. (app-0). Let $t^0_{b_{11}} = \underline{\lambda}x.t^0_{b_{111}}$, $t^0_{b_{12}} = v^0_{b_{12}}$ and $t^0_{b_2} = t^0_{b_{111}}[x := v^0_{b_{12}}]$. Given $t^0_{a_{11}} \simeq \underline{\lambda}x.t^0_{b_{111}}$, by Lemma 316, $t^0_{a_{11}} \in \mathrm{VALUE}^0_{\mathrm{sub}}$. Given $t^0_{a_{12}} \simeq v^0_{b_{12}}$, by Lemma 316, $t^0_{a_{12}} \in \mathrm{VALUE}^0_{\mathrm{sub}}$. Let $t^0_{a_{11}} = v^0_{a_{11}}$ and $t^0_{a_{12}} = v^0_{a_{12}}$. We proceed by cases on $v^0_{a_{11}} \simeq \underline{\lambda}x.t^0_{b_{111}}$. The only case is (lamu-sim), so let $v^0_{a_{11}} = \lambda x.t^0_{a_{111}}$ where $t^0_{a_{111}} \simeq t^0_{b_{111}}$. Then $(\lambda x.t^0_{a_{111}})\ v^0_{a_{12}} \longrightarrow^0 t^0_{a_{111}}[v^0_{a_{12}}/x]$. By (subst-sim), $t^0_{a_{111}}[v^0_{a_{12}}/x] \simeq t^0_{b_{111}}[x := v^0_{b_{12}}]$.

Case 3. (lam-sim). We proceed by cases on i.

Case i. (0). Let $t^0_{a_1} = \lambda x.t^0_{a_{11}}$ and $t^0_{b_1} = \lambda x.t^0_{b_{11}}$ where $t^0_{a_{11}} \simeq t^0_{b_{11}}$. We proceed by cases on $t^0_{b_1} \longrightarrow^0 t^0_{b_2}$. The only case is (lambda-0), so let $\lambda x.t^0_{b_{11}} \longrightarrow^0 \underline{\lambda}x.t^0_{b_{11}}$ and $t^0_{b_2} = \underline{\lambda}x.t^0_{b_{11}}$. Then $\lambda x.t^0_{a_{11}} \longrightarrow^{0*} \lambda x.t^0_{a_{11}}$ and by (lamu-sim) $\lambda x.t^0_{a_{11}} \simeq \underline{\lambda}x.t^0_{b_{11}}$.

Case ii. $(i+1)$. Let $t_{a_1}^{i+1} = \lambda x.t_{a_{11}}^{i+1}$ and $t_{b_1}^{i+1} = \lambda x.t_{b_{11}}^{i+1}$ where $t_{a_{11}}^{i+1} \simeq t_{b_{11}}^{i+1}$. We proceed by cases on $t_{b_1}^{i+1} \longrightarrow^{i+1} t_{b_2}^{i+1}$. The only case is (lambda-(i+1)), so let $t_{b_{11}}^{i+1} \longrightarrow^{i+1} t_{b_{21}}^{i+1}$ and $t_{b_2}^{i+1} = \lambda x.t_{b_{21}}^{i+1}$. By the induction hypothesis, $t_{a_{11}}^{i+1} \longrightarrow^{(i+1)*} t_{a_{21}}^{i+1}$ and $t_{a_{21}}^{i+1} \simeq t_{b_{21}}^{i+1}$. Then $\lambda x.t_{a_{11}}^{i+1} \longrightarrow^{(i+1)*} \lambda x.t_{a_{21}}^{i+1}$ and by (lam-sim) $\lambda x.t_{a_{21}}^{i+1} \simeq \lambda x.t_{b_{21}}^{i+1}$.

Case 4. (lamu-sim). Let $t_{a_1}^{i} = \lambda x.t_{a_{11}}^{0}$ and $t_{b_1}^{i} = \underline{\lambda} x.t_{b_{11}}^{0}$ where $t_{a_{11}}^{0} \simeq t_{b_{11}}^{0}$. This case is vacuous because $\underline{\lambda} x.t_{b_{11}}^{0} \not\longrightarrow^{i}$.

Case 5. (code-sim). Let $t_{a_1}^{i} = \langle t_{a_{11}}^{i+1} \rangle$ and $t_{b_1}^{i} = \langle t_{b_{11}}^{i+1} \rangle$ where $t_{a_{11}}^{i+1} \simeq t_{b_{11}}^{i+1}$. We proceed by cases on $t_{b_1}^{i+1} \longrightarrow^{i+1} t_{b_2}^{i+1}$. The only case that applies is (code-i), so let $t_{b_{11}}^{i+1} \longrightarrow^{i+1} t_{b_{21}}^{i+1}$ and $t_{b_2}^{i} = \langle t_{b_{21}}^{i+1} \rangle$. By the induction hypothesis, $t_{a_{11}}^{i+1} \longrightarrow^{(i+1)*} t_{a_{21}}^{i+1}$ and $t_{a_{21}}^{i+1} \simeq t_{b_{21}}^{i+1}$. Then $\langle t_{a_{11}}^{i+1} \rangle \longrightarrow^{i*} \langle t_{a_{21}}^{i+1} \rangle$ and by (code-sim) $\langle t_{a_{21}}^{i+1} \rangle \simeq \langle t_{b_{21}}^{i+1} \rangle$.

Case 6. (splice-sim). Let $t_{a_1}^{i+1} = {\sim} t_{a_{11}}^{i}$ and $t_{b_1}^{i+1} = {\sim} t_{b_{11}}^{i}$ where $t_{a_{11}}^{i} \simeq t_{b_{11}}^{i}$. We proceed by cases on $t_{b_1}^{i+1} \longrightarrow^{i+1} t_{b_2}^{i+1}$.

Case i. (splice-(i+1)). Let $t_{b_{11}}^{i} \longrightarrow^{i} t_{b_{21}}^{i}$ and $t_{b_2}^{i+1} = {\sim} t_{b_{21}}^{i}$. By the induction hypothesis, $t_{a_{11}}^{i} \longrightarrow^{i*} t_{a_{21}}^{i}$ and $t_{a_{21}}^{i} \simeq t_{b_{21}}^{i}$. Then ${\sim} t_{a_{11}}^{i} \longrightarrow^{(i+1)*} {\sim} t_{a_{21}}^{i}$ and by (splice-sim) ${\sim} t_{a_{21}}^{i} \simeq {\sim} t_{b_{21}}^{i}$.

Case ii. (splice-1). Let $t_{b_{11}}^{0} = \langle v_{b_{111}}^{1} \rangle$, ${\sim}\langle v_{b_{111}}^{1} \rangle \longrightarrow^{1} v_{b_{111}}^{1}$ and $t_{b_2}^{1} = v_{b_{111}}^{1}$. Given $t_{a_{11}}^{0} \simeq \langle v_{b_{111}}^{1} \rangle$ and $\langle v_{b_{111}}^{1} \rangle \in \text{VALUE}_{\text{exp}}^{0}$, by Lemma 316, $t_{a_{11}}^{0} \longrightarrow^{0*} v_{a_{21}}^{0}$ and $v_{a_{21}}^{0} \simeq \langle v_{b_{111}}^{1} \rangle$. We then proceed by cases on $v_{a_{21}}^{0} \simeq \langle v_{b_{111}}^{1} \rangle$. The only case is (code-sim), so let $v_{a_{21}}^{0} = \langle v_{a_{211}}^{1} \rangle$ where $v_{a_{211}}^{1} \simeq v_{b_{111}}^{1}$. Then ${\sim} t_{a_{11}}^{0} \longrightarrow^{1*} {\sim}\langle v_{a_{211}}^{1} \rangle \longrightarrow^{1} v_{a_{211}}^{1}$.

Case 7. (run-sim). Let $t_{a_1}^{i} = !t_{a_{11}}^{i}$ and $t_{b_1}^{i} = !t_{b_{11}}^{i}$ where $t_{a_{11}}^{i} \simeq t_{b_{11}}^{i}$. We proceed by cases on $t_{b_1}^{i} \longrightarrow^{i} t_{b_2}$.

Case i. (run-i). Let $t_{b_{11}}^{i} \longrightarrow^{i} t_{b_{21}}^{i}$ and $t_{b_2}^{i} = !t_{b_{21}}^{i}$. By the induction hypothesis, $t_{a_{11}}^{i} \longrightarrow^{i*} t_{a_{21}}^{i}$ and $t_{a_{21}}^{i} \simeq t_{b_{21}}^{i}$. Then $!t_{a_{11}}^{i} \longrightarrow^{i*} !t_{a_{21}}^{i}$ and by (run-sim) $!t_{a_{21}}^{i} \simeq !t_{b_{21}}^{i}$.

Case ii. (run-0). Let $t_{b_{11}}^{0} = \langle v_{b_{111}}^{1} \rangle$, $t_{b_1}^{0} = !\langle v_{b_{111}}^{1} \rangle$, $!\langle v_{b_{111}}^{1} \rangle \longrightarrow^{0} v_{b_{111}}^{1}$ and $t_{b_2}^{0} = v_{b_{111}}^{1}$. Given $t_{a_{11}}^{0} \simeq \langle v_{b_{111}}^{1} \rangle$ and $\langle v_{b_{111}}^{1} \rangle \in \text{VALUE}_{\text{exp}}^{0}$, by Lemma 316, $t_{a_{11}}^{0} \longrightarrow^{0*} v_{a_{21}}^{0}$ and $v_{a_{21}}^{0} \simeq \langle v_{b_{111}}^{1} \rangle$. We proceed by cases on $v_{a_{21}}^{0} \simeq \langle v_{b_{111}}^{1} \rangle$. The only case that applies is (code-sim), so let $v_{a_{21}}^{0} = \langle v_{a_{211}}^{1} \rangle$ where $v_{a_{211}}^{1} \simeq v_{b_{111}}^{1}$. Then $!t_{a_{11}}^{0} \longrightarrow^{0*} !\langle v_{a_{211}}^{1} \rangle \longrightarrow^{0} v_{a_{211}}^{1}$.

Case 8. (num-sim). This case is vacuous.

Case 9. (plus-sim). Let $t_{a_1}^{i} = t_{a_{11}}^{i} + t_{a_{12}}^{i}$ and $t_{b_1}^{i} = t_{b_{11}}^{i} + t_{b_{12}}^{i}$ where $t_{a_{11}}^{i} \simeq t_{b_{11}}^{i}$ and $t_{a_{12}}^{i} \simeq t_{b_{12}}^{i}$. We proceed by cases on $t_{b_1}^{i} \longrightarrow^{i} t_{b_2}^{i}$.

Case i. (plusL-i). Let $t_{b_{11}}^{i} \longrightarrow^{i} t_{b_{21}}^{i}$ and $t_{b_2}^{i} = t_{b_{21}}^{i} + t_{b_{12}}^{i}$. By the induction hypothesis, $t_{a_{11}}^{i} \longrightarrow^{i*} t_{a_{21}}^{i}$ and $t_{a_{21}}^{i} \simeq t_{b_{21}}^{i}$. Then $t_{a_{11}}^{i} + t_{a_{12}}^{i} \longrightarrow^{i*} t_{a_{21}}^{i} + t_{a_{12}}^{i}$ and by (plus-sim) $t_{a_{21}}^{i} + t_{a_{12}}^{i} \simeq t_{b_{21}}^{i} + t_{b_{12}}^{i}$.

Case ii. (plusR-i). Let $t_{b_{11}}^{i} = v_{b_{11}}^{i}$, $t_{b_{12}}^{i} \longrightarrow^{i} t_{b_{22}}^{i}$ and $t_{b_2}^{i} = v_{b_{11}}^{i} + t_{b_{22}}^{i}$. Given $t_{a_{11}}^{i} \simeq v_{b_{11}}^{i}$, by Lemma 316, $t_{a_{11}}^{i} \longrightarrow^{i*} v_{a_{11}}^{i}$ and $v_{a_{11}}^{i} \simeq v_{b_{11}}^{i}$. By the induction hypothesis, $t_{a_{12}}^{i} \longrightarrow^{i*}$

$t^i_{a_{22}}$ and $t^i_{a_{22}} \simeq t^i_{b_{22}}$. Then $t^i_{a_{11}} + t^i_{a_{12}} \longrightarrow^{i*} v^i_{a_{11}} + t^i_{a_{12}} \longrightarrow^{i*} v^i_{a_{11}} + t^i_{a_{22}}$ and by (app-sim) $v^i_{a_{11}} + t^i_{a_{22}} \simeq v^i_{b_{11}} + t^i_{b_{22}}$.

Case iii. (plus-0). Let $t^0_{b_{11}} = n_1$, $t^0_{b_{12}} = n_2$ and $t^0_{b_2} = n$ where $n = n_1 + n_2$. Given $t^0_{a_{11}} \simeq n_1$, by Lemma 316, $t^0_{a_{11}} \longrightarrow^{0*} v^0_{a_{11}}$ and $v^0_{a_{11}} \simeq n_1$. Given $t^0_{a_{12}} \simeq n_2$, by Lemma 316, $t^0_{a_{12}} \longrightarrow^{0*} v^0_{a_{12}}$ and $v^0_{a_{12}} \simeq n_2$. We proceed by cases on $v^0_{a_{11}} \simeq n_1$. The only case is (num-sim), so let $v^0_{a_{11}} = n_1$. We proceed by cases on $v^0_{a_{12}} \simeq n_2$. The only case is (num-sim), so let $v^0_{a_{12}} = n_2$. Then $t^0_{a_{11}} + t^0_{a_{12}} \longrightarrow^{0*} n_1 + t^0_{a_{12}} \longrightarrow^{0*} n_1 + n_2 \longrightarrow^0 n$ where $n = n_1 + n_2$. By (num-sim), $n \simeq n$.

Case 10. (subst-sim). Let $t^i_{a_1} = t^i_{a_{11}}[w_{a_{11}}/x]$ and $t^i_{b_1} = t^i_{b_{11}}[x := w_{b_{11}}]$ where $t^i_{a_{11}} \simeq t^i_{b_{11}}$ and $w_{a_{11}} \simeq w_{b_{11}}$. Given $t^i_{a_1} \simeq t^i_{b_{11}}[x := w_{b_{11}}]$, by Lemma 315, $t^i_{b_{11}}[x := w_{b_{11}}] \longrightarrow^{x*} s^i_{b_{21}}$ and $t^i_{a_1} \simeq s^i_{b_{21}}$. Since $t^i_{b_{11}}[x := w_{b_{11}}]$ is not in substitution normal form but $s^i_{b_{21}}$ is in substitution normal form, $t^i_{b_{11}}[x := w_{b_{11}}] \longrightarrow^{xi} t^i_{b_{21}} \longrightarrow^{xi*} s^i_{b_{21}}$. By Lemma 325, $t^i_{a_1} \simeq t^i_{b_{21}}$. Then $t^i_{b_1} \longrightarrow^i t^i_{b_{21}}$, $t^i_{a_1} \longrightarrow^{i*} t^i_{a_1}$ and $t^i_{a_1} \simeq t^i_{b_{21}}$.

□

D.1.7 Soundness and Completeness

Theorem 327 (Soundness of Explicit MetaML w.r.t. (Substitutional) MetaML)**.** *If $t^i_{a_1} \simeq t^i_{b_1}$ and $t^i_{a_1} \longrightarrow^{i*} v^i_{a_2}$ in (Substitutional) MetaML, then $t^i_{b_1} \longrightarrow^{i*} v^i_{b_2}$ in Explicit MetaML and $v^i_{a_2} \simeq v^i_{b_2}$.*

Proof. We proceed by induction on the length of $t^i_{a_1} \longrightarrow^{i*} v^i_{a_2}$.

Case 1. (0). Let $t^i_{a_1} = v^i_{a_2}$. By Lemma 317, $t^i_{b_1} \longrightarrow^{i*} v^i_{b_2}$ and $v^i_{a_2} \simeq v^i_{b_2}$.

Case 2. $(n+1)$. Let $t^i_{a_1} \longrightarrow^i t^i_{a_2} \longrightarrow^{i(n)} v^i_{a_2}$. Given $t^i_{a_1} \simeq t^i_{b_1}$ and $t^i_{a_1} \longrightarrow^i t^i_{a_2}$, by Lemma 323, $t^i_{b_1} \longrightarrow^{i*} t^i_{b_2}$ and $t^i_{a_2} \simeq t^i_{b_2}$. Given $t^i_{a_2} \simeq t^i_{b_2}$ and $t^i_{a_2} \longrightarrow^{i(n)} v^i_{a_2}$, by the induction hypothesis, $t^i_{b_2} \longrightarrow^{i*} v^i_{b_2}$ and $v^i_{a_2} \simeq v^i_{b_2}$. We have $t^i_{b_1} \longrightarrow^{i*} t^i_{b_2} \longrightarrow^{i*} v^i_{b_2}$ and $v^i_{a_2} \simeq v^i_{b_2}$.

□

Theorem 328 (Completeness of Explicit MetaML w.r.t. (Substitutional) MetaML)**.** *If $t^i_{a_1} \simeq t^i_{b_1}$ and $t^i_{b_1} \longrightarrow^{i*} v^i_{b_2}$ in Explicit MetaML, then $t^i_{a_1} \longrightarrow^{i*} v^i_{a_2}$ in (Substitutional) MetaML and $v^i_{a_2} \simeq v^i_{b_2}$.*

Proof. We proceed by induction on the length of $t^i_{b_1} \longrightarrow^{i*} v^i_{b_2}$.

Case 1. (0). Let $t^i_{b_1} = v^i_{b_2}$. By Lemma 316, $t^i_{a_1} \in \text{VALUE}^i_{\text{sub}}$. Let $v^i_{a_2} = t^i_{a_1}$. We have $t^i_{a_1} \longrightarrow^{i*} v^i_{a_2}$ and $v^i_{a_2} \simeq v^i_{b_2}$.

Case 2. $(n+1)$. Let $t^i_{b_1} \longrightarrow^i t^i_{b_2} \longrightarrow^{i(n)} v^i_{b_2}$. Given $t^i_{a_1} \simeq t^i_{b_1}$ and $t^i_{b_1} \longrightarrow^i t^i_{b_2}$, by Lemma 326, $t^i_{a_1} \longrightarrow^{i*} t^i_{a_2}$ and $t^i_{a_2} \simeq t^i_{b_2}$. Given $t^i_{a_2} \simeq t^i_{b_2}$ and $t^i_{b_2} \longrightarrow^{i(n)} v^i_{b_2}$, by the induction hypothesis, $t^i_{a_2} \longrightarrow^{i*} v^i_{a_2}$ and $v^i_{a_2} \simeq v^i_{b_2}$. We have $t^i_{a_1} \longrightarrow^{i*} v^i_{a_2}$ and $v^i_{a_2} \simeq v^i_{b_2}$.

□

D.1.7.1 An alternative proof.

We demonstrate a different proof of Theorem 328 which does not use Lemma 326. We start with two lemmas. Their proofs are omitted.

Lemma 329. *If $t^i_{a_1} \simeq t^i_{b_1}$, $t^i_{b_1} \longrightarrow t^i_{b_2}$ and $t_{a_1} \not\simeq t_{b_2}$, then $t^i_{a_1} \longrightarrow^i t^i_{a_2}$.*

Remark 330. Lemma 329 does not imply whether or not $t^i_{a_2} \simeq t^i_{b_2}$.

Lemma 331. *If $t^i_{a_1} \simeq t^i_{b_1}$ and $t^i_{a_1} \longrightarrow t^i_{a_2}$, then $t^i_{b_1} \longrightarrow^{i+} t^i_{b_2}$ and $t^i_{a_2} \simeq t^i_{b_2}$.*

Remark 332. Lemma 331 is stronger than Lemma 323. In other words, Lemma 331 implies Lemma 323.

Theorem 333 (Completeness of Explicit ISWIM w.r.t. (Substitutional) ISWIM)**.** *If $t^i_{a_1} \simeq t^i_{b_1}$ and $t^i_{b_1} \longrightarrow^{i*} v^i_{b_2}$ in Explicit MetaML, then $t^i_{a_1} \longrightarrow^{i*} v^i_{a_2}$ in Substitutional MetaML and $v^i_{a_2} \simeq v^i_{b_2}$.*

Proof. We proceed by induction on the length of $t^i_{b_1} \longrightarrow^{i*} v^i_{b_2}$.

Case 1. (0). Let $t^i_{b_1} = v^i_{b_2}$. By Lemma 316, $t^i_{a_1} \in \text{VALUE}_{\text{sub}}$. Let $v^i_{a_2} = t^i_{a_1}$. We have $t^i_{a_1} \longrightarrow^{i*} v^i_{a_2}$ and $v^i_{a_2} \simeq v^i_{b_2}$.

Case 2. $(n+1)$. Let $t^i_{b_1} \longrightarrow^i t^i_{b_2} \longrightarrow^{i(n)} v^i_{b_2}$. We proceed by cases on $t^i_{a_1} \simeq t^i_{b_1}$, in particular on whether it is (subst-sim) or not.

Case i. (subst-sim). Let $t^i_{a_1} = t^i_{a_{11}}[w_{a_1}/x]$ and $t^i_{b_1} = t^i_{b_{11}}[x := w_{b_1}]$ where $t^i_{a_{11}} \simeq t^i_{b_{11}}$ and $w_{a_1} \simeq w_{b_1}$. By Lemma 315, $t^i_{b_1} \longrightarrow^{\text{x}i*} s^i_{b_2}$. Observe that $t^i_{b_1}$ is not in substitution normal form but $s^i_{b_2}$ is in substitution normal form. By the determinism of the small-step semantics, $t^i_{b_1} \longrightarrow^{\text{x}i} t^i_{b_2} \longrightarrow^{\text{x}i(p)} s^i_{b_2} \longrightarrow^{i(q)} v^i_{b_2}$ and $p+q=n$ where $p,q \geq 0$. By Lemma 325, $t^i_{a_1} \simeq t^i_{b_2}$. By the induction hypothesis, $t^i_{a_1} \longrightarrow^{i*} v^i_{a_2}$ and $v^i_{a_2} \simeq v^i_{b_2}$.

Case ii. (other cases). We proceed by cases on whether $t^i_{a_1} \simeq t^i_{b_2}$.

Case a. ($t^i_{a_1} \simeq t^i_{b_2}$). Then by the induction hypothesis, $t^i_{a_1} \longrightarrow^{i*} v^i_{a_2}$ and $v^i_{a_2} \simeq v^i_{b_2}$.

Case b. ($t^i_{a_1} \not\simeq t^i_{b_2}$). By Lemma 329, $t^i_{a_1} \longrightarrow t^i_{a_2}$. By Lemma 331, $t^i_{b_1} \longrightarrow^{i+} t^i_{b_3}$ and $t^i_{a_2} \simeq t^i_{b_3}$. By the determinism of the small-step semantics, $t^i_{b_1} \longrightarrow^i t^i_{b_2} \longrightarrow^{i(p)} t^i_{b_3} \longrightarrow^{i(q)} v^i_{b_2}$ and $p+q=n$ where $p,q \geq 0$. By the induction hypothesis, $t^i_{a_2} \longrightarrow^{i*} v^i_{a_2}$ and $v^i_{a_2} \simeq v^i_{b_2}$.

□

Theorem 334 (Kleene Equality of Evaluators)**.** *For any $t \in \text{PRGM}_{\text{MetaML}}$, $eval_{\text{MetaML:SubSOS}}(t)$ is Kleene equal to $eval_{\text{MetaML:ExpSOS}}(t)$.*

Proof. We first show that if $eval_{\text{MetaML:SubSOS}}(t) = a$ where $a \in \text{ANS}_{\text{MetaML}}$, then $eval_{\text{MetaML:ExpSOS}}(t) = a$.

Case 1. If $eval_{\text{MetaML:SubSOS}}(t) = \texttt{function}$, then $t \longrightarrow^{0*} \lambda x.t'^0$ in (Substitutional) MetaML. Observe that $t \simeq t$. By Theorem 327, $t \longrightarrow^{0*} v$ in Explicit MetaML and $\lambda x.t'^0 \simeq v$. We proceed by cases on $\lambda x.t'^0 \simeq v$. The only case is (lamu-sim), thus $v = \underline{\lambda} x.t''^0$ and $t'^0 \simeq t''^0$. We have $eval_{\text{MetaML:ExpSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:SubSOS}}(t) = \texttt{code}$, then $t \longrightarrow^{0*} \langle v'^{1} \rangle$ in (Substitutional) MetaML. Observe that $t \simeq t$. By Theorem 327, $t \longrightarrow^{0*} v$ in Explicit MetaML and $\langle v'^{1} \rangle \simeq v$. We proceed by cases on $\langle v'^{1} \rangle \simeq v$. The only case is (splice-sim), thus $v = \langle v''^{1} \rangle$ and $v'^{1} \simeq v''^{1}$. We have $eval_{\text{MetaML:ExpSOS}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:SubSOS}}(t) = n$, then $t \longrightarrow^{0*} n$ in (Substitutional) MetaML. Observe that $t \simeq t$. By Theorem 327, $t \longrightarrow^{0*} v$ in Explicit MetaML and $n \simeq v$. We proceed by cases on $n \simeq v$. The only case is (num-sim), thus $v = n$. We have $eval_{\text{MetaML:ExpSOS}}(t) = n$.

We then show that if $eval_{\text{MetaML:ExpSOS}}(t) = a$ where $a \in \text{ANS}_{\text{MetaML}}$, then $eval_{\text{MetaML:SubSOS}}(t) = a$.

Case 1. If $eval_{\text{MetaML:ExpSOS}}(t) = \texttt{function}$, then $t \longrightarrow^{0*} \underline{\lambda}x.t'^{0}$ in Explicit MetaML. Observe that $t \simeq t$. By Theorem 328, $t \longrightarrow^{0*} v$ in (Substitutional) MetaML and $v \simeq \underline{\lambda}x.t'^{0}$. We proceed by cases on $v \simeq \underline{\lambda}x.t'^{0}$. The only case is (lamu-sim), thus $v = \lambda x.t''^{0}$ and $t''^{0} \simeq t'^{0}$. We have $eval_{\text{MetaML:SubSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:ExpSOS}}(t) = \texttt{code}$, then $t \longrightarrow^{0*} \langle v'^{1} \rangle$ in Explicit MetaML. Observe that $t \simeq t$. By Theorem 328, $t \longrightarrow^{0*} v$ in (Substitutional) MetaML and $v \simeq \langle v'^{1} \rangle$. We proceed by cases on $v \simeq \langle v'^{1} \rangle$. The only case is (splice-sim), thus $v = \langle v''^{1} \rangle$ and $v''^{1} \simeq v''$. We have $eval_{\text{MetaML:SubSOS}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:ExpSOS}}(t) = n$, then $t \longrightarrow^{0*} n$ in Explicit MetaML. Observe that $t \simeq t$. By Theorem 328, $t \longrightarrow^{0*} v$ in (Substitutional) MetaML and $v \simeq n$. We proceed by cases on $v \simeq n$. The only case is (num-sim), thus $v = n$. We have $eval_{\text{MetaML:SubSOS}}(t) = n$.

We observe that $eval_{\text{MetaML:SubSOS}}(t)$ is undefined if and only if $eval_{\text{MetaML:ExpSOS}}(t)$ is undefined. Therefore, $eval_{\text{MetaML:SubSOS}}(t)$ is Kleene equal to $eval_{\text{MetaML:ExpSOS}}(t)$. □

D.2 Equivalence of MetaML and Suspended MetaML

We demonstrate the equivalence of the substitutional structural operational semantics of MetaML and the structural operational semantics of Suspended MetaML. We use subscripts "$_{\text{sub}}$" and "$_{\text{sus}}$" to differentiate the syntax of (Substitutional) MetaML from the syntax of Suspended MetaML.

D.2.1 Well-boundness Judgement

Definition 335 (Well-boundness Judgement)**.** Let the well-boundness judgement $\vdash\ wb$ be a ternary relation on the power set of variables, the power set of variables and the set of runtime terms.

$$\vdash wb \subseteq \mathscr{P}(\text{VAR}) \times \mathscr{P}(\text{VAR}) \times \text{RTERM}$$

$$\frac{}{\mathscr{U};\mathscr{V} \vdash x\ wb} \text{ where } x \in \mathscr{U}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t_1\ wb \quad \mathscr{U};\mathscr{V} \vdash t_2\ wb}{\mathscr{U};\mathscr{V} \vdash t_1\ t_2\ wb}$$

$$\frac{\mathscr{U} \cup \{x\};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash \lambda x.t\ wb} \text{ where } x \notin \mathscr{V}$$

$$\frac{\mathscr{U} \cup \{x\};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash \underline{\lambda} x.t\ wb} \text{ where } x \notin \mathscr{V}$$

$$\frac{\mathscr{U} \cup \{x\};\mathscr{V} \cup \{x\} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash \hat{\lambda} x.t\ wb} \text{ where } x \notin \mathscr{V}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash \langle t \rangle\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash \sim t\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash !t\ wb}$$

$$\frac{}{\mathscr{U};\mathscr{V} \vdash n\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t_1\ wb \quad \mathscr{U};\mathscr{V} \vdash t_2\ wb}{\mathscr{U};\mathscr{V} \vdash t_1 + t_2\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash w\ wb \quad \mathscr{U} \cup \{x\};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash t[x := w]\ wb} \text{ where } x \notin \mathscr{V}$$

Lemma 336. *If $\mathscr{U};\mathscr{V} \vdash t\ wb$, then $FV(t) \subseteq \mathscr{U}$.*

Proof. We proceed by structural induction on $\mathscr{U};\mathscr{V} \vdash t\ wb$.

Case 1. $\dfrac{}{\mathscr{U};\mathscr{V} \vdash x\ wb}$ where $x \in \mathscr{U}$.

We immediately have $FV(x) = \{x\} \subseteq \mathscr{U}$ since $x \in \mathscr{U}$.

Case 2. $\dfrac{\mathscr{U};\mathscr{V} \vdash t_1\ wb \quad \mathscr{U};\mathscr{V} \vdash t_2\ wb}{\mathscr{U};\mathscr{V} \vdash t_1\ t_2\ wb}$.

By the induction hypothesis, $FV(t_1) \subseteq \mathscr{U}$ and $FV(t_2) \subseteq \mathscr{U}$. Then, $FV(t_1\,t_2) \subseteq FV(t_1) \cup FV(t_2) \subseteq \mathscr{U}$.

Case 3. $\dfrac{\mathscr{U} \cup \{x\};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash \lambda x.t\ wb}$ where $x \notin \mathscr{V}$.

By the induction hypothesis, $FV(t) \subseteq \mathscr{U} \cup \{x\}$. Then, $FV(\lambda x.t) = FV(t) \setminus \{x\} \subseteq (\mathscr{U} \cup \{x\}) \setminus \{x\} = \mathscr{U} \setminus \{x\} \subseteq \mathscr{U}$.

Case 4. $\dfrac{\mathscr{U} \cup \{x\}; \mathscr{V} \vdash t\ wb}{\mathscr{U}; \mathscr{V} \vdash \underline{\lambda} x.t\ wb}$ where $x \notin \mathscr{V}$.

This is analogous to the case of $\mathscr{U}; \mathscr{V} \vdash \lambda x.t\ wb$.

Case 5. $\dfrac{\mathscr{U} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash t\ wb}{\mathscr{U}; \mathscr{V} \vdash \hat{\lambda} x.t\ wb}$ where $x \notin \mathscr{V}$.

By the induction hypothesis, $FV(t) \subseteq \mathscr{U} \cup \{x\}$. Then, $FV(\hat{\lambda} x.t) = FV(t) \backslash \{x\} \subseteq (\mathscr{U} \cup \{x\}) \backslash \{x\} = \mathscr{U} \backslash \{x\} \subseteq \mathscr{U}$.

Case 6. $\dfrac{\mathscr{U}; \mathscr{V} \vdash t\ wb}{\mathscr{U}; \mathscr{V} \vdash \langle t \rangle\ wb}$.

By the induction hypothesis, $FV(t) \subseteq \mathscr{U}$. Then, $FV(\langle t \rangle) = FV(t) \subseteq \mathscr{U}$.

Case 7. $\dfrac{\mathscr{U}; \mathscr{V} \vdash t\ wb}{\mathscr{U}; \mathscr{V} \vdash \sim t\ wb}$.

This is analogous to the case of $\mathscr{U}; \mathscr{V} \vdash \sim t\ wb$.

Case 8. $\dfrac{\mathscr{U}; \mathscr{V} \vdash t\ wb}{\mathscr{U}; \mathscr{V} \vdash !t\ wb}$.

This is analogous to the case of $\mathscr{U}; \mathscr{V} \vdash \sim t\ wb$.

Case 9. $\dfrac{}{\mathscr{U}; \mathscr{V} \vdash n\ wb}$.

We immediately have $FV(n) = \emptyset \subseteq \mathscr{U}$.

Case 10. $\dfrac{\mathscr{U}; \mathscr{V} \vdash t_1\ wb \quad \mathscr{U}; \mathscr{V} \vdash t_2\ wb}{\mathscr{U}; \mathscr{V} \vdash t_1 + t_2\ wb}$.

This is analogous to the case of $\mathscr{U}; \mathscr{V} \vdash t_1\ t_2\ wb$.

Case 11. $\dfrac{\mathscr{U}; \mathscr{V} \vdash w\ wb \quad \mathscr{U} \cup \{x\}; \mathscr{V} \vdash t\ wb}{\mathscr{U}; \mathscr{V} \vdash t[x := w]\ wb}$ where $x \notin \mathscr{V}$.

By the induction hypothesis, $FV(w) \subseteq \mathscr{U}$ and $FV(t) \subseteq \mathscr{U} \cup \{x\}$. Then, $FV(t[x := w]) = FV(w) \cup (FV(t) \backslash \{x\}) \subseteq \mathscr{U} \cup ((\mathscr{U} \cup \{x\}) \backslash \{x\}) = \mathscr{U}$.

□

Lemma 337. *If $\mathscr{U}; \mathscr{V} \vdash t\ wb$, then $\mathscr{U} \cup \mathscr{W}; \mathscr{V} \vdash t\ wb$.*

Proof. By induction on the structure of $\mathscr{U}; \mathscr{V} \vdash t\ wb$. □

Lemma 338. *If $\mathscr{U}; \mathscr{V} \vdash t\ wb$ and $x_N \notin \mathscr{U} \cup \mathscr{V} \cup Var(t)$, then $\mathscr{U} \cup \{x_N\}; \mathscr{V} \cup \{x_N\} \vdash t\ wb$.*

Proof. By induction on the structure of $\mathscr{U}; \mathscr{V} \vdash t\ wb$. □

Lemma 339. *If $\mathscr{U} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash v^{i+1}\ wb$, then $\mathscr{U} \cup \{x\}; \mathscr{V} \backslash \{x\} \vdash v^{i+1}\ wb$.*

Proof. By induction on the structure of $\mathscr{U} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash v^{i+1}\ wb$. □

Lemma 340. *If $\mathscr{U}; \mathscr{V} \vdash t_0^i\ wb$, $\textsc{Var}(t_1^i) \subseteq \mathscr{X}$, $\mathscr{V} \subseteq \mathscr{U} \subseteq \mathscr{X}$ and $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t_0^i \longrightarrow^{i*} t_0'^i$, then $\mathscr{U}; \mathscr{V} \vdash t_0'^i\ wb$.*

Proof. We proceed by induction on the structure of $\mathscr{U};\mathscr{V} \vdash t_0^i\ wb$ and the structure of $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_0^i \longrightarrow^{i*} t_0'^i$.

Case 1. $\dfrac{}{\mathscr{U};\mathscr{V} \vdash x\ wb}$ where $x \in \mathscr{U}$.

This case is vacuous because $\mathscr{U};\mathscr{V};\mathscr{X} \vdash x \not\longrightarrow$.

Case 2. $\dfrac{\mathscr{U};\mathscr{V} \vdash t_1\ wb \quad \mathscr{U};\mathscr{V} \vdash t_2\ wb}{\mathscr{U};\mathscr{V} \vdash t_1\ t_2\ wb}$.

In this case, $t_0 = t_1\ t_2$.

We proceed by cases on $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_0^i \longrightarrow^{i*} t_0'^i$.

Case i. $\dfrac{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1 \longrightarrow^i t_1'}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1 t_2 \longrightarrow^i t_1' t_2}$.

By induction hypothesis, $\mathscr{U};\mathscr{V} \vdash t_1'\ wb$. Then $\mathscr{U};\mathscr{V} \vdash t_1' t_2\ wb$.

Case ii. $\dfrac{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_2 \longrightarrow^i t_2'}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash v_1 t_2 \longrightarrow^i v_1 t_2'}$.

This case is analogous to the previous case.

Case iii. $\dfrac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash (\underline{\lambda} x_0.t^0)[x_1 := w_1]...[x_n := w_n]\ v^0 \longrightarrow^0 t^0[x_0 := v^0][x_1 := w_1]...[x_n := w_n]}$.

Then $t_1 = (\lambda x_0.t^0)[x_1 := w_1]...[x_n := w_n]$ and $t_2 = v^0$.

Given $\mathscr{U};\mathscr{V} \vdash (\underline{\lambda} x_0.t^0)[x_1 := w_1]...[x_n := w_n]\ wb$, the following holds:

(1) $\mathscr{U};\mathscr{V} \vdash w_n\ wb$,

(2) $\mathscr{U} \cup \{x_n\};\mathscr{V} \vdash w_{n-1}\ wb$,

...

(n) $\mathscr{U} \cup \{x_n, x_{n-1}, ..., x_2\};\mathscr{V} \vdash w_1\ wb$,

(n+1) $\mathscr{U} \cup \{x_n, x_{n-1}, ..., x_2, x_1, x_0\};\mathscr{V} \vdash t^0\ wb$,

(n+2) $\{x_1, x_2, ..., x_n\} \cap \mathscr{V} = \emptyset$, and

(n+3) $x_0 \notin \mathscr{V}$.

We also have

(n+4) $\mathscr{U};\mathscr{V} \vdash v^0\ wb$.

To show $\mathscr{U};\mathscr{V} \vdash t^0[x_0 := v^0][x_1 := w_1]...[x_n := w_n]\ wb$, it is sufficient to show that

(1) $\mathscr{U};\mathscr{V} \vdash w_n\ wb$,

(2) $\mathscr{U} \cup \{x_n\};\mathscr{V} \vdash w_{n-1}\ wb$,

...

(n) $\mathscr{U} \cup \{x_n, x_{n-1}, ..., x_2\};\mathscr{V} \vdash w_1\ wb$,

(n+1) $\mathscr{U} \cup \{x_n, x_{n-1}, ..., x_2, x_1\};\mathscr{V} \vdash v^0\ wb$,

(n+2) $\mathscr{U} \cup \{x_n, x_{n-1}, ..., x_2, x_1, x_0\};\mathscr{V} \vdash t^0\ wb$, and

(n+3) $\{x_0, x_1, x_2, ..., x_n\} \cap \mathscr{V} = \emptyset$.

All of (1)-(n+3) above hold.

Case 3. $\dfrac{\mathscr{U} \cup \{x\}; \mathscr{V} \vdash t\ wb}{\mathscr{U}; \mathscr{V} \vdash \lambda x.t\ wb}$ where $x \notin \mathscr{V}$.

In this case, $t_0 = \lambda x.t$. We proceed by cases on $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t_0 \longrightarrow t_0'$.

Case i. $\dfrac{}{\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \lambda x.t \longrightarrow^0 \underline{\lambda} x.t}$.

Given $\mathscr{U}; \mathscr{V} \vdash \lambda x.t\ wb$, we immediately have $\mathscr{U}; \mathscr{V} \vdash \underline{\lambda} x.t\ wb$.

Case ii. $\dfrac{}{\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \lambda x.t \longrightarrow^{i+1} \hat{\lambda} x_N.t[x := x_N]}$ where $t^{i+1} \notin \text{VALUE}^{i+1}$ and $x_N \notin \mathscr{X}$

To show $\mathscr{U}; \mathscr{V} \vdash \hat{\lambda} x_N.t[x := x_N]\ wb$, it is sufficient to show that

- $x_N \notin \mathscr{V}$, and
- $\mathscr{U} \cup \{x_N\}; \mathscr{V} \cup \{x_N\} \vdash t[x := x_N]\ wb$, which can be shown by
  - $x \notin \mathscr{V} \cup \{x_N\}$,
  - $\mathscr{U} \cup \{x_N\}; \mathscr{V} \cup \{x_N\} \vdash x_N\ wb$, and
  - $\mathscr{U} \cup \{x_N, x\}; \mathscr{V} \cup \{x_N\} \vdash t\ wb$.

We show the above all hold as follows.

- Given $x_N \notin \mathscr{X}$ and $\mathscr{V} \subseteq \mathscr{U}$, we get $x_N \notin \mathscr{V}$.
- Given $x \notin \mathscr{V}$ and $x \not\equiv x_N$, we get $x \notin \mathscr{V} \cup \{x_N\}$.
- Given $x_N \in \mathscr{U} \cup \{x_N\}$, we get $\mathscr{U} \cup \{x_N\}; \mathscr{V} \cup \{x_N\} \vdash x_N\ wb$.
- Given $\mathscr{U} \cup \{x\}; \mathscr{V} \vdash t\ wb$ and $x_N \notin (\mathscr{U} \cup \{x\}) \cup \mathscr{Y} \cup Var(t)$, by Lemma 338, we get $\mathscr{U} \cup \{x\} \cup \{x_N\}; \mathscr{V} \cup \{x_N\} \vdash t\ wb$.

Case 4. $\dfrac{\mathscr{U} \cup \{x\}; \mathscr{V} \vdash t\ wb}{\mathscr{U}; \mathscr{V} \vdash \underline{\lambda} x.t\ wb}$ where $x \notin \mathscr{V}$.

This case is vacuous because $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \underline{\lambda} x.t \not\longrightarrow$.

Case 5. $\dfrac{\mathscr{U} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash t\ wb}{\mathscr{U}; \mathscr{V} \vdash \hat{\lambda} x.t\ wb}$ where $x \notin \mathscr{V}$.

In this case, $t_0 = \hat{\lambda} x.t$. We proceed by cases on $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t_0 \longrightarrow t_0'$.

Case i. $\dfrac{\mathscr{U} \cup \{x\}; \mathscr{V} \cup \{x\}; \mathscr{X} \vdash t \longrightarrow^{i+1} t'}{\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \hat{\lambda} x.t \longrightarrow^{i+1} \hat{\lambda} x.t'}$

To show $\mathscr{U}; \mathscr{V} \vdash \hat{\lambda} x.t'\ wb$, it is sufficient to show that

- $x \notin \mathscr{V}$, and
- $\mathscr{U} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash t'\ wb$.

Given $\mathscr{U}; \mathscr{V} \vdash \hat{\lambda} x.t\ wb$, we have $\mathscr{U} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash t\ wb$ and $x \notin \mathscr{V}$. By the induction hypothesis, we have $\mathscr{U} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash t'\ wb$.

Case ii. $\dfrac{}{\mathscr{U} \mid \hat{\lambda} x.v^{i+1} \longrightarrow^{i+1} \mathscr{U} \mid \lambda x.v^{i+1}}$

In this case, $t = v^{i+1}$.

To show $\mathscr{U}; \mathscr{V} \vdash \lambda x.v^{i+1}\ wb$, it is sufficient to show that

- $x \notin \mathscr{V}$, and
- $\mathscr{U} \cup \{x\}; \mathscr{V} \vdash v^{i+1}\ wb$.

Given $\mathscr{U}; \mathscr{V} \vdash \hat{\lambda} x.v\ wb$, we have $x \notin \mathscr{V}$ and $\mathscr{U} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash v^{i+1}\ wb$. By Lemma 339, we get $\mathscr{U} \cup \{x\}; \mathscr{V} \backslash \{x\} \vdash v^{i+1}\ wb$. Since $x \notin \mathscr{V}$, we get $\mathscr{U} \cup \{x\}; \mathscr{V} \vdash v^{i+1}\ wb$.

Case 6. $\dfrac{\mathscr{U}; \mathscr{V} \vdash t\ wb}{\mathscr{U}; \mathscr{V} \vdash \langle t \rangle\ wb}$.

In this case, $t_0 = \langle t \rangle$. We proceed by cases on $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t_0 \longrightarrow t_0'$.

Case i. $\dfrac{\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t \longrightarrow^{i+1} t'}{\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \langle t \rangle \longrightarrow^{i} \langle t' \rangle}$

By induction hypothesis, $\mathscr{U}; \mathscr{V} \vdash t'\ wb$. Then $\mathscr{U}; \mathscr{V} \vdash \langle t' \rangle\ wb$.

Case 7. $\dfrac{\mathscr{U}; \mathscr{V} \vdash t\ wb}{\mathscr{U}; \mathscr{V} \vdash {\sim} t\ wb}$.

This case is analogous to the previous case.

Case 8. $\dfrac{\mathscr{U}; \mathscr{V} \vdash t\ wb}{\mathscr{U}; \mathscr{V} \vdash !t\ wb}$.

This case is analogous to the previous case.

Case 9. $\dfrac{\mathscr{U}; \mathscr{V} \vdash t_1\ wb \quad \mathscr{U}; \mathscr{V} \vdash t_2\ wb}{\mathscr{U}; \mathscr{V} \vdash t_1 + t_2\ wb}$.

This case is analogous to the $\mathscr{U}; \mathscr{V} \vdash t_1\ t_2\ wb$ case.

Case 10. $\overline{\mathscr{U}; \mathscr{V} \vdash n\ wb}$.

This case is vacuous because $n \not\longrightarrow$.

Case 11. $\dfrac{\mathscr{U}; \mathscr{V} \vdash w\ wb \quad \mathscr{U} \cup \{x\}; \mathscr{V} \vdash t\ wb}{\mathscr{U}; \mathscr{V} \vdash t[x := w]\ wb}$ where $x \notin \mathscr{V}$.

In this case, $t_0 = t[x := w]$.

We proceed by cases on $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t_0 \longrightarrow t_0'$.

Case i. $\overline{\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash (\lambda x_1.t_1^0)\overline{[x_j := w_j]}[x := w] \longrightarrow^0 (\underline{\lambda} x_1.t_1^0)\overline{[x_j := w_j]}[x := w]}$.

In this case, $t = (\lambda x_1.t_1^0)\overline{[x_j := w_j]}$.

Given $\mathscr{U}; \mathscr{V} \vdash (\lambda x_1.t_1^0)\overline{[x_j := w_j]}[x := w]\ wb$, we immediately have $\mathscr{U}; \mathscr{V} \vdash (\underline{\lambda} x_1.t_1^0)\overline{[x_j := w_j]}[x := w]\ wb$.

Case ii. $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t[x := w] \longrightarrow^{\mathrm{x}i} t'$.

We then proceed by induction on the structure of $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t[x := w] \longrightarrow^{\mathrm{x}i} t'$.

Case a. $\overline{\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash n[x := w] \longrightarrow^{\mathrm{x}i} n}$.

In this case, $t = n$.

We immediately have $\mathscr{U}; \mathscr{V} \vdash n\ wb$.

Case b. $\overline{\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash x[x := w] \longrightarrow^{\mathrm{x}i} w}$

In this case, $t = x$. We have $\mathscr{U}; \mathscr{V} \vdash w\ wb$.

Case c. $\dfrac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash x_1[x := w] \longrightarrow^{\mathrm{x}i} x_1}$ where $x_1 \not\equiv x$

In this case, $t = x_1$.

Since $\mathscr{U} \cup \{x\};\mathscr{V} \vdash x_1$, we know $x_1 \in \mathscr{U} \cup \{x\}$. As $x_1 \not\equiv x$, $x_1 \in \mathscr{U}$. Then $\mathscr{U};\mathscr{V} \vdash x_1\ wb$.

Case d. $\dfrac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash (t_1\ t_2)[x := w] \longrightarrow^{\mathrm{x}i} (t_1[x := w])\ (t_2[x := w])}$

In this case, $t = t_1\ t_2$.

To show $\mathscr{U};\mathscr{V} \vdash (t_1[x := w])\ (t_2[x := w])\ wb$, it is sufficient to show $\mathscr{U};\mathscr{V} \vdash t_i[x := w]\ wb$ for $i = 1,2$, which can be shown by

- $\mathscr{U};\mathscr{V} \vdash w\ wb$,
- $\mathscr{U} \cup \{x\};\mathscr{V} \vdash t_i\ wb$ for $i = 1,2$, and
- $x \notin \mathscr{V}$.

All the above hold.

Case e. $\dfrac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash (t_1 + t_2)[x := w] \longrightarrow^{\mathrm{x}i} (t_1[x := w]) + (t_2[x := w])}$

This is analogous to the previous case.

Case f. $\dfrac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash (\lambda x_1.t_1)[x := w] \longrightarrow^{\mathrm{x}(i+1)} \lambda x_N.t_1[x_1 := x_N][x := w]}$ where $x_N \notin \mathscr{X}$

In this case, $t = \lambda x_1.t_1$.

Given $\mathscr{U} \cup \{x\};\mathscr{V} \vdash \lambda x_1.t_1\ wb$, we have $\mathscr{U} \cup \{x,x_1\};\mathscr{V} \vdash t_1\ wb$ and $x_1 \notin \mathscr{V}$.

To show $\mathscr{U};\mathscr{V} \vdash \lambda x_N.t_1[x_1 := x_N][x := w]\ wb$, it is sufficient to show

- $x_N \notin \mathscr{V}$, and
- $\mathscr{U} \cup \{x_N\};\mathscr{V} \vdash t_1[x_1 := x_N][x := w]\ wb$, the latter of which can be shown by
  - $x \notin \mathscr{V}$,
  - $\mathscr{U} \cup \{x_N\};\mathscr{V} \vdash w\ wb$, and
  - $\mathscr{U} \cup \{x_N,x\};\mathscr{V} \vdash t_1[x_1 := x_N]\ wb$, the last of which can be shown by
    - $x_1 \notin \mathscr{V}$,
    - $\mathscr{U} \cup \{x_N,x\};\mathscr{V} \vdash x_N\ wb$,
    - $\mathscr{U} \cup \{x_N,x,x_1\};\mathscr{V} \vdash t_1\ wb$.

We show the above all hold as follows.

- We already know $x \notin \mathscr{V}$ and $x_1 \notin \mathscr{V}$.
- Given $x_N \notin \mathscr{X}$ and $\mathscr{V} \subseteq \mathscr{X}$, we get $x_N \notin \mathscr{V}$.
- Since $x_N \in \mathscr{X} \cup \{x_N,x\}$, we have $\mathscr{U} \cup \{x_N,x\};\mathscr{V} \vdash x_N\ wb$.
- Given $\mathscr{U};\mathscr{V} \vdash w\ wb$, by Lemma 337, $\mathscr{U} \cup \{x_N\};\mathscr{V} \vdash w\ wb$. $\mathscr{U} \cup \{x_N,x,x_1\};\mathscr{V} \vdash t_1\ wb$ holds analogously.

Case g. $\dfrac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash \langle t_1 \rangle[x := w] \longrightarrow^{\mathrm{x}i} \langle t_1[x := w] \rangle}$

In this case, $t = \langle t_1 \rangle$.

Given $\mathscr{U} \cup \{x\};\mathscr{V} \vdash \langle t_1 \rangle\ wb$, we have $\mathscr{U} \cup \{x\};\mathscr{V} \vdash t_1\ wb$.

To show $\mathscr{U};\mathscr{V} \vdash \langle t_1[x := w]\rangle$ *wb*, it is sufficient to show $\mathscr{U};\mathscr{V} \vdash t_1[x := w]$ *wb*, which can be shown by

- $x \notin \mathscr{V}$,
- $\mathscr{U};\mathscr{V} \vdash w$ *wb*, and
- $\mathscr{U} \cup \{x\};\mathscr{V} \vdash t_1$ *wb*.

All the above hold.

Case h. $\dfrac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash (!t_1)[x := w] \longrightarrow^{\mathrm{x}i} !t_1[x := w]}$

This case is analogous to the previous case.

Case i. $\dfrac{}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash (\sim t_1)[x := w] \longrightarrow^{\mathrm{x}i} \sim t_1[x := w]}$

This case is analogous to the previous case.

Case j. $\dfrac{\mathscr{U} \cup \{x\};\mathscr{V};\mathscr{X} \vdash t_1[x_1 := w_1] \longrightarrow^{\mathrm{x}i} t_2}{\mathscr{U};\mathscr{V};\mathscr{X} \vdash t_1[x_1 := w_1][x := w] \longrightarrow^{\mathrm{x}i} t_2[x := w]}$

In this case, $t = t_1[x_1 := w]$.

Given $\mathscr{U} \cup \{x\};\mathscr{V} \vdash t_1[x_1 := w]$ *wb*, by induction hypothesis, we get $\mathscr{U} \cup \{x\};\mathscr{V} \vdash t_2$ *wb*. Together with $x \notin \mathscr{V}$ and $\mathscr{U};\mathscr{V} \vdash w$ *wb*, we get $\mathscr{U};\mathscr{V} \vdash t_2[x := w]$ *wb*.

□

D.2.2 Unload Function

Definition 341 (Unload Function)**.** Let $i \in \mathbb{N}$. Define the unload function U to be a total function from the set of Suspended MetaML runtime terms $\mathrm{RTERM}^i_{\mathrm{sus}}$ to the set of Substitutional MetaML terms $\mathrm{TERM}^i_{\mathrm{sub}}$.

$$U \;:\; \mathscr{P}(\mathrm{VAR}) \times \mathrm{RTERM}^i_{\mathrm{sus}} \to \mathrm{TERM}^i_{\mathrm{sub}}$$

$$\begin{array}{rcl}
U(\mathscr{Z} \mid x) & = & x \\
U(\mathscr{Z} \mid t_1\, t_2) & = & U(\mathscr{Z} \mid t_1)\, U(\mathscr{Z} \mid t_2) \\
U(\mathscr{Z} \mid \lambda x.t^0) & = & \lambda x.U(\mathscr{Z} \mid t^0) \\
U(\mathscr{Z} \mid \lambda x.t^{i+1}) & = & \lambda x_N.U(\mathscr{Z} \cup \{x_N\} \mid t^{i+1}[x := x_N]) \\
 & & \text{where } t^{i+1} \notin \mathrm{VALUE}^{i+1}_{\mathrm{sus}}\ x_N \notin \mathscr{Z} \\
U(\mathscr{Z} \mid \lambda x.v^{i+1}) & = & \lambda x.U(\mathscr{Z} \mid v^{i+1}) \\
U(\mathscr{Z} \mid \underline{\lambda} x.t) & = & \lambda x.U(\mathscr{Z} \mid t) \\
U(\mathscr{Z} \mid \hat{\lambda} x.t) & = & \lambda x.U(\mathscr{Z} \mid t) \\
U(\mathscr{Z} \mid \langle t\rangle) & = & \langle U(\mathscr{Z} \mid t)\rangle \\
U(\mathscr{Z} \mid \sim t) & = & \sim U(\mathscr{Z} \mid t) \\
U(\mathscr{Z} \mid !t) & = & !U(\mathscr{Z} \mid t) \\
U(\mathscr{Z} \mid n) & = & n \\
U(\mathscr{Z} \mid t_1 + t_2) & = & U(\mathscr{Z} \mid t_1) + U(\mathscr{Z} \mid t_2) \\
U(\mathscr{Z} \mid t[x_0 := w_0]\overline{[x_i := w_i]}) & = & U(\mathscr{Z} \mid t[w_0/x_0]\overline{[x_i := w_i]})
\end{array}$$

Remark 342. $U(\mathscr{Z} \mid t)$ may be omitted to $U(t)$ if $\mathscr{Z}$ is clear by the context.

Proposition 343. $U(\mathscr{Z} \mid t^i_{\mathrm{s}}) = t^i_{\mathrm{s}}$.

Proposition 344. $U(\mathscr{Z} \mid t\overline{[w_i/x_i]}) = U(\mathscr{Z} \mid t)\overline{[U(\mathscr{Z} \mid w_i)/x_i]}$.

D.2.3 Bisimulation Relation

Definition 345 (Bisimulation Relation)**.** Define the bisimulation relation $\simeq$ to be a binary relation between the set of terms in (Substitutional) MetaML and the set of runtime terms in Suspended MetaML.

$$\simeq \subseteq \textsc{Term}^i_{\mathrm{sub}} \times \textsc{RTerm}^i_{\mathrm{sus}}$$

$$t^i_1 \simeq t^i_2 \text{ if and only if } t^i_1 = U(\mathscr{Z} \mid t^i_2) \text{ where } \textsc{Var}(t^i_2) \subseteq \mathscr{Z}$$

D.2.4 Explicit Substitution Descendant Relation

Definition 346 (Explicit Substitution Descendant Relation)**.** For any $t_1, t_2 \in \textsc{RTerm}_{\mathrm{sus}}$, $t_1 \prec^{\mathrm{x}} t_2$ if and only if $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t_2 \longrightarrow^{\mathrm{x}} t_1$. We call $\prec^{\mathrm{x}}$ the explicit substitution descendant relation.

Proposition 347 (Well-foundedness of Explicit Substitution Descendant Relation)**.** *The explicit substitution descendant relation $\prec^{\mathrm{x}}$ is well-founded.*

The proof is analogous to the proof of Lemma 322.

D.2.5 Canonisation

Lemma 348 (Canonisation of Substitutional MetaML)**.** *If $t^i_{a_1} \simeq v^i_{b_1}$, then $t^i_{a_1} \in \textsc{Value}^i_{\mathrm{sub}}$.*

Proof. We proceed by induction on the structure of $v^i_{b_1} \in \textsc{Value}^i_{\mathrm{sus}}$.

Case 1. $(v^{i+1}_{b_1} = x)$. Then $t^{i+1}_{a_1} = U(x) = x$ and $x \in \textsc{Value}^{i+1}_{\mathrm{sub}}$.

Case 2. $(v^{i+1}_{b_1} = v^{i+1}_{b_{11}}\ v^{i+1}_{b_{12}})$. Then $t^{i+1}_{a_1} = U(v^{i+1}_{b_{11}}\ v^{i+1}_{b_{12}}) = U(v^{i+1}_{b_{11}})\ U(v^{i+1}_{b_{12}})$. Let $t^{i+1}_{a_1} = t^{i+1}_{a_{11}}\ t^{i+1}_{a_{12}}$. We have $t^i_{a_{11}} \simeq v^i_{b_{11}}$ and $t^i_{a_{12}} \simeq v^i_{b_{12}}$. By the induction hypothesis, $t^{i+1}_{a_{11}} \in \textsc{Value}^{i+1}_{\mathrm{sub}}$ and $t^{i+1}_{a_{12}} \in \textsc{Value}^{i+1}_{\mathrm{sub}}$. Then $t^{i+1}_{a_{11}}\ t^{i+1}_{a_{12}} \in \textsc{Value}^{i+1}_{\mathrm{sub}}$.

Case 3. $(v^{i+1}_{b_1} = \lambda x.v^{i+1}_{b_{11}})$. Then $t^{i+1}_{a_1} = U(\lambda x.v^{i+1}_{b_{11}}) = \lambda x.U(v^{i+1}_{b_{11}})$. Let $t^{i+1}_{a_1} = \lambda x.t^{i+1}_{a_{11}}$. We have $t^i_{a_{11}} \simeq v^i_{b_{11}}$. By the induction hypothesis, $t^{i+1}_{a_{11}} \in \textsc{Value}^{i+1}_{\mathrm{sub}}$. Then $\lambda x.t^{i+1}_{a_{11}} \in \textsc{Value}^{i+1}_{\mathrm{sub}}$.

Case 4. $(v^i_{b_1} = (\underline{\lambda} x.t^0_{b_{11}})\overline{[x_i := w_i]})$. Then $t^i_{a_1} = U(\mathscr{Z} \mid (\underline{\lambda} x.t^0_{b_{11}})\overline{[x_i := w_i]}) = U(\mathscr{Z} \mid (\underline{\lambda} x.t^0_{b_{11}})\overline{[w_i/x_i]})$. Let $x_N \notin \mathscr{Z}$. We have $t^i_{a_1} = U((\underline{\lambda} x.t^0_{b_{11}})\overline{[w_i/x_i]}) = U(\underline{\lambda} x_N.t^0_{b_{11}}[x_N/x]\overline{[w_i/x_i]}) = \lambda x_N.U(t^0_{b_{11}}[x_N/x]\overline{[w_i/x_i]}) \in \textsc{Value}^0_{\mathrm{sub}} \subseteq \textsc{Value}^i_{\mathrm{sub}}$.

Case 5. $(v^i_{b_1} = \langle v^{i+1}_{b_{11}} \rangle)$. Then $t^i_{a_1} = U(\langle v^{i+1}_{b_{11}} \rangle) = \langle U(v^{i+1}_{b_{11}}) \rangle$. Let $t^i_{a_1} = \langle t^{i+1}_{a_{11}} \rangle$. We have $t^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{11}}$. By the induction hypothesis, $t^{i+1}_{a_{11}} \in \textsc{Value}^{i+1}_{\mathrm{sub}}$. Then $\langle t^{i+1}_{a_{11}} \rangle \in \textsc{Value}^i_{\mathrm{sub}}$.

Case 6. $(v^{i+2}_{b_1} = {\sim} v^{i+1}_{b_{11}})$. This case is analogous to the $(v^i_{b_1} = \langle v^{i+1}_{b_{11}} \rangle)$ case.

Case 7. $(v^{i+1}_{b_1} = !v^{i+1}_{b_{11}})$. This case is analogous to the $(v^i_{b_1} = \langle v^{i+1}_{b_{11}} \rangle)$ case.

Case 8. $(v^i_{b_1} = n)$. Then $t^i_{a_1} = U(n) = n$ and $n \in \text{VALUE}^i_{\text{sub}}$.

Case 9. $(v^{i+1}_{b_1} = v^{i+1}_{b_{11}} + v^{i+1}_{b_{12}})$. This case is analogous to the $(v^{i+1}_{b_1} = v^{i+1}_{b_{11}}\ v^{i+1}_{b_{12}})$ case.

□

Lemma 349 (Canonisation of Suspended MetaML)**.** *If $v^i_{a_1} \simeq t^i_{b_1}$, $\mathscr{U};\mathscr{V} \vdash t^i_{b_1}$ wb, $\text{VAR}(t^i_{b_1}) \subseteq \mathscr{X}$ and $\mathscr{V} \subseteq \mathscr{U} \subseteq \mathscr{X}$, and either $t^i_{b_1}$ is in substitution normal form and $FV(t^i_{b_1}) \subseteq \mathscr{V} = \mathscr{U}$, or $t^i_{b_1}$ is not in substitution normal form and $FV(t^i_{b_1}) \subseteq \mathscr{U}$, then $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^i_{b_1} \longrightarrow^{i*} v^i_{b_2}$ and $v^i_{a_1} \simeq v^i_{b_2}$.*

Proof. We proceed by simultaneous induction on the structure of $v^i_{a_1} \in \text{VALUE}^i_{\text{sub}}$ and on the explicit substitution descendant relation $\prec^{\text{x}} t^i_{b_1}$.

Case 1. $(v^{i+1}_{a_1} = x)$. We proceed by cases on $t^{i+1}_{b_1} \in \text{RTERM}^{i+1}_{\text{sus}}$.

Case i. $(t^{i+1}_{b_1} = x)$. Then $x \in \text{VALUE}^{i+1}_{\text{sus}}$.

Case ii. $(t^{i+1}_{b_1} = x_0[x_1 := w_1]...[x_m := w_m])$. We proceed by cases on m.

Case a. $(m = 1)$. Given $x \simeq x_0[x_1 := w_1]$, we have:

$$\begin{array}{rrll} & x & & \\ = & U(x_0[x_1 := w_1]) & & \\ = & U(x_0[w_1/x_1]) & & \\ = & U(w_1) & & \text{where } x_0 \equiv x_1 \\ \text{or} \quad = & U(x_0) & & \text{where } x_0 \not\equiv x_1 \end{array}$$

We have $x \simeq w_1$ or $x \simeq x_0$.

Then by (var-eq-subst) or (var-df-subst) and (subst-subst), we have:

$$\begin{array}{rrll} \mathscr{U};\mathscr{V};\mathscr{X} \vdash & x_0[x_1 := w_1] & & \\ \longrightarrow^{\text{x}} & w_1 & & \text{where } x_0 \equiv x_1 \\ \text{or} \quad \longrightarrow^{\text{x}} & x_0 & & \text{where } x_0 \not\equiv x_1 \end{array}$$

Case 1. $(x_0 \equiv x_1)$. We have $x \simeq w_1$. Then $w_1 = x$. We have $x \in \text{VALUE}^{i+1}_{\text{sus}}$.

Case 2. $(x_0 \not\equiv x_1)$. We have $x \simeq x_0$. Then $x_0 = x$. We have $x \in \text{VALUE}^{i+1}_{\text{sus}}$.

Case b. $(m > 1)$. Given $x \simeq x_0[x_1 := w_1]...[x_m := w_m]$, we have:

$$\begin{array}{rrll} & x & & \\ = & U(x_0[x_1 := w_1][x_2 := w_2]...[x_m := w_m]) & & \\ = & U(x_0[w_1/x_1][x_2 := w_2]...[x_m := w_m]) & & \\ = & U(w_1[x_2 := w_2]...[x_m := w_m]) & & \text{where } x_0 \equiv x_1 \\ \text{or} \quad = & U(x_0[x_2 := w_2]...[x_m := w_m]) & & \text{where } x_0 \not\equiv x_1 \end{array}$$

We have $x \simeq w_1[x_2 := w_2]...[x_m := w_m]$ or $x \simeq x_0[x_2 := w_2]...[x_m := w_m]$.

Then by (var-eq-subst) or (var-df-subst) and (subst-subst), we have:

$$\begin{array}{rrll} \mathscr{U};\mathscr{V};\mathscr{X} \vdash & x_0[x_1 := w_1][x_2 := w_2]...[x_m := w_m] & & \\ \longrightarrow^{\text{x}} & w_1[x_2 := w_2]...[x_m := w_m] & & \text{where } x_0 \equiv x_1 \\ \text{or} \quad \longrightarrow^{\text{x}} & x_0[x_2 := w_2]...[x_m := w_m] & & \text{where } x_0 \not\equiv x_1 \end{array}$$

Case 1. $(x_0 \equiv x_1)$. We have $x \simeq w_1[x_2 := w_2]...[x_m := w_m]$ and $w_1[x_2 := w_2]...[x_m := w_m] \prec^{\mathrm{x}} x_0[x_1 := w_1][x_2 := w_2]...[x_m := w_m]$.
By Lemma 340, $\mathscr{U}; \mathscr{V} \vdash w_1[x_2 := w_2]...[x_m := w_m]\ wb$. By Lemma 336, $FV(w_1[x_2 := w_2]...[x_m := w_m]) \subseteq \mathscr{U}$.
By the induction hypothesis, we have $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash w_1[x_2 := w_2]...[x_m := w_m] \longrightarrow^* v_{b_2}^{i+1}$ and $x \simeq v_{b_2}^{i+1}$.

Case 2. $(x_0 \not\equiv x_1)$. It is analogous to the previous case.

Case 2. $(v_{a_1}^{i+1} = v_{a_{11}}^{i+1}\ v_{a_{12}}^{i+1})$. We proceed by cases on $t_{b_1}^{i+1} \in \mathrm{RTERM}_{\mathrm{sus}}^{i+1}$.

Case i. $(t_{b_1}^{i+1} = t_{b_{11}}^{i+1}\ t_{b_{12}}^{i+1})$. Given $v_{a_{11}}^{i+1}\ v_{a_{12}}^{i+1} \simeq t_{b_{11}}^{i+1}\ t_{b_{12}}^{i+1}$, we have:

$$\begin{array}{ll} & v_{a_{11}}^{i+1}\ v_{a_{12}}^{i+1} \\ = & U(t_{b_{11}}^{i+1}\ t_{b_{12}}^{i+1}) \\ = & U(t_{b_{11}}^{i+1})\ U(t_{b_{12}}^{i+1}) \end{array}$$

Then $v_{a_{11}}^{i+1} \simeq t_{b_{11}}^{i+1}$ and $v_{a_{12}}^{i+1} \simeq t_{b_{12}}^{i+1}$.
We have $\mathscr{U}; \mathscr{V} \vdash t_{b_{11}}^{i}\ wb$, $\mathscr{U}; \mathscr{V} \vdash t_{b_{12}}^{i}\ wb$, $FV(t_{b_{11}}^{i}) \subseteq \mathscr{V}$, $FV(t_{b_{12}}^{i}) \subseteq \mathscr{V}$ and $\mathscr{V} = \mathscr{U}$.
By the induction hypothesis, $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t_{b_{11}}^{i+1} \longrightarrow^* v_{b_{21}}^{i+1}$, $v_{a_{11}}^{i+1} \simeq v_{b_{21}}^{i+1}$, $\mathscr{U}; \mathscr{V}; \mathscr{X} \cup \mathrm{VAR}(v_{b_{21}}^{i+1}) \vdash t_{b_{12}}^{i+1} \longrightarrow v_{b_{22}}^{i+1}$ and $v_{a_{12}}^{i+1} \simeq v_{b_{22}}^{i+1}$.
Hence $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t_{b_{11}}^{i+1}\ t_{b_{12}}^{i+1} \longrightarrow^* v_{b_{21}}^{i+1}\ v_{b_{22}}^{i+1}$ and $v_{a_{11}}^{i+1}\ v_{a_{12}}^{i+1} \simeq v_{b_{21}}^{i+1}\ v_{b_{22}}^{i+1}$.

Case ii. $(t_{b_1}^{i+1} = (t_{b_{11}}^{i+1}\ t_{b_{12}}^{i+1})[x_1 := w_1]...[x_m := w_m]$ where $(t_{b_{11}}^{i+1}\ t_{b_{12}}^{i+1})[x_1 := w_1]...[x_m := w_m]$ is well formed.). Given $v_{a_{11}}^{i+1}\ v_{a_{12}}^{i+1} \simeq (t_{b_{11}}^{i+1}\ t_{b_{12}}^{i+1})[x_1 := w_1]...[x_m := w_m]$, we have:

$$\begin{array}{ll} & v_{a_{11}}^{i+1}\ v_{a_{12}}^{i+1} \\ = & U((t_{b_{11}}^{i+1}\ t_{b_{12}}^{i+1})[x_1 := w_1]...[x_m := w_m]) \\ = & U((t_{b_{11}}^{i+1}\ t_{b_{12}}^{i+1})[w_1/x_1]...[w_m/x_m]) \\ = & U((t_{b_{11}}^{i+1}[w_1/x_1]...[w_m/x_m])\ (t_{b_{12}}^{i+1}[w_1/x_1]...[w_m/x_m])) \\ = & U(t_{b_{11}}^{i+1}[w_1/x_1]...[w_m/x_m])\ U(t_{b_{12}}^{i+1}[w_1/x_1]...[w_m/x_m]) \\ = & U(t_{b_{11}}^{i+1}[x_1 := w_1]...[x_m := w_m])\ U(t_{b_{12}}^{i+1}[x_1 := w_1]...[x_m := w_m]) \end{array}$$

Then $v_{a_{11}}^{i+1} \simeq t_{b_{11}}^{i+1}[x_1 := w_1]...[x_m := w_m]$ and $v_{a_{12}}^{i+1} \simeq t_{b_{12}}^{i+1}[x_1 := w_1]...[x_m := w_m]$.
We have $\mathscr{U}; \mathscr{V} \vdash t_{b_{11}}^{i+1}[x_1 := w_1]...[x_m := w_m]\ wb$, $\mathscr{U}; \mathscr{V} \vdash t_{b_{12}}^{i}[x_1 := w_1]...[x_m := w_m]\ wb$, $FV(t_{b_{11}}^{i+1}[x_1 := w_1]...[x_m := w_m]) \subseteq \mathscr{U}$ and $FV(t_{b_{12}}^{i}[x_1 := w_1]...[x_m := w_m]) \subseteq \mathscr{U}$.
By the induction hypothesis, $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t_{b_{11}}^{i+1}[x_1 := w_1]...[x_m := w_m] \longrightarrow^* v_{b_{21}}^{i+1}$, $v_{a_{11}}^{i+1} \simeq v_{b_{21}}^{i+1}$, $\mathscr{U}; \mathscr{V}; \mathscr{X} \cup \mathrm{VAR}(v_{b_{21}}^{i+1}) \vdash t_{b_{12}}^{i+1}[x_1 := w_1]...[x_m := w_m] \longrightarrow^* v_{b_{22}}^{i+1}$ and $v_{a_{12}}^{i+1} \simeq v_{b_{22}}^{i+1}$.
Hence $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash (t_{b_{11}}^{i+1}\ t_{b_{12}}^{i+1})[x_1 := w_1]...[x_m := w_m] \longrightarrow^* v_{b_{21}}^{i+1}\ v_{b_{22}}^{i+1}$ and $v_{a_{11}}^{i+1}\ v_{a_{12}}^{i+1} \simeq v_{b_{21}}^{i+1}\ v_{b_{22}}^{i+1}$.

Case 3. $(v_{a_1}^{0} = \lambda x.t_{a_{11}}^{0})$. We proceed by cases on $t_{b_1}^{0} \in \mathrm{RTERM}_{\mathrm{sus}}^{0}$.

Case i. $(t_{b_1}^{0} = (\underline{\lambda} x_0.t_{b_{11}}^{0})[x_1 := w_1]...[x_m := w_m])$. Then $(\underline{\lambda} x_0.t_{b_{11}}^{0})[x_1 := w_1]...[x_m := w_m] \in \mathrm{VALUE}_{\mathrm{sus}}^{0}$.

Case ii. $(t^0_{b_1} = (\lambda x_0.t^0_{b_{11}})[x_1 := w_1]...[x_m := w_m])$. Given $\lambda x.t^0_{a_{11}} \simeq (\lambda x_0.t^0_{b_{11}})[x_1 := w_1]...[x_m := w_m]$, we have:

$$\begin{array}{rl} & \lambda x.t^0_{a_{11}} \\ = & U((\lambda x_0.t^0_{b_{11}})[x_1 := w_1]...[x_m := w_m]) \\ = & U((\underline{\lambda} x_0.t^0_{b_{11}})[x_1 := w_1]...[x_m := w_m]) \end{array}$$

Then $\lambda x.t^0_{a_{11}} \simeq (\underline{\lambda} x_0.t^0_{b_{11}})[x_1 := w_1]...[x_m := w_m]$.
By (lambda-0) we have:

$$\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash (\lambda x_0.t^0_{b_{11}})[x_1 := w_1]...[x_m := w_m] \longrightarrow (\underline{\lambda} x_0.t^0_{b_{11}})[x_1 := w_1]...[x_m := w_m]$$

Case iii. $(t^0_{b_1} = x_0[x_1 := w_1]...[x_m := w_m])$. This is analogous to the $(t^{i+1}_{b_1} = x_0[x_1 := w_1]...[x_m := w_m])$ subcase of the $(v^{i+1}_{a_1} = x)$ case.

Case 4. $(v^{i+1}_{a_1} = \lambda x.v^{i+1}_{a_{11}})$. We proceed by cases on $t^{i+1}_{b_1} \in \text{RTERM}^{i+1}_{\text{sus}}$.

Case i. $(t^{i+1}_{b_1} = \lambda x_0.v^{i+1}_{b_{11}})$. Then $\lambda x_0.v^{i+1}_{b_{11}} \in \text{VALUE}^{i+1}_{\text{sus}}$.

Case ii. $(t^{i+1}_{b_1} = \lambda x_0.t^{i+1}_{b_{11}}$ where $t^{i+1}_{b_{11}} \notin \text{VALUE}^{i+1}_{\text{sus}})$.
Let $x_N \notin \mathscr{X} \cup \text{VAR}(v^{i+1}_{a_1})$. Then $\lambda x.v^{i+1}_{a_{11}} \sim_\alpha \lambda x_N.v^{i+1}_{a_{11}}[x_N/x]$.
We have $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \lambda x_0.t^{i+1}_{b_{11}} \longrightarrow \hat{\lambda} x_N.t^{i+1}_{b_{11}}[x_0 := x_N]$.
By Lemma 340, $\mathscr{U}; \mathscr{V} \vdash \hat{\lambda} x_N.t^{i+1}_{b_{11}}[x_0 := x_N]$ *wb*. By Lemma 336, $FV(\hat{\lambda} x_N.t^{i+1}_{b_{11}}[x_0 := x_N]) \subseteq \mathscr{V} = \mathscr{U}$.
Then $\mathscr{U} \cup \{x_N\}; \mathscr{V} \cup \{x_N\} \vdash t^{i+1}_{b_{11}}[x_0 := x_N]$ *wb*, $x_N \notin \mathscr{V}$, $FV(t^{i+1}_{b_{11}}[x_0 := x_N]) \subseteq \mathscr{V} \cup \{x_N\}$ and $\mathscr{V} = \mathscr{U}$.
We have:

$$\begin{array}{rl} & \lambda x_N.v^{i+1}_{a_{11}}[x_N/x] \\ = & U(\mathscr{X} \mid \lambda x_0.t^{i+1}_{b_{11}}) \\ = & \lambda x_N.U(\mathscr{X} \cup \{x_N\} \mid t^{i+1}_{b_{11}}[x_0 := x_N]) \end{array}$$

Then $v^{i+1}_{a_{11}}[x_N/x] \simeq t^{i+1}_{b_{11}}[x_0 := x_N]$.
By the induction hypothesis, we have $\mathscr{U} \cup \{x_N\}; \mathscr{V} \cup \{x_N\}; \mathscr{X} \cup \{x_N\} \vdash t^{i+1}_{b_{11}}[x_0 := x_N] \longrightarrow^* v^{i+1}_{b_{12}}$ and $v^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{12}}$.
Hence $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \hat{\lambda} x_N.t^{i+1}_{b_{11}}[x_0 := x_N] \longrightarrow^* \lambda x_N.v^{i+1}_{b_{12}}$ and $\lambda x_N.v^{i+1}_{a_{11}}[x_N/x] \simeq \lambda x_N.v^{i+1}_{b_{12}}$.

Case iii. $(t^{i+1}_{b_1} = \hat{\lambda} x.t^{i+1}_{b_{11}})$. Given $\lambda x.v^{i+1}_{a_{11}} \simeq \hat{\lambda} x.t^{i+1}_{b_{11}}$, we have:

$$\begin{array}{rl} & \lambda x.v^{i+1}_{a_{11}} \\ = & U(\hat{\lambda} x.t^{i+1}_{b_{11}}) \\ = & \lambda x.U(t^{i+1}_{b_{11}}) \end{array}$$

Then $v^{i+1}_{a_{11}} \simeq t^{i+1}_{b_{11}}$.
We have $\mathscr{U} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash t^{i+1}_{b_{11}}$ *wb*, $x \notin \mathscr{V}$, $FV(t^{i+1}_{b_{11}}) \subseteq \mathscr{V} \cup \{x\}$ and $\mathscr{V} = \mathscr{U}$.

By the induction hypothesis, we have $\mathscr{U} \cup \{x\}; \mathscr{V} \cup \{x\}; \mathscr{X} \vdash t_{b_{11}}^{i+1} \longrightarrow^* v_{b_{12}}^{i+1}$ and $v_{a_{11}}^{i+1} \simeq v_{b_{12}}^{i+1}$.
Hence $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \hat{\lambda} x_N.t_{b_{11}}^{i+1} \longrightarrow^* \lambda x_N.v_{b_{12}}^{i+1}$ and $\lambda x.v_{a_{11}}^{i+1} \simeq \lambda x_N.v_{b_{12}}^{i+1}$.

Case iv. $(t_{b_1}^0 = (\underline{\lambda} x_0.t_{b_{11}}^0)[x_1 := w_1]...[x_m := w_m])$. Then $(\underline{\lambda} x_0.t_{b_{11}}^0)[x_1 := w_1]...[x_m := w_m] \in \text{VALUE}_{\text{sus}}^0 \subseteq \text{VALUE}_{\text{sus}}^{i+1}$.

Case v. $(t_{b_1}^{i+1} = x_0[x_1 := w_1]...[x_m := w_m])$. This is analogous to the $(t_{b_1}^{i+1} = x_0[x_1 := w_1]...[x_m := w_m])$ subcase of the $(v_{a_1}^{i+1} = x)$ case.

Case vi. $(t_{b_1}^{i+1} = (\lambda x_0.t_{b_{11}}^{i+1})[x_1 := w_1]...[x_m := w_m])$. Given $\lambda x.v_{a_{11}}^{i+1} \simeq (\lambda x_0.t_{b_{11}}^{i+1})[x_1 := w_1]...[x_m := w_m]$, we have:

$$
\begin{array}{lll}
 & \lambda x.v_{a_{11}}^{i+1} & \\
= & U(\mathscr{X} \mid (\lambda x_0.t_{b_{11}}^{i+1})[x_1 := w_1]...[x_m := w_m]) & \\
= & U(\mathscr{X} \mid (\lambda x_0.t_{b_{11}}^{i+1})[w_1/x_1][x_2 := w_2]...[x_m := w_m]) & \\
= & U(\mathscr{X} \cup \{x_N\} \mid (\lambda x_N.t_{b_{11}}^{i+1}[x_N/x_0][w_1/x_1])[x_2 := w_2]...[x_m := w_m]) & \text{where } x_N \notin \mathscr{X} \\
= & U(\mathscr{X} \cup \{x_N\} \mid (\lambda x_N.t_{b_{11}}^{i+1}[x_0 := x_N][x_1 := w_1])[x_2 := w_2]...[x_m := w_m]) &
\end{array}
$$

Then $\lambda x.v_{a_{11}}^{i+1} \simeq (\lambda x_N.t_{b_{11}}^{i+1}[x_0 := x_N][x_1 := w_1])[x_2 := w_2]...[x_m := w_m]$.
By (lam-subst) and (subst-subst) we have:

$$
\begin{array}{lll}
\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash & (\lambda x_0.t_{b_{11}}^{i+1})[x_1 := w_1][x_2 := w_2]...[x_m := w_m] & \\
\quad \longrightarrow^{\text{x}} & (\lambda x_N.t_{b_{11}}^{i+1}[x_0 := x_N][x_1 := w_1])[x_2 := w_2]...[x_m := w_m] & \text{where } x_N \notin \mathscr{X}
\end{array}
$$

By Lemma 340, $\mathscr{U}; \mathscr{V} \vdash (\lambda x_N.t_{b_{11}}^{i+1}[x_0 := x_N][x_1 := w_1])[x_2 := w_2]...[x_m := w_m]$ *wb*. By Lemma 336, $FV((\lambda x_N.t_{b_{11}}^{i+1}[x_0 := x_N][x_1 := w_1])[x_2 := w_2]...[x_m := w_m]) \subseteq \mathscr{U}$.
Since $(\lambda x_N.t_{b_{11}}^{i+1}[x_0 := x_N][x_1 := w_1])[x_2 := w_2]...[x_m := w_m] \prec^{\text{x}} (\lambda x_0.t_{b_{11}}^{i+1})[x_1 := w_1][x_2 := w_2]...[x_m := w_m]$, by the induction hypothesis, we have $\mathscr{U}; \mathscr{V}; \mathscr{X} \cup \{x_N\} \vdash (\lambda x_N.t_{b_{11}}^{i+1}[x_0 := x_N][x_1 := w_1])[x_2 := w_2]...[x_m := w_m] \longrightarrow^* v_{b_{12}}^{i+1}$ and $\lambda x.v_{a_{11}}^{i+1} \simeq v_{b_{12}}^{i+1}$.

Case 5. $(v_{a_1}^i = \langle v_{a_{11}}^{i+1} \rangle)$. We proceed by cases on $t_{b_1}^i \in \text{RTERM}_{\text{sus}}^i$.

Case i. $(t_{b_1}^i = \langle t_{b_{11}}^{i+1} \rangle)$. Given $\langle v_{a_{11}}^{i+1} \rangle \simeq \langle t_{b_{11}}^{i+1} \rangle$, we have:

$$
\begin{array}{lll}
 & \langle v_{a_{11}}^{i+1} \rangle & \\
= & U(\langle t_{b_{11}}^{i+1} \rangle) & \\
= & \langle U(t_{b_{11}}^{i+1}) \rangle &
\end{array}
$$

Then $v_{a_{11}}^{i+1} \simeq t_{b_{11}}^{i+1}$.
We have $\mathscr{U}; \mathscr{V} \vdash t_{b_{11}}^{i+1}$ *wb* and $FV(t_{b_{11}}^{i+1}) \subseteq \mathscr{V} = \mathscr{U}$.
By the induction hypothesis, $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t_{b_{11}}^{i+1} \longrightarrow^* v_{b_{21}}^{i+1}$ and $v_{a_{11}}^{i+1} \simeq v_{b_{21}}^{i+1}$.
Hence $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \langle t_{b_{11}}^{i+1} \rangle \longrightarrow^* \langle v_{b_{21}}^{i+1} \rangle$ and $\langle v_{a_{11}}^{i+1} \rangle \simeq \langle v_{b_{21}}^{i+1} \rangle$.

Case ii. $(t^i_{b_1} = \langle t^{i+1}_{b_{11}} \rangle [x_1 := w_1]...[x_m := w_m])$. Given $\langle v^{i+1}_{a_{11}} \rangle \simeq \langle t^{i+1}_{b_{11}} \rangle [x_1 := w_1]...[x_m := w_m]$, we have:

$$\begin{array}{rl} & \langle v^{i+1}_{a_{11}} \rangle \\ = & U(\langle t^{i+1}_{b_{11}} \rangle [x_1 := w_1]...[x_m := w_m]) \\ = & \langle U(t^{i+1}_{b_{11}} [x_1 := w_1]...[x_m := w_m]) \rangle \end{array}$$

We have $v^{i+1}_{a_{11}} \simeq t^{i+1}_{b_{11}} [x_1 := w_1]...[x_m := w_m]$.

By (code-subst) and (subst-subst), we have:

$$\begin{array}{rl} \mathscr{U}; \mathscr{V}; \mathscr{X} \vdash & \langle t^{i+1}_{b_{11}} \rangle [x_1 := w_1]...[x_m := w_m] \\ \longrightarrow^{\mathrm{x}*} & \langle t^{i+1}_{b_{11}} [x_1 := w_1]...[x_m := w_m] \rangle \end{array}$$

We have $\mathscr{U}; \mathscr{V} \vdash t^{i+1}_{b_{11}} [x_1 := w_1]...[x_m := w_m]\ wb$ and $FV(t^{i+1}_{b_{11}} [x_1 := w_1]...[x_m := w_m]) \subseteq \mathscr{U}$.

By the induction hypothesis, $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t^{i+1}_{b_{11}} [x_1 := w_1]...[x_m := w_m] \longrightarrow^* v^{i+1}_{b_{21}}$ and $v^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{21}}$.

Hence $\langle t^{i+1}_{b_{11}} \rangle [x_1 := w_1]...[x_m := w_m] \longrightarrow^* \langle v^{i+1}_{b_{21}} \rangle$ and $\langle v^{i+1}_{a_{11}} \rangle \simeq \langle v^{i+1}_{b_{21}} \rangle$.

Case iii. $(v^0_{a_1} = \langle v^1_{a_{11}} \rangle$ and $t^0_{b_1} = x_0[x_1 := w_1]...[x_m := w_m])$. This is analogous to the $(t^{i+1}_{b_1} = x_0[x_1 := w_1]...[x_m := w_m])$ subcase of the $(v^{i+1}_{a_1} = x)$ case.

Case 6. $(v^{i+2}_{a_1} = {\sim} v^{i+1}_{a_{11}})$. This case is analogous to the $(v^i_{a_1} = \langle v^{i+1}_{a_{11}} \rangle)$ case.

Case 7. $(v^{i+1}_{a_1} = !v^{i+1}_{a_{11}})$. This case is analogous to the $(v^i_{a_1} = \langle v^{i+1}_{a_{11}} \rangle)$ case.

Case 8. $(v^i_{a_1} = n)$. We proceed by cases on $t^i_{b_1} \in \mathrm{RTERM}^i_{\mathrm{sus}}$ where $t^i_{b_1}$ is in substitution normal form.

Case i. $(t^i_{b_1} = n)$. Then $n \in \mathrm{VALUE}^i_{\mathrm{sus}}$.

Case ii. $(t^i_{b_1} = x_0[x_1 := w_1]...[x_m := w_m])$. This is analogous to the $(t^{i+1}_{b_1} = x_0[x_1 := w_1]...[x_m := w_m])$ subcase of the $(v^{i+1}_{a_1} = x)$ case.

Case iii. $(t^i_{b_1} = n_0[x_1 := w_1]...[x_m := w_m])$. Given $n \simeq n_0[x_1 := w_1]...[x_m := w_m]$, we have:

$$\begin{array}{rl} & n \\ = & U(n_0[x_1 := w_1]...[x_m := w_m]) \\ = & n_0 \end{array}$$

Then $n \simeq n_0$.

By (num-subst) and (subst-subst), we have:

$$\begin{array}{rl} & n_0[x_1 := w_1]...[x_m := w_m] \\ \longrightarrow^{\mathrm{x}*} & n_0 \end{array}$$

Case 9. $(v^{i+1}_{a_1} = v^{i+1}_{a_{11}} + v^{i+1}_{a_{12}})$. This case is analogous to the $(v^{i+1}_{a_1} = v^{i+1}_{a_{11}}\ v^{i+1}_{a_{12}})$ case.

□

D.2.6 Bisimulation

Lemma 350 (Simulation: Suspended MetaML simulates (Substitutional) MetaML.)**.** *If* $t^i_{a_1} \simeq t^i_{b_1}$, $t^i_{a_1} \longrightarrow^i t^i_{a_2}$, $\mathscr{U};\mathscr{V} \vdash t^i_{b_1}$ *wb,* $\mathrm{VAR}(t^i_{b_1}) \subseteq \mathscr{X}$, $\mathscr{V} \subseteq \mathscr{U} \subseteq \mathscr{X}$, *and either* $t^i_{b_1}$ *is in substitution normal form and* $FV(t^i_{b_1}) \subseteq \mathscr{V} = \mathscr{U}$, *or* $t^i_{b_1}$ *is not in substitution normal form and* $FV(t^i_{b_1}) \subseteq \mathscr{U}$, *then* $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^i_{b_1} \longrightarrow^{i*} t^i_{b_2}$ *and* $t^i_{a_2} \simeq t^i_{b_2}$.

Proof. We proceed by simultaneous induction on the structure of $t^i_{a_1} \in \mathrm{TERM}^i_{\mathrm{sub}}$ and on the explicit substitution descendant relation $\prec^{\mathrm{x}} t^i_{b_1}$.

Case 1. $(t^i_{a_1} = x)$. This case is vacuous because $x \not\longrightarrow$.

Case 2. $(t^i_{a_1} = t^i_{a_{11}}\ t^i_{a_{12}})$. We proceed by cases on $t^i_{b_1} \in \mathrm{RTERM}^i_{\mathrm{sus}}$.

Case i. Suppose $t^i_{b_1}$ is in substitution normal form. Let $t^i_{b_1} = t^i_{b_{11}}\ t^i_{b_{12}}$
We have $t^i_{a_{11}} \simeq t^i_{b_{11}}$, $t^i_{a_{12}} \simeq t^i_{b_{12}}$, $\mathscr{U};\mathscr{V} \vdash t^i_{b_{11}}$ *wb*, $\mathscr{U};\mathscr{V} \vdash t^i_{b_{12}}$ *wb*, $\mathrm{VAR}(t^i_{b_{11}}) \subseteq \mathscr{X}$, $\mathrm{VAR}(t^i_{b_{12}}) \subseteq \mathscr{X}$, $FV(t^i_{b_{11}}) \subseteq \mathscr{V}$, $FV(t^i_{b_{12}}) \subseteq \mathscr{V}$ and $\mathscr{V} = \mathscr{U}$.
We proceed by cases on $t^i_{a_1} \longrightarrow t^i_{a_2}$.

Case a. (appL-i). Let $t^i_{a_{11}}\ t^i_{a_{12}} \longrightarrow t^i_{a_{21}}\ t^i_{a_{12}}$ where $t^i_{a_{11}} \longrightarrow t^i_{a_{21}}$.
By the induction hypothesis, $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^i_{b_{11}} \longrightarrow^* t^i_{b_{21}}$ and $t^i_{a_{21}} \simeq t^i_{b_{21}}$.
Hence $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^i_{b_{11}}\ t^i_{b_{12}} \longrightarrow^* t^i_{b_{21}}\ t^i_{b_{12}}$ and $t^i_{a_{21}}\ t^i_{a_{12}} \simeq t^i_{b_{21}}\ t^i_{b_{12}}$.

Case b. (appR-i). Let $t^i_{a_{11}} = v^i_{a_{11}}$ and $v^i_{a_{11}}\ t^i_{a_{12}} \longrightarrow v^i_{a_{11}}\ t^i_{a_{22}}$ where $t^i_{a_{12}} \longrightarrow t^i_{a_{22}}$.
By Lemma 349, $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^i_{b_{11}} \longrightarrow^* v^i_{b_{11}}$.
Then $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^i_{b_{11}}\ t^i_{b_{12}} \longrightarrow^* v^i_{b_{11}}\ t^i_{b_{12}}$.
By the induction hypothesis, $\mathscr{U};\mathscr{V};\mathscr{X} \cup \mathrm{VAR}(v^i_{b_{11}}) \vdash t^i_{b_{12}} \longrightarrow^* t^i_{b_{22}}$ and $t^i_{a_{22}} \simeq t^i_{b_{22}}$.
Then $\mathscr{U};\mathscr{V};\mathscr{X} \cup \mathrm{VAR}(v^i_{b_{11}}) \vdash v^i_{b_{11}}\ t^i_{b_{12}} \longrightarrow^* v^i_{b_{11}}\ t^i_{b_{22}}$.
Hence $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^i_{b_{11}}\ t^i_{b_{12}} \longrightarrow^* v^i_{b_{11}}\ t^i_{b_{22}}$ and $t^i_{a_{11}}\ t^i_{a_{22}} \simeq v^i_{b_{11}}\ t^i_{b_{22}}$.

Case c. (app-0). Let $t^0_{a_{11}} = \lambda x.t^0_{a_{111}}$, $t^0_{a_{12}} = v^0_{a_{12}}$ and $\lambda x.t^0_{a_{111}}\ v^0_{a_{12}} \longrightarrow t^0_{a_{111}}[v^0_{a_{12}}/x]$.
Given $\lambda x.t^0_{a_{111}} \simeq t^0_{b_{11}}$, by Lemma 349, $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^0_{b_{11}} \longrightarrow^* v^0_{b_{11}}$ and $\lambda x.t^0_{a_{111}} \simeq v^0_{b_{11}}$. Then $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^0_{b_{11}}\ t^0_{b_{12}} \longrightarrow^* v^0_{b_{11}}\ t^0_{b_{12}}$.
Given $v^0_{a_{12}} \simeq t^0_{b_{12}}$, by Lemma 349, $\mathscr{U};\mathscr{V};\mathscr{X} \cup \mathrm{VAR}(v^0_{b_{11}}) \vdash t^0_{b_{12}} \longrightarrow^* v^0_{b_{12}}$ and $v^0_{a_{12}} \simeq v^0_{b_{12}}$. Then $\mathscr{U};\mathscr{V};\mathscr{X} \cup \mathrm{VAR}(v^0_{b_{11}}) \vdash v^0_{b_{11}}\ t^0_{b_{12}} \longrightarrow^* v^0_{b_{11}}\ v^0_{b_{12}}$.
We have $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^0_{b_{11}}\ t^0_{b_{12}} \longrightarrow^* v^0_{b_{11}}\ v^0_{b_{12}}$.
By Lemma 340, $\mathscr{U};\mathscr{V} \vdash v^0_{b_{11}}$ *wb*, $\mathscr{U};\mathscr{V} \vdash v^0_{b_{12}}$ *wb*. By Lemma 336, $FV(v^0_{b_{11}}) \subseteq \mathscr{V}$ and $FV(v^0_{b_{12}}) \subseteq \mathscr{V}$.
We proceed by cases on $v^0_{b_{11}}$ in $\lambda x.t^0_{a_{111}} \simeq v^0_{b_{11}}$. The only possible case is $v^0_{b_{11}} = (\underline{\lambda} x_0.t^0_{b_{111}})[x_1 := w_1]...[x_m := w_m]$.
Let $x_N \notin \mathscr{X} \cup \mathrm{VAR}(t^0_{a_{11}}\ t^0_{a_{12}}) \cup \mathrm{VAR}(v^0_{b_{11}}\ v^0_{b_{12}})$.
We have $\lambda x.t^0_{a_{111}} \sim_\alpha \lambda x_N.t^0_{a_{111}}[x_N/x]$ and $(\underline{\lambda} x_0.t^0_{b_{111}})[x_1 := w_1]...[x_m := w_m] \sim_\alpha (\lambda x_N.t^0_{b_{111}}[x_N/x_0])[x_1 := w_1]...[x_m := w_m]$.
We have:

$$\begin{array}{rl} & \lambda x_N.t^0_{a_{111}}[x_N/x] \\ = & U((\underline{\lambda} x_0.t^0_{b_{111}})[x_1 := w_1]...[x_m := w_m]) \\ = & U((\underline{\lambda} x_N.t^0_{b_{111}}[x_N/x_0])[x_1 := w_1]...[x_m := w_m]) \\ = & U((\underline{\lambda} x_N.t^0_{b_{111}}[x_N/x_0])[w_1/x_1]...[w_m/x_m]) \\ = & U(\underline{\lambda} x_N.t^0_{b_{111}}[x_N/x_0][w_1/x_1]...[w_m/x_m]) \\ = & \lambda x_N.U(t^0_{b_{111}}[x_N/x_0][w_1/x_1]...[w_m/x_m]) \end{array}$$

Then $t^0_{a_{111}}[x_N/x] \simeq t^0_{b_{111}}[x_N/x_0][w_1/x_1]...[w_m/x_m]$.

Observe that

$$\lambda x_N.t^0_{a_{111}}[x_N/x]\ v^0_{a_{12}} \longrightarrow t^0_{a_{111}}[x_N/x][v^0_{a_{12}}/x_N]$$

and

$$\begin{array}{rll} \mathscr{U};\mathscr{V};\mathscr{X} \cup \textsc{Var}(v^0_{b_{11}}\ v^0_{b_{12}}) \vdash & (\underline{\lambda} x_0.t^0_{b_{111}})[x_1 := w_1]...[x_m := w_m]\ v^0_{b_{12}} \\ \longrightarrow & t^0_{b_{111}}[x_0 := v^0_{b_{12}}][x_1 := w_1]...[x_m := w_m] \ . \end{array}$$

Since $FV((\underline{\lambda} x_0.t^0_{b_{111}})[x_1 := w_1]...[x_m := w_m]\ v^0_{b_{12}}) \subseteq \mathscr{V}$, we have $FV(v^0_{b_{12}}) \subseteq \mathscr{V}$.

Since $\mathscr{U};\mathscr{V} \vdash (\lambda x_0.t^0_{b_{111}})[x_1 := w_1]...[x_m := w_m]\ wb$, we have $x_i \notin \mathscr{V}$. Hence $x_i \notin FV(v^0_{b_{12}})$.

We have:

$$\begin{array}{rl} & U(t^0_{b_{111}}[x_0 := v^0_{b_{12}}][x_1 := w_1]...[x_m := w_m]) \\ = & U(t^0_{b_{111}}[v^0_{b_{12}}/x_0][w_1/x_1]...[w_m/x_m]) \\ = & U(t^0_{b_{111}}[x_N/x_0][v^0_{b_{12}}/x_N][w_1/x_1]...[w_m/x_m]) \\ = & U(t^0_{b_{111}}[x_N/x_0][w_1/x_1]...[w_m/x_m][v^0_{b_{12}}/x_N]) \\ = & U(t^0_{b_{111}})[x_N/x_0][U(w_1)/x_1]...[U(w_m)/x_m][U(v^0_{b_{12}})/x_N] \\ = & U(t^0_{b_{111}}[x_N/x_0][w_1/x_1]...[w_m/x_m])[U(v^0_{b_{12}})/x_N] \\ = & t^0_{a_{111}}[x_N/x][v^0_{a_{12}}/x_N] \end{array}$$

We get:

$$t^0_{a_{111}}[x_N/x][v^0_{a_{12}}/x_N] \simeq t^0_{b_{111}}[x_0 := v^0_{b_{12}}][x_1 := w_1]...[x_m := w_m]$$

Case ii. Suppose $t^i_{b_1}$ is not in substitution normal form. Let $t^i_{b_1} = (t^i_{b_{11}}\ t^i_{b_{12}})[x_1 := w_1]...[x_m := w_m]$.

We have $\mathscr{U};\mathscr{V} \vdash t^i_{b_{11}}[x_1 := w_1]...[x_m := w_m]\ t^i_{b_{12}}[x_1 := w_1]...[x_m := w_m]\ wb$, $\textsc{Var}(t^i_{b_{11}}[x_1 := w_1]...[x_m := w_m]\ t^i_{b_{12}}[x_1 := w_1]...[x_m := w_m]) \subseteq \mathscr{X}$ and $FV(t^i_{b_{11}}[x_1 := w_1]...[x_m := w_m]\ t^i_{b_{12}}[x_1 := w_1]...[x_m := w_m]) \subseteq \mathscr{U}$.

By (app-subst) and (subst-subst), we have $\mathscr{U};\mathscr{V};\mathscr{X} \vdash (t^i_{b_{11}}\ t^i_{b_{12}})[x_1 := w_1]...[x_m := w_m] \longrightarrow^{\mathrm{x}*} t^i_{b_{11}}[x_1 := w_1]...[x_m := w_m]\ t^i_{b_{12}}[x_1 := w_1]...[x_m := w_m]$.

Then $t^i_{b_{11}}[x_1 := w_1]...[x_m := w_m]\ t^i_{b_{12}}[x_1 := w_1]...[x_m := w_m] \prec^{\mathrm{x}} (t^i_{b_{11}}\ t^i_{b_{12}})[x_1 := w_1]...[x_m := w_m]$.

Given $t^i_{a_{11}}\ t^i_{a_{12}} \simeq (t^i_{b_{11}}\ t^i_{b_{12}})[x_1 := w_1]...[x_m := w_m]$, we have $t^i_{a_{11}} \simeq t^i_{b_{11}}[x_1 := w_1]...[x_m :=$

$w_m]$ and $t^i_{a_{12}} \simeq t^i_{b_{12}}[x_1 := w_1]...[x_m := w_m]$.
Suppose $t^i_{a_{11}}\ t^i_{a_{12}} \longrightarrow t^i_{a_2}$, by the induction hypothesis, $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t^i_{b_{11}}[x_1 := w_1]...[x_m := w_m]\ t^i_{b_{12}}[x_1 := w_1]...[x_m := w_m] \longrightarrow^* t^i_{b_2}$ and $t^i_{a_2} \simeq t^i_{b_2}$.

Case 3. $(t^{i+1}_{a_1} = \lambda x.t^{i+1}_{a_{11}})$. We proceed by cases on $t^{i+1}_{a_1} \longrightarrow t^{i+1}_{a_2}$. The only case is (lambda-(i+1)).
Let $\mathscr{U}; \mathscr{V} \vdash t^{i+1}_{b_1}$ *wb*. Let $\textsc{Var}(t^{i+1}_{b_1}) \subseteq \mathscr{X}$. Let $\mathscr{V} \subseteq \mathscr{U} \subseteq \mathscr{X}$.
We proceed by cases on $t^{i+1}_{b_1} \in \textsc{RTerm}^{i+1}_{\text{sus}}$.

Case i. Suppose $t^{i+1}_{b_1}$ is in substitution normal form. We have $\mathscr{U}; \mathscr{V} \vdash t^{i+1}_{b_1}$ *wb*, $\textsc{Var}(t^{i+1}_{b_1}) \subseteq \mathscr{X}$, $FV(t^{i+1}_{b_1}) \subseteq \mathscr{V}$ and $\mathscr{V} = \mathscr{U}$.

Case a. Let $t^{i+1}_{b_1} = \lambda x_0.t^{i+1}_{b_{11}}$.
Let $x_N \notin \mathscr{X} \cup \textsc{Var}(t^{i+1}_{a_1})$.
We have $\lambda x.t^{i+1}_{a_{11}} \sim_\alpha \lambda x_N.t^{i+1}_{a_{11}}[x_N/x]$. We also have $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \lambda x_0.t^{i+1}_{b_{11}} \longrightarrow \hat{\lambda} x_N.t^{i+1}_{b_{11}}[x_0 := x_N]$.
By Lemma 340, we have $\mathscr{U}; \mathscr{V} \vdash \hat{\lambda} x_N.t^{i+1}_{b_{11}}[x_0 := x_N]$ *wb*. By Lemma 336, $FV(\hat{\lambda} x_N.t^{i+1}_{b_{11}}[x_0 := x_N]) \subseteq \mathscr{V}$.
We have $\mathscr{U} \cup \{x_N\}; \mathscr{V} \cup \{x_N\} \vdash t^{i+1}_{b_{11}}[x_0 := x_N]$ *wb*, $x_N \notin \mathscr{V}$, $FV(t^{i+1}_{b_{11}}[x_0 := x_N]) \subseteq \mathscr{V} \cup \{x_N\}$ and $\mathscr{V} = \mathscr{U}$.
We have:

$$\begin{array}{rl} & \lambda x_N.t^{i+1}_{a_{11}}[x_N/x] \\ = & U(\mathscr{X} \mid \lambda x_0.t^{i+1}_{b_{11}}) \\ = & \lambda x_N.U(\mathscr{X} \cup \{x_N\} \mid t^{i+1}_{b_{11}}[x_0 := x_N]) \end{array}$$

Then $t^{i+1}_{a_{11}}[x_N/x] \simeq t^{i+1}_{b_{11}}[x_0 := x_N]$.
Suppose $\lambda x_N.t^{i+1}_{a_{11}}[x_N/x] \longrightarrow \lambda x_N.t^{i+1}_{a_{21}}$ where $t^{i+1}_{a_{11}}[x_N/x] \longrightarrow t^{i+1}_{a_{21}}$.
By the induction hypothesis, $\mathscr{U} \cup \{x_N\}; \mathscr{V} \cup \{x_N\}; \mathscr{X} \cup \{x_N\} \vdash t^{i+1}_{b_{11}}[x_0 := x_N] \longrightarrow^* t^{i+1}_{b_{21}}$ and $t^{i+1}_{a_{21}} \simeq t^{i+1}_{b_{21}}$.
Hence $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \lambda x_N.t^{i+1}_{b_{11}} \longrightarrow^* \hat{\lambda} x_N.t^{i+1}_{b_{21}}$ and $\lambda x_N.t^{i+1}_{a_{21}} \simeq \hat{\lambda} x_N.t^{i+1}_{b_{21}}$.

Case b. Let $t^{i+1}_{b_1} = \hat{\lambda} x.t^{i+1}_{b_{11}}$. We have:

$$\begin{array}{rl} & \lambda x.t^{i+1}_{a_{11}} \\ = & U(\hat{\lambda} x.t^{i+1}_{b_{11}}) \\ = & \lambda x.U(t^{i+1}_{b_{11}}) \end{array}$$

Then $t^{i+1}_{a_{11}} \simeq t^{i+1}_{b_{11}}$.
We have $\mathscr{U} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash t^{i+1}_{b_{11}}$ *wb*, $x \notin \mathscr{V}$, $FV(t^{i+1}_{b_{11}}) \subseteq \mathscr{V} \cup \{x\}$ and $\mathscr{V} = \mathscr{U}$.
Suppose $\lambda x.t^{i+1}_{a_{11}} \longrightarrow \lambda x.t^{i+1}_{a_{21}}$ where $t^{i+1}_{a_{11}} \longrightarrow t^{i+1}_{a_{21}}$.
By the induction hypothesis, $\mathscr{U} \cup \{x\}; \mathscr{V} \cup \{x\}; \mathscr{X} \vdash t^{i+1}_{b_{11}} \longrightarrow^* t^{i+1}_{b_{21}}$ and $t^{i+1}_{a_{21}} \simeq t^{i+1}_{b_{21}}$.
Hence $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \hat{\lambda} x.t^{i+1}_{b_{11}} \longrightarrow^* \hat{\lambda} x.t^{i+1}_{b_{21}}$ and $\lambda x.t^{i+1}_{a_{21}} \simeq \hat{\lambda} x.t^{i+1}_{b_{21}}$.

Case ii. Suppose $t^{i+1}_{b_1}$ is in not substitution normal form. We have $\mathscr{U}; \mathscr{V} \vdash t^{i+1}_{b_1}$ *wb*, $\textsc{Var}(t^{i+1}_{b_1}) \subseteq \mathscr{X}$, $FV(t^{i+1}_{b_1}) \subseteq \mathscr{U}$.

Case a. Let $t_{b_1}^{i+1} = (\lambda x_0.t_{b_{11}}^{i+1})[x_1 := w_1][x_2 := w_2]...[x_m := w_m]$. We proceed by cases on m.

Case 1. ($m = 1$). Let $x_{N_1}, x_{N_2} \notin \mathscr{X} \cup \textsc{Var}(t_{a_1}^{i+1})$ and $x_{N_1} \not\equiv x_{N_2}$.
We have $\lambda x.t_{a_{11}}^{i+1} \sim_\alpha \lambda x_{N_2}.t_{a_{11}}^{i+1}[x_{N_2}/x]$. We also have $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash (\lambda x_0.t_{b_{11}}^{i+1})[x_1 := w_1] \longrightarrow^{\mathrm{x}} \lambda x_{N_1}.t_{b_{11}}^{i+1}[x_0 := x_{N_1}][x_1 := w_1]$ and $\mathscr{U}; \mathscr{V}; \mathscr{X} \cup \{x_{N_1}\} \vdash \lambda x_{N_1}.t_{b_{11}}^{i+1}[x_0 := x_{N_1}][x_1 := w_1] \longrightarrow^{\mathrm{x}} \hat{\lambda} x_{N_2}.t_{b_{11}}^{i+1}[x_0 := x_{N_1}][x_1 := w_1][x_{N_1} := x_{N_2}]$.
By Lemma 340, $\mathscr{U}; \mathscr{V} \vdash \hat{\lambda} x_{N_2}.t_{b_{11}}^{i+1}[x_0 := x_{N_1}][x_1 := w_1][x_{N_1} := x_{N_2}]$ *wb*. By Lemma 336, $\hat{\lambda} x_{N_2}.t_{b_{11}}^{i+1}[x_0 := x_{N_1}][x_1 := w_1][x_{N_1} := x_{N_2}] \subseteq \mathscr{U}$.
We have $\mathscr{U} \cup \{x_{N_2}\}; \mathscr{V} \cup \{x_{N_2}\} \vdash t_{b_{11}}^{i+1}[x_0 := x_{N_1}][x_1 := w_1][x_{N_1} := x_{N_2}]$ *wb*, $x_{N_2} \notin \mathscr{V}$, $FV(t_{b_{11}}^{i+1}[x_0 := x_{N_1}][x_1 := w_1][x_{N_1} := x_{N_2}]) \subseteq \mathscr{U} \cup \{x_{N_2}\}$.
We have:

$$\begin{array}{rl} & \lambda x_{N_2}.t_{a_{11}}^{i+1}[x_{N_2}/x] \\ = & U(\mathscr{X} \mid (\lambda x_0.t_{b_{11}}^{i+1})[x_1 := w_1]) \\ = & U(\mathscr{X} \mid (\lambda x_0.t_{b_{11}}^{i+1})[w_1/x_1]) \\ = & U(\mathscr{X} \cup \{x_{N_1}\} \mid \lambda x_{N_1}.t_{b_{11}}^{i+1}[x_{N_1}/x_0][w_1/x_1]) \\ = & \lambda x_{N_2}.U(\mathscr{X} \cup \{x_{N_1}, x_{N_2}\} \mid t_{b_{11}}^{i+1}[x_{N_1}/x_0][w_1/x_1][x_{N_1} := x_{N_2}]) \\ = & \lambda x_{N_2}.U(\mathscr{X} \cup \{x_{N_1}, x_{N_2}\} \mid t_{b_{11}}^{i+1}[x_0 := x_{N_1}][x_1 := w_1][x_{N_1} := x_{N_2}]) \end{array}$$

Then $t_{a_{11}}^{i+1}[x_{N_2}/x] \simeq t_{b_{11}}^{i+1}[x_0 := x_{N_1}][x_1 := w_1][x_0 := x_{N_2}]$.
Suppose $\lambda x_{N_2}.t_{a_{11}}^{i+1}[x_{N_2}/x] \longrightarrow \lambda x_{N_2}.t_{a_{21}}^{i+1}$ where $t_{a_{11}}^{i+1}[x_{N_2}/x] \longrightarrow t_{a_{21}}^{i+1}$.
By the induction hypothesis, $\mathscr{U} \cup \{x_{N_2}\}; \mathscr{V} \cup \{x_{N_2}\}; \mathscr{X} \cup \{x_{N_1}, x_{N_2}\} \vdash t_{b_{11}}^{i+1}[x_0 := x_{N_1}][x_1 := w_1][x_0 := x_{N_2}] \longrightarrow^* t_{b_{21}}^{i+1}$ and $t_{a_{21}}^{i+1} \simeq t_{b_{21}}^{i+1}$.
Hence $\mathscr{U}; \mathscr{V}; \mathscr{X} \cup \{x_{N_1}, x_{N_2}\} \vdash \lambda x_{N_2}.t_{b_{11}}^{i+1}[x_0 := x_{N_1}][x_1 := w_1][x_0 := x_{N_2}] \longrightarrow^* \hat{\lambda} x_{N_2}.t_{b_{21}}^{i+1}$ and $\lambda x_{N_2}.t_{a_{21}}^{i+1} \simeq \hat{\lambda} x_{N_2}.t_{b_{21}}^{i+1}$.

Case 2. ($m > 1$). Let $x_N \notin \mathscr{X}$.
We have $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash (\lambda x_0.t_{b_{11}}^{i+1})[x_1 := w_1][x_2 := w_2]...[x_m := w_m] \longrightarrow^{\mathrm{x}} (\lambda x_N.t_{b_{11}}^{i+1}[x_0 := x_N][x_1 := w_1])[x_2 := w_2]...[x_m := w_m]$.
By Lemma 340, $\mathscr{U}; \mathscr{V} \vdash (\lambda x_N.t_{b_{11}}^{i+1}[x_0 := x_N][x_1 := w_1])[x_2 := w_2]...[x_m := w_m]$ *wb*. By Lemma 336, $(\lambda x_N.t_{b_{11}}^{i+1}[x_0 := x_N][x_1 := w_1])[x_2 := w_2]...[x_m := w_m] \subseteq \mathscr{U}$.
Then $(\lambda x_N.t_{b_{11}}^{i+1}[x_0 := x_N][x_1 := w_1])[x_2 := w_2]...[x_m := w_m] \prec^{\mathrm{x}} (\lambda x_0.t_{b_{11}}^{i+1})[x_1 := w_1][x_2 := w_2]...[x_m := w_m]$.
Provably we have $\lambda x.t_{a_{11}}^{i+1} = U((\lambda x_0.t_{b_{11}}^{i+1})[x_1 := w_1]...[x_m := w_m]) = U((\lambda x_N.t_{b_{11}}^{i+1}[x_0 := x_N][x_1 := w_1])[x_2 := w_2]...[x_m := w_m])$.
Given $\lambda x.t_{a_{11}}^{i+1} \longrightarrow \lambda x.t_{a_{21}}^{i+1}$, by the induction hypothesis, $\mathscr{U}; \mathscr{V}; \mathscr{X} \cup$

$\{x_N\} \vdash (\lambda x_N.t_{b_{11}}^{i+1}[x_0 := x_N][x_1 := w_1])[x_2 := w_2]...[x_m := w_m] \longrightarrow^*$
$t_{b_2}^{i+1}$ and $\lambda x.t_{a_{21}}^{i+1} \simeq t_{b_2}^{i+1}$.

Case 4. ($t_{a_1}^i = n$). This case is vacuous because $n \not\longrightarrow^i$.

Case 5. ($t_{a_1}^i = t_{a_{11}}^i + t_{a_{12}}^i$). This case is analogous to the ($t_{a_1}^i = t_{a_{11}}^i\ t_{a_{12}}^i$) case.

Case 6. ($t_{a_1}^i = \langle t_{a_{11}}^{i+1} \rangle$). We proceed by cases on $t_{a_1}^i \longrightarrow t_{a_2}^i$. The only case is (code-i).
Then $\langle t_{a_{11}}^{i+1} \rangle \longrightarrow \langle t_{a_{21}}^{i+1} \rangle$ where $t_{a_{11}}^{i+1} \longrightarrow t_{a_{21}}^{i+1}$.
Let $\mathscr{U}; \mathscr{V} \vdash t_{b_1}^i$ *wb*. Let $\textsc{Var}(t_{b_1}^i) \subseteq \mathscr{X}$. Let $\mathscr{V} \subseteq \mathscr{U} \subseteq \mathscr{X}$.
We proceed by cases on $t_{b_1}^i \in \textsc{RTerm}_{\text{sus}}^i$.

Case i. Suppose $t_{b_1}^i$ is in substitution normal form. We have $\mathscr{U}; \mathscr{V} \vdash t_{b_1}^i$ *wb*, $\textsc{Var}(t_{b_1}^i) \subseteq \mathscr{X}$, $FV(t_{b_1}^i) \subseteq \mathscr{V}$ and $\mathscr{V} = \mathscr{U}$.

Case a. Let $t_{b_1}^i = \langle t_{b_{11}}^{i+1} \rangle$.
Then $t_{a_{11}}^{i+1} \simeq t_{b_{11}}^{i+1}$.
We have $\mathscr{U}; \mathscr{V} \vdash t_{b_{11}}^{i+1}$ *wb*, $FV(t_{b_{11}}^{i+1}) \subseteq \mathscr{V}$ and $\mathscr{V} = \mathscr{U}$.
By the induction hypothesis, $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t_{b_{11}}^{i+1} \longrightarrow^* t_{b_{21}}^{i+1}$ and $t_{a_{21}}^{i+1} \simeq t_{b_{21}}^{i+1}$.
Hence $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \langle t_{b_{11}}^{i+1} \rangle \longrightarrow^* \langle t_{b_{11}}^{i+1} \rangle$ and $\langle t_{a_{21}}^{i+1} \rangle \simeq \langle t_{b_{21}}^{i+1} \rangle$.

Case ii. Suppose $t_{b_1}^i$ is in not substitution normal form. We have $\mathscr{U}; \mathscr{V} \vdash t_{b_1}^i$ *wb*, $\textsc{Var}(t_{b_1}^i) \subseteq \mathscr{X}$ and $FV(t_{b_1}^i) \subseteq \mathscr{U}$.

Case a. Let $t_{b_1}^{i+1} = \langle t_{b_{11}}^{i+1} \rangle [x_1 := w_1]...[x_m := w_m]$.
We have $\mathscr{U}; \mathscr{V} \vdash t_{b_{11}}^{i+1}[x_1 := w_1]...[x_m := w_m]$ *wb* and $FV(t_{b_{11}}^{i+1}[x_1 := w_1]...[x_m := w_m]) \subseteq \mathscr{U}$.
By (code-subst) and (subst-subst), we have $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \langle t_{b_{11}}^{i+1} \rangle [x_1 := w_1]...[x_m := w_m] \longrightarrow^{\text{x}*} \langle t_{b_{11}}^{i+1}[x_1 := w_1]...[x_m := w_m] \rangle$.
Then $\langle t_{b_{11}}^{i+1}[x_1 := w_1]...[x_m := w_m] \rangle \prec^{\text{x}} \langle t_{b_{11}}^{i+1} \rangle [x_1 := w_1]...[x_m := w_m]$.
Provably we have $\langle t_{a_{11}}^{i+1} \rangle = U(\langle t_{b_{11}}^{i+1} \rangle [x_1 := w_1]...[x_m := w_m]) = U(\langle t_{b_{11}}^{i+1}[x_1 := w_1]...[x_m := w_m] \rangle)$.
Given $\langle t_{a_{11}}^{i+1} \rangle \longrightarrow \langle t_{a_{21}}^{i+1} \rangle$, by the induction hypothesis, $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash \langle t_{b_{11}}^{i+1}[x_1 := w_1]...[x_m := w_m] \rangle \longrightarrow^* t_{b_2}^i$ and $\langle t_{a_{21}}^{i+1} \rangle \simeq t_{b_2}^i$.

Case 7. ($t_{a_1}^{i+1} = {\sim} t_{a_{11}}^i$). This case is analogous to the ($t_{a_1}^i = \langle t_{a_{11}}^{i+1} \rangle$) case.

Case 8. ($t_{a_1}^i = !t_{a_{11}}^i$). This case is analogous to the ($t_{a_1}^i = \langle t_{a_{11}}^{i+1} \rangle$) case.

□

Lemma 351 (Simulation: (Substitutional) MetaML simulates Suspended MetaML.)**.** *If $t_{a_1}^i \simeq t_{b_1}^i$, $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t_{b_1}^i \longrightarrow^i t_{b_2}^i$, $\mathscr{U}; \mathscr{V} \vdash t_{b_1}^i$ wb, $\textsc{Var}(t_{b_1}^i) \subseteq \mathscr{X}$, $\mathscr{V} \subseteq \mathscr{U} \subseteq \mathscr{X}$, and either $t_{b_1}^i$ is in substitution normal form and $FV(t_{b_1}^i) \subseteq \mathscr{V} = \mathscr{U}$, or $t_{b_1}^i$ is not in substitution normal form and $FV(t_{b_1}^i) \subseteq \mathscr{U}$, then $t_{a_1}^i \longrightarrow^{i*} t_{a_2}^i$ and $t_{a_2}^i \simeq t_{b_2}^i$.*

Proof. We proceed by induction on the structure of $t^i_{b_1} \in \mathrm{RTERM}^i_{\mathrm{sus}}$.

Case 1. ($t^i_{b_1} = x$). This case is vacuous because $t^i_{b_1} \not\longrightarrow^i$.

Case 2. ($t^i_{b_1} = t^i_{b_{11}}\ t^i_{b_{12}}$). Let $t^i_{a_1} = t^i_{a_{11}}\ t^i_{a_{12}}$. We have $t^i_{a_{11}} \simeq t^i_{b_{11}}$ and $t^i_{a_{12}} \simeq t^i_{b_{12}}$.
We also have $\mathscr{U};\mathscr{V} \vdash t^i_{b_{11}}\ wb$, $\mathscr{U};\mathscr{V} \vdash t^i_{b_{12}}\ wb$, $FV(t^i_{b_{11}}) \subseteq \mathscr{V}$, $FV(t^i_{b_{12}}) \subseteq \mathscr{V}$ and $\mathscr{V} = \mathscr{U}$.
We proceed by cases on $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^i_{b_1} \longrightarrow t^i_{b_2}$.

Case i. (appL-i). Let $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^i_{b_{11}}\ t^i_{b_{12}} \longrightarrow t^i_{b_{21}}\ t^i_{b_{12}}$ where $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^i_{b_{11}} \longrightarrow t^i_{b_{21}}$.
By the induction hypothesis, $t^i_{a_{11}} \longrightarrow^* t^i_{a_{21}}$ and $t^i_{a_{21}} \simeq t^i_{b_{21}}$.
We have $t^i_{a_{11}}\ t^i_{a_{12}} \longrightarrow^* t^i_{a_{21}}\ t^i_{a_{12}}$ and $t^i_{a_{21}}\ t^i_{a_{12}} \simeq t^i_{b_{21}}\ t^i_{b_{12}}$.

Case ii. (appR-i). Let $t^i_{b_{11}} = v^i_{b_{11}}$ and $\mathscr{U};\mathscr{V};\mathscr{X} \vdash v^i_{b_{11}}\ t^i_{b_{12}} \longrightarrow v^i_{b_{11}}\ t^i_{b_{22}}$ where $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^i_{b_{12}} \longrightarrow t^i_{b_{22}}$.
By Lemma 348, $t^i_{a_{11}} = v^i_{a_{11}}$.
By the induction hypothesis, $t^i_{a_{12}} \longrightarrow^* t^i_{a_{22}}$ and $t^i_{a_{22}} \simeq t^i_{b_{22}}$.
We have $v^i_{a_{11}}\ t^i_{a_{12}} \longrightarrow^* v^i_{a_{11}}\ t^i_{a_{22}}$ and $v^i_{a_{11}}\ t^i_{a_{12}} \simeq v^i_{b_{11}}\ t^i_{b_{22}}$.

Case iii. (app). Let $t^0_{b_{11}} = (\underline{\lambda}x.t^0_{b_{111}})[x_1 := w_1]...[x_m := w_m]$, $t^0_{b_{12}} = v^0_{b_{12}}$ and $\mathscr{U};\mathscr{V};\mathscr{X} \vdash (\underline{\lambda}x.t^0_{b_{11}})[x_1 := w_1]...[x_m := w_m]\ v^0_{b_{12}} \longrightarrow t^0_{b_{11}}[x := v^0_{b_{12}}][x_1 := w_1]...[x_m := w_m]$.
By Lemma 348, we know $t^0_{a_{12}} = v^0_{a_{12}}$.
Let $x_N \notin \mathscr{X}$. We have $(\underline{\lambda}x.t^0_{b_{111}})[x_1 := w_1]...[x_m := w_m] \sim_\alpha (\underline{\lambda}x_N.t^0_{b_{111}}[x_N/x_0])[x_1 := w_1]...[x_m := w_m]$.
We have:

$$\begin{array}{rl} & t^0_{a_{11}} \\ = & U((\underline{\lambda}x.t^0_{b_{111}})[x_1 := w_1]...[x_m := w_m]) \\ = & U((\underline{\lambda}x.t^0_{b_{111}})[w_1/x_1]...[w_m/x_m]) \\ = & U(\underline{\lambda}x_N.t^0_{b_{111}}[x_N/x_0][w_1/x_1]...[w_m/x_m]) \\ = & \lambda x_N.U(t^0_{b_{111}}[x_N/x_0][w_1/x_1]...[w_m/x_m]) \\ = & \lambda x_N.U(t^0_{b_{111}})[x_N/x_0][U(w_1)/x_1]...[U(w_m)/x_m]) \end{array}$$

Observe that

$$\begin{array}{rl} & \lambda x_N.U(t^0_{b_{111}})[x_N/x_0][U(w_1)/x_1]...[U(w_m)/x_m])\ v^0_{a_{12}} \\ \longrightarrow & U(t^0_{b_{111}})[x_N/x_0][U(w_1)/x_1]...[U(w_m)/x_m][v^0_{a_{12}}/x_N] \end{array}$$

and

$$\begin{array}{rl} \mathscr{U};\mathscr{V};\mathscr{X} \vdash & (\underline{\lambda}x.t^0_{b_{111}})[x_1 := w_1]...[x_m := w_m]\ v^0_{b_{12}} \\ \longrightarrow & t^0_{b_{111}}[x_0 := v^0_{b_{12}}][x_1 := w_1]...[x_m := w_m]\ . \end{array}$$

Since $\mathscr{U};\mathscr{V} \vdash (\underline{\lambda}x.t^0_{b_{111}})[x_1 := w_1]...[x_m := w_m]\ wb$, we have $x_i \notin \mathscr{V}$.

Hence $x_i \notin FV(v^0_{b_{12}})$.

We have:

$$
\begin{array}{rl}
& U(t^0_{b_{111}}[x_0 := v^0_{b_{12}}][x_1 := w_1]...[x_m := w_m]) \\
= & U(t^0_{b_{111}}[v^0_{b_{12}}/x_0][w_1/x_1]...[w_m/x_m]) \\
= & U(t^0_{b_{111}}[x_N/x_0][v^0_{b_{12}}/x_N][w_1/x_1]...[w_m/x_m]) \\
= & U(t^0_{b_{111}}[x_N/x_0][w_1/x_1]...[w_m/x_m][v^0_{b_{12}}/x_N]) \\
= & U(t^0_{b_{111}})[x_N/x_0][U(w_1)/x_1]...[U(w_m)/x_m][U(v^0_{b_{12}})/x_N] \\
= & U(t^0_{b_{111}})[x_N/x_0][U(w_1)/x_1]...[U(w_m)/x_m])[v^0_{a_{12}}/x_N]
\end{array}
$$

We get:

$$
\begin{array}{rl}
& U(t^0_{b_{111}})[x_N/x_0][U(w_1)/x_1]...[U(w_m)/x_m][v^0_{a_{12}}/x_N] \\
\simeq & t^0_{b_{111}}[x_0 := v^0_{b_{12}}][x_1 := w_1]...[x_m := w_m]
\end{array}
$$

Case 3. $(t^{i+1}_{b_1} = \lambda x.t^{i+1}_{b_{11}})$. Let $t^{i+1}_{a_1} = \lambda x.t^{i+1}_{a_{11}}$.
Let $x_N \notin \mathscr{X} \cup \mathrm{VAR}(t^{i+1}_{a_1})$. We have $\lambda x.t^{i+1}_{a_{11}} \sim_\alpha \lambda x_N.t^{i+1}_{a_{11}}[x_N/x]$.
We also have $\lambda x_N.t^{i+1}_{a_{11}}[x_N/x] = U(\lambda x.t^{i+1}_{b_{11}}) = \lambda x_N.U(t^{i+1}_{b_{11}}[x := x_N])$.
Then $t^{i+1}_{a_{11}}[x_N/x] \simeq t^{i+1}_{b_{11}}[x := x_N]$.
We proceed by cases on $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^{i+1}_{b_1} \longrightarrow t^{i+1}_{b_2}$. The only case is (lambda-(i+1)-t).
Let $\mathscr{U};\mathscr{V};\mathscr{X} \vdash \lambda x.t^{i+1}_{b_{11}} \longrightarrow \hat{\lambda} x_N.t^{i+1}_{b_{11}}[x := x_N]$.
We have $\lambda x_N.t^{i+1}_{a_{11}}[x_N/x] \longrightarrow^* \lambda x_N.t^{i+1}_{a_{11}}[x_N/x]$ and $\lambda x_N.t^{i+1}_{a_{11}}[x_N/x] \simeq \lambda x_N.t^{i+1}_{b_{12}}[x := x_N]$.

Case 4. $(t^{i+1}_{b_1} = \hat{\lambda} x.t^{i+1}_{b_{11}})$. Let $t^{i+1}_{a_1} = \lambda x.t^{i+1}_{a_{11}}$. We have $t^{i+1}_{a_{11}} \simeq t^{i+1}_{b_{11}}$.
We also have $\mathscr{U} \cup \{x\};\mathscr{V} \cup \{x\} \vdash t^{i+1}_{b_{11}}\ wb$, $x \notin \mathscr{V}$, $FV(t^{i+1}_{b_{11}}) \subseteq \mathscr{V} \cup \{x\}$ and $\mathscr{V} = \mathscr{U}$.
We proceed by cases on $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^{i+1}_{b_1} \longrightarrow t^{i+1}_{b_2}$.

Case i. (lambda-(i+1)-r). Let $\mathscr{U};\mathscr{V};\mathscr{X} \vdash \hat{\lambda} x.t^{i+1}_{b_{11}} \longrightarrow \hat{\lambda} x.t^{i+1}_{b_{12}}$ where $\mathscr{U} \cup \{x\};\mathscr{V} \cup \{x\};\mathscr{X} \vdash t^{i+1}_{b_{11}} \longrightarrow t^{i+1}_{b_{12}}$.
By the induction hypothesis, $t^{i+1}_{a_{11}} \longrightarrow^* t^{i+1}_{a_{12}}$ and $t^{i+1}_{a_{12}} \simeq t^{i+1}_{b_{12}}$.
We have $\lambda x.t^{i+1}_{a_{11}} \longrightarrow^* \lambda x.t^{i+1}_{a_{12}}$ and $\lambda x.t^{i+1}_{a_{12}} \simeq \hat{\lambda} x.t^{i+1}_{b_{12}}$.

Case ii. (lambda-(i+1)-v). Let $t^{i+1}_{b_{11}} = v^{i+1}_{b_{11}}$ and $\mathscr{U};\mathscr{V};\mathscr{X} \vdash \hat{\lambda} x.v^{i+1}_{b_{11}} \longrightarrow \lambda x.v^{i+1}_{b_{11}}$.
We have $\lambda x.t^{i+1}_{a_{11}} \longrightarrow^* \lambda x.t^{i+1}_{a_{11}}$ and $\lambda x.t^{i+1}_{a_{11}} \simeq \lambda x.v^{i+1}_{b_{11}}$.

Case 5. $(t^i_{b_1} = t^i_{b_{11}} + t^i_{b_{12}})$. This case is analogous to the $(t^i_{b_1} = t^i_{b_{11}}\ t^i_{b_{12}})$ case.

Case 6. $(t^i_{b_1} = \langle t^{i+1}_{b_{11}} \rangle)$. Let $t^i_{a_1} = \langle t^{i+1}_{a_{11}} \rangle$. We have $t^{i+1}_{a_{11}} \simeq t^{i+1}_{b_{11}}$.
We also have $\mathscr{U};\mathscr{V} \vdash t^{i+1}_{b_{11}}\ wb$, $FV(t^{i+1}_{b_{11}}) \subseteq \mathscr{V}$, and $\mathscr{V} = \mathscr{U}$.
We proceed by cases on $\mathscr{U};\mathscr{V} \vdash t^i_{b_1} \longrightarrow t^i_{b_2}$. The only case is (code-i).
Let $\mathscr{U};\mathscr{V};\mathscr{X} \vdash \langle t^{i+1}_{b_{11}} \rangle \longrightarrow \langle t^{i+1}_{b_{12}} \rangle$ where $\mathscr{U};\mathscr{V};\mathscr{X} \vdash t^{i+1}_{b_{11}} \longrightarrow t^{i+1}_{b_{12}}$.
By the induction hypothesis, $t^{i+1}_{a_{11}} \longrightarrow^* t^{i+1}_{a_{12}}$ and $t^{i+1}_{a_{12}} \simeq t^{i+1}_{b_{12}}$.
We have $\langle t^{i+1}_{a_{11}} \rangle \longrightarrow^* \langle t^{i+1}_{a_{12}} \rangle$ and $\langle t^{i+1}_{a_{12}} \rangle \simeq \langle t^{i+1}_{b_{12}} \rangle$.

Case 7. $(t^{i+1}_{b_1} = \sim t^i_{b_{11}})$. This case is analogous to the $(t^i_{b_1} = \langle t^{i+1}_{b_{11}} \rangle)$ case.

Case 8. $(t^i_{b_1} = !t^i_{b_{11}})$. This case is analogous to the $(t^i_{b_1} = \langle t^{i+1}_{b_{11}} \rangle)$ case.

Case 9. $(t^i_{b_1} = t^i_{b_{11}}[x_1 := w_1]...[x_2 := w_2])$.
We proceed by cases on $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t^i_{b_1} \longrightarrow t^i_{b_2}$. The only case is (inj-subst) in which $\mathscr{U}; \mathscr{V}; \mathscr{X} \vdash t^i_{b_1} \longrightarrow^{\mathrm{x}} t^i_{b_2}$.
It is provable that $t^i_{a_1} = t^i_{a_2} = U(t^i_{b_1}) = U(t^i_{b_2})$. We have $t^i_{a_1} \longrightarrow^* t^i_{a_2}$ and $t^i_{a_2} \simeq t^i_{b_2}$.

□

D.2.7 Soundness and Completeness

Theorem 352 (Soundness of Suspended MetaML w.r.t. (Substitutional) MetaML)**.** *If $t^0_{a_1} \simeq t^0_{b_1}$, $\emptyset;\emptyset \vdash t^0_{b_1}$ wb, $FV(t^0_{b_1}) = \emptyset$ and $t^0_{a_1} \longrightarrow^{0*} v^0_{a_2}$ in Substitutional MetaML, then $\triangleright\, t^0_{b_1} \longrightarrow^* v^0_{b_2}$ in Suspended MetaML and $v^0_{a_2} \simeq v^0_{b_2}$.*

Proof. We proceed by induction on the length of $t^0_{a_1} \longrightarrow^{0*} v^0_{a_2}$.

Case 1. (0). Let $t^0_{a_1} = v^0_{a_2}$. By Lemma 349, $\emptyset;\emptyset;\textsc{Var}(t^0_{b_1}) \vdash t^0_{b_1} \longrightarrow^{0*} v^0_{b_2}$ and $v^0_{a_2} \simeq v^0_{b_2}$. We have $\triangleright\, t^0_{b_1} \longrightarrow^* v^0_{b_2}$.

Case 2. $(n+1)$. Let $t^0_{a_1} \longrightarrow^0 t^0_{a_2} \longrightarrow^{0(n)} v^0_{a_2}$.
We have $\emptyset;\emptyset \vdash t^0_{b_1}$ *wb* and $FV(t^0_{b_1}) = \emptyset$.
No matter whether $t^0_{b_1}$ is in substitution normal form, by Lemma 350, we have $\emptyset;\emptyset;\textsc{Var}(t^0_{b_1}) \vdash t^0_{b_1} \longrightarrow^{0*} t^0_{b_2}$ and $t^0_{a_2} \simeq t^0_{b_2}$. Then $\triangleright\, t^0_{b_1} \longrightarrow^* t^0_{b_2}$.
By Lemma 340, $\emptyset;\emptyset \vdash t^0_{b_2}$ *wb*. By Lemma 336, $FV(t^0_{b_2}) = \emptyset$.
By the induction hypothesis, $\triangleright\, t^0_{b_2} \longrightarrow^* v^0_{b_2}$ and $v^0_{a_2} \simeq v^0_{b_2}$.
We have $\triangleright\, t^0_{b_1} \longrightarrow^* v^0_{b_2}$.

□

Theorem 353 (Completeness of Suspended MetaML w.r.t. (Substitutional) MetaML)**.** *If $t^0_{a_1} \simeq t^0_{b_1}$, $\emptyset;\emptyset \vdash t^0_{b_1}$ wb, $FV(t^0_{b_1}) = \emptyset$ and $\triangleright\, t^0_{b_1} \longrightarrow^* v^0_{b_2}$ in Suspended MetaML, then $t^0_{a_1} \longrightarrow^{0*} v^0_{a_2}$ in Substitutional MetaML and $v^0_{a_2} \simeq v^0_{b_2}$.*

Proof. We proceed by induction on the length of $\triangleright\, t^0_{b_1} \longrightarrow^* v^0_{b_2}$.

Case 1. (0). Let $t^0_{b_1} = v^0_{b_2}$. By Lemma 348, $t^0_{a_1} \in \textsc{Value}^0_{\mathrm{sub}}$. Let $v^0_{a_2} = t^0_{a_1}$. We have $t^0_{a_1} \longrightarrow^{0*} v^0_{a_2}$ and $v^0_{a_2} \simeq v^0_{b_2}$.

Case 2. $(n+1)$. Let $\triangleright\, t^0_{b_1} \longrightarrow t^0_{b_2} \longrightarrow^{(n)} v^0_{b_2}$.
We have $\emptyset;\emptyset;\textsc{Var}(t^0_{b_1}) \vdash t^0_{b_1} \longrightarrow^0 t^0_{b_2}$, $\emptyset;\emptyset \vdash t^0_{b_1}$ *wb* and $FV(t^0_{b_1}) = \emptyset$.
No matter whether $t^0_{b_1}$ is in substitution normal form, by Lemma 351, $t^0_{a_1} \longrightarrow^{0*} t^0_{a_2}$ and $t^0_{a_2} \simeq t^0_{b_2}$.
By Lemma 340, $\emptyset;\emptyset \vdash t^0_{b_2}$ *wb*. By Lemma 336, $FV(t^0_{b_2}) = \emptyset$.
By the induction hypothesis, $t^0_{a_2} \longrightarrow^{0*} v^0_{a_2}$ and $v^0_{a_2} \simeq v^0_{b_2}$.
We have $t^0_{a_1} \longrightarrow^{0*} v^0_{a_2}$.

□

Theorem 354 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{MetaML}}$*,* $eval_{\text{MetaML:SubSOS}}(t)$ *is Kleene equal to* $eval_{\text{MetaML:SusSOS}}(t)$*.*

Proof. We first show that if $eval_{\text{MetaML:SubSOS}}(t) = a$ where $a \in \text{ANS}_{\text{MetaML}}$, then $eval_{\text{MetaML:SusSOS}}(t) = a$.

Case 1. If $eval_{\text{MetaML:SubSOS}}(t) = \texttt{function}$, then $t \longrightarrow^{0*} \lambda x.t'^{0}$ in Substitutional MetaML. Observe that $t \simeq t$. By Theorem 352, $\triangleright\, t \longrightarrow^{*} v$ in Suspended MetaML and $\lambda x.t'^{0} \simeq v$. Then $v = (\underline{\lambda} x_0.t''^{0})\overline{[x_i := w_i]}$ and $U((\underline{\lambda} x_0.t''^{0})\overline{[x_i := w_i]}) = \lambda x.t'^{0}$. We have $eval_{\text{MetaML:SusSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:SubSOS}}(t) = \texttt{code}$, then $t \longrightarrow^{0*} \langle v^1 \rangle$ in Substitutional MetaML. Observe that $t \simeq t$. By Theorem 352, $\triangleright\, t \longrightarrow^{*} v'$ in Suspended MetaML and $\langle v^1 \rangle \simeq v'$. Then $v' = \langle v''^1 \rangle$ and $U(\langle v''^1 \rangle) = \langle v^1 \rangle$. We have $eval_{\text{MetaML:SusSOS}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:SubSOS}}(t) = n$, then $t \longrightarrow^{0*} n$ in Substitutional MetaML. Observe that $t \simeq t$. By Theorem 352, $\triangleright\, t \longrightarrow^{*} v$ in Suspended MetaML and $n \simeq v$. Then $v = n$. We have $eval_{\text{MetaML:SusSOS}}(t) = n$.

We then show that if $eval_{\text{MetaML:SusSOS}}(t) = a$ where $a \in \text{ANS}_{\text{MetaML}}$, then $eval_{\text{MetaML:SubSOS}}(t) = a$.

Case 1. If $eval_{\text{MetaML:SusSOS}}(t) = \texttt{function}$, then $\triangleright\, t \longrightarrow^{*} (\underline{\lambda} x.t'^{0})\overline{[x_i := w_i]}$ in Suspended MetaML. Observe that $t \simeq t$. By Theorem 353, $t \longrightarrow^{0*} v$ in Substitutional MetaML and $v \simeq (\underline{\lambda} x.t'^{0})\overline{[x_i := w_i]}$. Then $v = U((\underline{\lambda} x.t'^{0})\overline{[x_i := w_i]}) = \lambda x.U(t'^{0})\overline{[U(w_i)/x_i]}$. We have $eval_{\text{MetaML:SubSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:SusSOS}}(t) = \texttt{code}$, then $\triangleright\, t \longrightarrow^{*} \langle v^1 \rangle$ in Suspended MetaML. Observe that $t \simeq t$. By Theorem 353, $t \longrightarrow^{0*} v'$ in Substitutional MetaML and $v' \simeq \langle v^1 \rangle$. Then $v' = \langle v''^1 \rangle$ and $v' = U(\langle v^1 \rangle)$. We have $eval_{\text{MetaML:SubSOS}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:SusSOS}}(t) = n$, then $\triangleright\, t \longrightarrow^{*} n$ in Suspended MetaML. Observe that $t \simeq t$. By Theorem 353, $t \longrightarrow^{0*} v$ in Substitutional MetaML and $v \simeq n$. Then $v = n$. We have $eval_{\text{MetaML:SubSOS}}(t) = n$.

We observe that $eval_{\text{MetaML:SubSOS}}(t)$ is undefined if and only if $eval_{\text{MetaML:SusSOS}}(t)$ is undefined. Therefore, $eval_{\text{MetaML:SubSOS}}(t)$ is Kleene equal to $eval_{\text{MetaML:SusSOS}}(t)$. □

D.3 Equivalence of MetaML and Environmental MetaML

We demonstrate the equivalence of the substitutional structural operational semantics of MetaML and the structural operational semantics of Environmental MetaML. We use subscripts "$_{\text{sub}}$" and "$_{\text{env}}$" to differentiate the syntax of (Substitutional) MetaML from the syntax of Environmental MetaML.

D.3.1 Well-boundness Judgement

Definition 355 (Well-boundness Judgement)**.** Let the well-boundness judgement $\vdash\ wb$ be a ternary relation on the power set of variables, the power set of variables and the set of configurations.

$$\vdash\ wb\ \subseteq\ \mathscr{P}(\textsc{Var}) \times \mathscr{P}(\textsc{Var}) \times \textsc{Conf}$$

$$\frac{}{\mathscr{U};\mathscr{V} \vdash x\ wb}\ \text{where } x \in \mathscr{U}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash c_1\ wb \quad \mathscr{U};\mathscr{V} \vdash c_2\ wb}{\mathscr{U};\mathscr{V} \vdash c_1\ c_2\ wb}$$

$$\frac{\mathscr{U} \cup \{x\};\mathscr{V} \vdash c\ wb}{\mathscr{U};\mathscr{V} \vdash \lambda x.c^0\ wb}\ \text{where } x \notin \mathscr{V}$$

$$\frac{\mathscr{U} \cup \{x\};\mathscr{V} \vdash c\ wb}{\mathscr{U};\mathscr{V} \vdash \lambda x.v^{i+1}\ wb}\ \text{where } x \notin \mathscr{V}$$

$$\frac{\mathscr{U} \cup \{x\};\mathscr{V} \cup \{x\} \vdash c\ wb}{\mathscr{U};\mathscr{V} \vdash \lambda x.c^{i+1}\ wb}\ \text{where } c^{i+1} \notin \textsc{Value}^{i+1} \text{ and } x \notin \mathscr{V}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash \langle t \rangle\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash {\sim} t\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash !t\ wb}$$

$$\frac{}{\mathscr{U};\mathscr{V} \vdash n\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t_1\ wb \quad \mathscr{U};\mathscr{V} \vdash t_2\ wb}{\mathscr{U};\mathscr{V} \vdash t_1 + t_2\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash t\ wb}{\mathscr{U};\mathscr{V} \vdash \blacktriangleleft t,\ \varepsilon \blacktriangleright\ wb}$$

$$\frac{\mathscr{U};\mathscr{V} \vdash \rho_m(x_{m_k})\ wb \quad \mathscr{U} \cup \{\overline{x_{m_k}}\};\mathscr{V} \vdash \blacktriangleleft t,\ (\overline{\rho_{i1}^{m-1}};\varepsilon) \blacktriangleright\ wb}{\mathscr{U};\mathscr{V} \vdash \blacktriangleleft t,\ (\overline{\rho_{i1}^{m}};\varepsilon) \blacktriangleright\ wb}$$

where $\textsc{Var}(\blacktriangleleft t,\ \overline{\rho_{i1}^{m}} \blacktriangleright) \subseteq \mathrm{dom}(\rho_i)$ for any ρ_i,
$\rho_i(y_j) = y_j$ for any ρ_i and $y_j \in \mathscr{V}$,
and $x_{m_k} \in FV(\blacktriangleleft t,\ (\overline{\rho_{i1}^{m-1}};\varepsilon) \blacktriangleright)$.

$$\frac{\mathscr{U};\mathscr{V} \vdash \blacktriangleleft t,\ \overline{\rho_i} \blacktriangleright\ wb}{\mathscr{U};\mathscr{V} \vdash \triangleleft t,\ \overline{\rho_i} \triangleright\ wb}$$

Lemma 356. *If $\mathscr{U};\mathscr{V} \vdash c\ wb$, then $FV(c) \subseteq \mathscr{U}$.*

Proof. By structural induction on $\mathscr{U};\mathscr{V} \vdash c\ wb$. □

Lemma 357. *If $\mathscr{V};\mathscr{V} \vdash c_0^i\ wb$, $\textsc{Var}(c_0^i) \subseteq \mathscr{X}$, $\mathscr{V} \subseteq \mathscr{X}$ and $\mathscr{V};\mathscr{X} \vdash c_0^i \longrightarrow^{i*} c_0'^i$, then $\mathscr{V};\mathscr{V} \vdash c_0'^i\ wb$.*

Proof. We proceed by induction on the structure of $\mathscr{V};\mathscr{V} \vdash c_0^i\ wb$ and the structure of $\mathscr{V};\mathscr{X} \vdash c_0^i \longrightarrow^{i*} c_0'^i$.

Case 1. $\dfrac{}{\mathscr{V};\mathscr{V} \vdash x\ wb}$ where $x \in \mathscr{V}$.

This case is vacuous because $\mathscr{V};\mathscr{X} \vdash x \not\longrightarrow$.

Case 2. $\dfrac{\mathscr{V};\mathscr{V} \vdash c_1\ wb \quad \mathscr{V};\mathscr{V} \vdash c_2\ wb}{\mathscr{V};\mathscr{V} \vdash c_1\ c_2\ wb}$.

In this case, $c_0 = c_1\ c_2$.

We proceed by cases on $\mathscr{V};\mathscr{X} \vdash c_0^i \longrightarrow^{i*} c_0'^i$.

Case i. $\dfrac{\mathscr{V};\mathscr{X} \vdash c_1 \longrightarrow^i c_1'}{\mathscr{V};\mathscr{X} \vdash c_1\ c_2 \longrightarrow^i c_1'\ c_2}$.

By induction hypothesis, $\mathscr{V};\mathscr{V} \vdash c_1'\ wb$. Then $\mathscr{V};\mathscr{V} \vdash c_1'\ c_2\ wb$.

Case ii. $\dfrac{\mathscr{V};\mathscr{X} \vdash c_2 \longrightarrow^i c_2'}{\mathscr{V};\mathscr{X} \vdash v_1\ c_2 \longrightarrow^i v_1\ c_2'}$.

This case is analogous to the previous case.

Case iii. $\dfrac{}{\mathscr{V};\mathscr{X} \vdash \triangleleft\ \underline{\lambda} x_0.t^0,\ (\rho_1;...\rho_n;\varepsilon)\ \triangleright\ v^0 \longrightarrow^0 \blacktriangleleft\ t^0,\ (\rho_1[x_0 \mapsto v^0];...,\rho_n;\varepsilon\ \blacktriangleright}$.

Then $c_1 = \triangleleft\ \underline{\lambda} x_0.t^0,\ (\rho_1;...\rho_n;\varepsilon)\ \triangleright$ and $c_2 = v^0$.

Given $\mathscr{V};\mathscr{V} \vdash \triangleleft\ \underline{\lambda} x_0.t^0,\ (\rho_1;...\rho_n;\varepsilon)\ \triangleright\ wb$, the following holds:

(1) $\mathscr{V};\mathscr{V} \vdash \rho_n(x_{n_i})\ wb$,

(2) $\mathscr{V} \cup \{\overline{x_{n_i}}\};\mathscr{V} \vdash \rho_{n-1}(x_{(n-1)_i})\ wb$,

...

(n) $\mathscr{V} \cup \{\overline{x_{n_i}}, \overline{x_{(n-1)_i}}, ..., \overline{x_{2_i}}\};\mathscr{V} \vdash \rho_1(x_{1_i})\ wb$,

(n+1) $\mathscr{V} \cup \{\overline{x_{n_i}}, \overline{x_{(n-1)_i}}, ..., \overline{x_{2_i}}, \overline{x_{1_i}}, x_0\};\mathscr{V} \vdash t^0\ wb$, and

(n+2) $x_0 \notin \mathscr{V}$.

We also have

(n+3) $\mathscr{V};\mathscr{V} \vdash v^0\ wb$, which implies $FV(v^0) \subseteq \mathscr{V}$.

(n+4) $\rho_i(y_j) = y_j$ for any $y_j \in \mathscr{V}$.

To show $\mathscr{V};\mathscr{V} \vdash \blacktriangleleft\ t^0,\ (\rho_1[x_0 \mapsto v^0];...,\rho_n;\varepsilon\ \blacktriangleright\ wb$, it is sufficient to show that

(1) $\mathscr{U};\mathscr{V} \vdash \rho_n(x_{n_i})\ wb$ and $\mathscr{U};\mathscr{V} \vdash FV(v_0)\ wb$,

(2) $\mathscr{U} \cup \{\overline{x_{n_i}}\} \cup FV(v_0);\mathscr{V} \vdash \rho_{n-1}(x_{(n-1)_i})\ wb$ and $\mathscr{U} \cup \{\overline{x_{n_i}}\} \cup FV(v_0);\mathscr{V} \vdash FV(v_0)\ wb$,

...

(n) $\mathscr{U} \cup \{\overline{x_{n_i}}, \overline{x_{(n-1)_i}}, ..., \overline{x_{2_i}}\} \cup FV(v_0);\mathscr{V} \vdash \rho_1(x_{1_i})\ wb$ and $\mathscr{U} \cup \{\overline{x_{n_i}}, \overline{x_{(n-1)_i}}, ..., \overline{x_{2_i}}\} \cup FV(v_0);\mathscr{V} \vdash v_0\ wb$,

(n+2) $\mathscr{U} \cup \{\overline{x_{n_i}}, \overline{x_{(n-1)_i}}, ..., \overline{x_{2_i}}, \overline{x_{1_i}}, x_0\} \cup FV(v_0);\mathscr{V} \vdash t^0\ wb$,

(n+3) $x_0 \notin \mathscr{V}$, and

(n+4) $\rho_i(y_j) = y_j$ for any $y_j \in \mathscr{V}$.

All of (1)-(n+4) above hold.

Case 3. $\dfrac{\mathscr{V} \cup \{x\}; \mathscr{V} \vdash c\ wb}{\mathscr{V}; \mathscr{V} \vdash \underline{\lambda} x.c\ wb}$ where $c \in \textsc{Conf}^0$ or $c \in \textsc{Value}^{i+1}$, and $x \notin \mathscr{V}$.

This case is vacuous because $\mathscr{V}; \mathscr{X} \vdash \underline{\lambda} x.c \not\longrightarrow$.

Case 4. $\dfrac{\mathscr{V} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash c^{i+1}\ wb}{\mathscr{V}; \mathscr{V} \vdash \lambda x.c^{i+1}\ wb}$ where $c^{i+1} \notin \textsc{Value}^{i+1}$ and $x \notin \mathscr{V}$.

In this case, $c_0 = \lambda x.c$. We proceed by cases on $\mathscr{V}; \mathscr{X} \vdash c_0 \longrightarrow c_0'$.

Case i. $\dfrac{\mathscr{V} \cup \{x\}; \mathscr{X} \vdash c \longrightarrow^{i+1} c'}{\mathscr{V}; \mathscr{X} \vdash \lambda x.c \longrightarrow^{i+1} \lambda x.c'}$

To show $\mathscr{V}; \mathscr{V} \vdash \lambda x.c'\ wb$, it is sufficient to show that

- $x \notin \mathscr{V}$, and
- $\mathscr{V} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash c'\ wb$.

Given $\mathscr{V}; \mathscr{V} \vdash \lambda x.c\ wb$, we have $\mathscr{V} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash c\ wb$ and $x \notin \mathscr{V}$. By the induction hypothesis, we have $\mathscr{V} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash v'\ wb$.

Case 5. $\dfrac{\mathscr{V}; \mathscr{V} \vdash c\ wb}{\mathscr{V}; \mathscr{V} \vdash \langle c \rangle\ wb}$.

In this case, $c_0 = \langle t \rangle$. We proceed by cases on $\mathscr{V}; \mathscr{X} \vdash c_0 \longrightarrow c_0'$.

Case i. $\dfrac{\mathscr{V}; \mathscr{X} \vdash c \longrightarrow^{i+1} c'}{\mathscr{V}; \mathscr{X} \vdash \langle c \rangle \longrightarrow^{i} \langle c' \rangle}$

By induction hypothesis, $\mathscr{V}; \mathscr{V} \vdash c'\ wb$. Then $\mathscr{V}; \mathscr{V} \vdash \langle c' \rangle\ wb$.

Case 6. $\dfrac{\mathscr{V}; \mathscr{V} \vdash c\ wb}{\mathscr{V}; \mathscr{V} \vdash {\sim}c\ wb}$.

This case is analogous to the previous case.

Case 7. $\dfrac{\mathscr{V}; \mathscr{V} \vdash c\ wb}{\mathscr{V}; \mathscr{V} \vdash !c\ wb}$.

This case is analogous to the previous case.

Case 8. $\dfrac{\mathscr{V}; \mathscr{V} \vdash c_1\ wb \quad \mathscr{V}; \mathscr{V} \vdash c_2\ wb}{\mathscr{V}; \mathscr{V} \vdash c_1 + c_2\ wb}$.

This case is analogous to the $\mathscr{V}; \mathscr{V} \vdash c_1\ c_2\ wb$ case.

Case 9. $\dfrac{}{\mathscr{V}; \mathscr{V} \vdash n\ wb}$.

This case is vacuous because $n \not\longrightarrow$.

Case 10. $\dfrac{\mathscr{V}; \mathscr{V} \vdash t\ wb}{\mathscr{V}; \mathscr{V} \vdash \blacklozenge\ t,\ \varepsilon\ \blacklozenge\ wb}$

In this case, $c_0 = \blacklozenge\ t,\ \varepsilon\ \blacklozenge$. We proceed by cases on $\mathscr{V}; \mathscr{X} \vdash c_0 \longrightarrow c_0'$.

Case i. $\dfrac{}{\mathscr{V}; \mathscr{X} \vdash \blacklozenge\ w,\ \varepsilon\ \blacklozenge \longrightarrow^{i} w}$ (den-env)

In this case, $t = w$. Given $\mathscr{V}; \mathscr{V} \vdash \blacklozenge\ w,\ \varepsilon\ \blacklozenge\ wb$, we get $\mathscr{V}; \mathscr{V} \vdash w\ wb$.

Case ii. The other (*-env) cases are trivial.

Case 11. $\dfrac{\mathscr{V};\mathscr{V} \vdash \rho_m(x_{m_k})\ wb \quad \mathscr{V} \cup \{\overline{x_{m_k}}\};\mathscr{V} \vdash \blacklozenge\ t,\ (\overline{\rho_{i1}^{m-1}};\varepsilon)\ \blacklozenge\ wb}{\mathscr{V};\mathscr{V} \vdash \blacklozenge\ t,\ (\overline{\rho_{i1}^{m}};\varepsilon)\ \blacklozenge\ wb}$ where $\textsc{Var}(\blacklozenge\ t,\ \overline{\rho_{i1}^{m}}\ \blacklozenge) \subseteq \mathrm{dom}(\rho_i)$ for any ρ_i, $\rho_i(y_j) = y_j$ for any ρ_i and $y_j \in \mathscr{V}$, and $x_{m_k} \in FV(\blacklozenge\ t,\ (\overline{\rho_{i1}^{m-1}};\varepsilon)\ \blacklozenge)$.

In this case, $c_0 = \blacklozenge\ t,\ (\overline{\rho_{i1}^{m}};\varepsilon)\ \blacklozenge$.
We proceed by cases on $\mathscr{V};\mathscr{X} \vdash c_0 \longrightarrow c_0'$.

Case i. $\dfrac{}{\mathscr{V};\mathscr{X} \vdash \blacklozenge\ \lambda x.t^{i+1},\ (\overline{\rho_{i1}^{m}};\varepsilon)\ \blacklozenge \longrightarrow^{i+1} \lambda x_N.\blacklozenge\ t^{i+1},\ (\rho_1[x \mapsto x_N][x_N \mapsto x_N];\overline{\rho_i[x_N \mapsto x_N]}_2^m;\varepsilon)\ \blacklozenge}$ where $x_N \notin \mathscr{X}$

Given $\mathscr{V};\mathscr{V} \vdash \blacklozenge\ \lambda x.t^{i+1},\ (\overline{\rho_{i1}^{m}};\varepsilon)\ \blacklozenge\ wb$, the following holds:

(1) $\mathscr{V};\mathscr{V} \vdash \rho_m(x_{m_k})\ wb$,

(2) $\mathscr{V} \cup \{\overline{x_{m_k}}\};\mathscr{V} \vdash \rho_{m-1}(x_{(m-1)_k})\ wb$,

...

(m) $\mathscr{V} \cup \{\overline{x_{m_k}}, \overline{x_{(m-1)_k}}, ..., \overline{x_{2_k}}\};\mathscr{V} \vdash \rho_1(x_{1_k})\ wb$,

(m+1) $\mathscr{V} \cup \{\overline{x_{m_k}}, \overline{x_{(m-1)_k}}, ..., \overline{x_{2_k}}, \overline{x_{1_k}}, x\};\mathscr{V} \vdash t^{i+1}\ wb$,

(m+2) $x \notin \mathscr{V}$, and

(m+3) $\rho_i(x_j) = x_j$ for any $x_j \in \mathscr{V}$.

To show $\mathscr{V};\mathscr{V} \vdash \lambda x_N.\blacklozenge\ t^{i+1},\ (\rho_1[x \mapsto x_N][x_N \mapsto x_N];\overline{\rho_i[x_N \mapsto x_N]}_2^m;\varepsilon)\ \blacklozenge\ wb$, it is sufficient to show that

- $x_N \notin \mathscr{V}$, and
- $\mathscr{V} \cup \{x_N\};\mathscr{V} \cup \{x_N\} \vdash \blacklozenge\ t^{i+1},\ (\rho_1[x \mapsto x_N][x_N \mapsto x_N];\overline{\rho_i[x_N \mapsto x_N]}_2^m;\varepsilon)\ \blacklozenge\ wb$, which can be shown by
  - **(1)** $\mathscr{V} \cup \{x_N\};\mathscr{V} \cup \{x_N\} \vdash \rho_m(x_{m_k})\ wb$ and $\mathscr{V} \cup \{x_N\};\mathscr{V} \cup \{x_N\} \vdash x_N\ wb$,
  - **(2)** $\mathscr{V} \cup \{x_N\} \cup \{\overline{x_{m_k}}\};\mathscr{V} \cup \{x_N\} \vdash \rho_{m-1}(x_{(m-1)_k})\ wb$ and $\mathscr{V} \cup \{x_N\} \cup \{\overline{x_{m_k}}\};\mathscr{V} \cup \{x_N\} \vdash x_N\ wb$,
  - **...**
  - **(m)** $\mathscr{V} \cup \{x_N\} \cup \{\overline{x_{m_k}}, \overline{x_{(m-1)_k}}, ..., \overline{x_{2_k}}\};\mathscr{V} \cup \{x_N\} \vdash \rho_1(x_{1_k})\ wb$ and $\mathscr{V} \cup \{x_N\} \cup \{\overline{x_{m_k}}, \overline{x_{(m-1)_k}}, ..., \overline{x_{2_k}}\};\mathscr{V} \cup \{x_N\} \vdash x_N\ wb$,
  - **(m+1)** $\mathscr{V} \cup \{x_N\} \cup \{\overline{x_{m_k}}, \overline{x_{(m-1)_k}}, ..., \overline{x_{2_k}}, \overline{x_{1_k}}, x\};\mathscr{V} \cup \{x_N\} \vdash t^{i+1}\ wb$,
  - **(m+2)** $x \notin \mathscr{V}$, and
  - **(m+3)** $\rho_i(x_j) = x_j$ for any $x_j \in \mathscr{V} \cup \{x_N\}$.

All above hold.

Case ii. $\dfrac{}{\mathscr{V};\mathscr{X} \vdash \blacklozenge\ x,\ (\overline{\rho_{i1}^{m}};\varepsilon)\ \blacklozenge \longrightarrow^{i} \blacklozenge\ \rho_1(x),\ (\overline{\rho_{i2}^{m}};\varepsilon)\ \blacklozenge}$.
Given n $\mathscr{V};\mathscr{V} \vdash \blacklozenge\ \lambda x.t^{i+1},\ (\overline{\rho_{i1}^{m}};\varepsilon)\ \blacklozenge\ wb$, the following holds:

(1) $\mathscr{V};\mathscr{V} \vdash \rho_m(x_{m_k})\ wb$,

(2) $\mathscr{V} \cup \{\overline{x_{m_k}}\};\mathscr{V} \vdash \rho_{m-1}(x_{(m-1)_k})\ wb$,

...

(m) $\mathscr{V} \cup \{\overline{x_{m_k}}, \overline{x_{(m-1)_k}}, ..., \overline{x_{2_k}}\}; \mathscr{V} \vdash \rho_1(x_{1_k})\ wb$,

(m+1) $\mathscr{V} \cup \{\overline{x_{m_k}}, \overline{x_{(m-1)_k}}, ..., \overline{x_{2_k}}, \overline{x_{1_k}}\}; \mathscr{V} \vdash x\ wb$, which means $x \in \mathscr{V} \cup \{\overline{x_{m_k}}, \overline{x_{(m-1)_k}}, ..., \overline{x_{2_k}}, \overline{x_{1_k}}\}$, and

(m+2) $\rho_i(x_j) = x_j$ for any $x_j \in \mathscr{V}$.

To show $\mathscr{V}; \mathscr{V} \vdash \text{◖}\ \rho_1(x),\ (\overline{\rho_{i2}}^m; \varepsilon)\ \text{◗}\ wb$, it is sufficient to show that

(1) $\mathscr{V}; \mathscr{V} \vdash \rho_m(x_{m_k})\ wb$,

(2) $\mathscr{V} \cup \{\overline{x_{m_k}}\}; \mathscr{V} \vdash \rho_{m-1}(x_{(m-1)_k})\ wb$,

...

(m) $\mathscr{V} \cup \{\overline{x_{m_k}}, \overline{x_{(m-1)_k}}, ..., \overline{x_{2_k}}\}; \mathscr{V} \vdash \rho_1(x)\ wb$,

(m+1) $\rho_i(x_j) = x_j$ for any $x_j \in \mathscr{V}$.

All above hold.

Case iii. The other (*-env) cases are trivial.

□

Remark 358. The above proof uses the following two properties.

1. If $\mathscr{U}; \mathscr{V} \vdash t\ wb$, then $\mathscr{U} \cup \mathscr{W}; \mathscr{V} \vdash t\ wb$.
2. If $\mathscr{U}; \mathscr{V} \vdash t\ wb$ and $x_N \notin \mathscr{U} \cup \mathscr{V} \cup Var(t)$, then $\mathscr{U} \cup \{x_N\}; \mathscr{V} \cup \{x_N\} \vdash t\ wb$.

D.3.2 Unload Function

Definition 359 (Unload Function)**.** Let $i \in \mathbb{N}$. Define the unload function U to be a total function from the union of the set of runtime terms and the set of configurations in Environmental MetaML to the set of terms in (Substitutional) MetaML.

$$U \;:\; \mathscr{P}(\textsc{Var}) \times (\textsc{RTerm}^i_{\text{env}} \cup \textsc{Conf}^i_{\text{env}}) \to \textsc{Term}^i_{\text{sub}}$$

$$\begin{array}{rcl}
U(\mathscr{Z} \mid x) & = & x \\
U(\mathscr{Z} \mid n) & = & n \\
\\
U(\mathscr{Z} \mid \lambda x.c^{i+1}) & = & \lambda x.U(\mathscr{Z} \mid c^{i+1}) \\
U(\mathscr{Z} \mid c^i_1\ c^i_2) & = & U(\mathscr{Z} \mid c^i_1)\ U(\mathscr{Z} \mid c^i_2) \\
U(\langle \mathscr{Z} \mid c^{i+1} \rangle) & = & \langle U(\mathscr{Z} \mid c^{i+1}) \rangle \\
U(\sim \mathscr{Z} \mid c^{i+1}) & = & \sim U(\mathscr{Z} \mid c^{i+1}) \\
U(\mathscr{Z} \mid !c^i) & = & !U(\mathscr{Z} \mid c^i) \\
U(\mathscr{Z} \mid c^i_1 + c^i_2) & = & U(\mathscr{Z} \mid c^i_1) + U(\mathscr{Z} \mid c^i_2) \\
\\
U(\lambda x.t^i) & = & \lambda x.U(t^i) \\
U(t^i_1\ t^i_2) & = & U(t^i_1)\ U(t^i_2) \\
U(\langle t^{i+1} \rangle) & = & \langle U(t^{i+1}) \rangle \\
U(\sim t^{i+1}) & = & \sim U(t^{i+1}) \\
U(!t^i) & = & !U(t^i) \\
U(t^i_1 + t^i_2) & = & U(t^i_1) + U(t^i_2) \\
\\
U(\mathscr{Z} \mid \blacktriangleleft\ t^i,\ \varepsilon\ \blacktriangleright) & = & U(\mathscr{Z} \mid t^i) \\
U(\mathscr{Z} \mid \blacktriangleleft\ \lambda x.t^{i+1},\ (\rho_1; \rho_2^*)\ \blacktriangleright) & = & U(\mathscr{Z} \cup \{x_N\} \mid \lambda x_N.\blacktriangleleft\ t^{i+1},\ (\rho_1[x \mapsto x_N]; (\rho_2[x_N \mapsto x_N])^*\ \blacktriangleright) \\
 & & \text{where } x_N \notin \mathscr{Z} \\
U(\mathscr{Z} \mid \blacktriangleleft\ t^i,\ (\rho_1; \rho_2^*)\ \blacktriangleright) & = & U(\mathscr{Z} \mid \blacktriangleleft\ t^i\overline{[w_{1i}/x_{1i}]},\ \rho_2^*\ \blacktriangleright) \\
U(\mathscr{Z} \mid \triangleleft\ \lambda x.t^0,\ \rho^*\ \triangleright) & = & U(\mathscr{Z} \mid \blacktriangleleft\ \lambda x.t^0,\ \rho^*\ \blacktriangleright)
\end{array}$$

Remark 360. $U(\mathscr{Z} \mid t)$ may be omitted to $U(t)$ if $\mathscr{Z}$ is clear by the context.

Proposition 361. $U(\mathscr{Z} \mid t^i_{\text{s}}) = t^i_{\text{s}}$.

Proposition 362. $U(\mathscr{Z} \mid t\overline{[w_i/x_i]}) = U(\mathscr{Z} \mid t)\overline{[U(\mathscr{Z} \mid w_i)/x_i]}$.

D.3.3 Bisimulation Relation

Definition 363 (Bisimulation Relation)**.** Define the bisimulation relation $\simeq$ to be a binary relation between the set of terms in (Substitutional) MetaML and the set of configurations in Environmental MetaML.

$$\simeq\ \subseteq\ \textsc{Term}^i_{\text{sub}} \times \textsc{Conf}^i_{\text{env}}$$

$t^i \simeq c^i$ if and only if $t^i = U(\mathscr{Z} \mid c^i)$ where $\textsc{Var}(c^i) \subseteq \mathscr{Z}$

D.3.4 Closure Descendant Relation

Definition 364 (Closure Descendant Relation)**.** For any $c_1, c_2 \in \textsc{Conf}_{\text{env}}$, $c_1 \prec^{\text{x}} c_2$ if and only if $c_2 \longrightarrow c_1$ is an instance of rule (*-env). We call $\prec^{\text{x}}$ the closure descendant relation.

Proposition 365 (Well-foundedness of Closure Descendant Relation)**.** *The closure descendant relation* $\prec^{\mathrm{x}}$ *is well-founded.*

The proof is analogous to the proof of Lemma 322.

D.3.5 Canonisation

Lemma 366 (Canonisation of (Substitutional) MetaML)**.** *If* $t^i_{a_1} \simeq v^i_{b_1}$*, then* $t^i_{a_1} \in \mathrm{VALUE}^i_{\mathrm{sub}}$.

Proof. We proceed by structural induction on $v^i_{b_1} \in \mathrm{VALUE}^i_{\mathrm{env}}$.

Case 1. $(v^{i+1}_{b_1} = x)$. Then $t^{i+1}_{a_1} = U(x) = x$ and $x \in \mathrm{VALUE}^{i+1}_{\mathrm{sub}}$.

Case 2. $(v^{i+1}_{b_1} = v^{i+1}_{b_{11}}\ v^{i+1}_{b_{12}})$. Then $t^{i+1}_{a_1} = U(v^{i+1}_{b_{11}}\ v^{i+1}_{b_{12}}) = U(v^{i+1}_{b_{11}})\ U(v^{i+1}_{b_{12}})$. Let $t^{i+1}_{a_1} = t^{i+1}_{a_{11}}\ t^{i+1}_{a_{12}}$. By the induction hypothesis, $t^{i+1}_{a_{11}} \in \mathrm{VALUE}^{i+1}_{\mathrm{sub}}$ and $t^{i+1}_{a_{12}} \in \mathrm{VALUE}^{i+1}_{\mathrm{sub}}$. Then $t^{i+1}_{a_{11}}\ t^{i+1}_{a_{12}} \in \mathrm{VALUE}^{i+1}_{\mathrm{sub}}$.

Case 3. $(v^{i+1}_{b_1} = \lambda x.v^{i+1}_{b_{11}})$. Then $t^{i+1}_{a_1} = U(\lambda x.v^{i+1}_{b_{11}}) = \lambda x.U(v^{i+1}_{b_{11}})$. Let $t^{i+1}_{a_1} = \lambda x.t^{i+1}_{a_{11}}$. By the induction hypothesis, $t^{i+1}_{a_{11}} \in \mathrm{VALUE}^{i+1}_{\mathrm{sub}}$. Then $\lambda x.t^{i+1}_{a_{11}} \in \mathrm{VALUE}^{i+1}_{\mathrm{sub}}$.

Case 4. $(v^0_{b_1} = \langle\!|\ \lambda x.t^0_{b_{11}},\ (\rho_1, \rho_2, \ldots \rho_m)\ |\!\rangle)$. Then $t^0_{a_1} = U(\langle\!|\ \lambda x.t^0_{b_{11}},\ (\rho_1, \rho_2, \ldots \rho_m)\ |\!\rangle) = U(\langle\!\langle\ \lambda x.t^0_{b_{11}},\ (\rho_1, \rho_2, \ldots \rho_m)\ \rangle\!\rangle)$. We have $t^0_{a_1} = \lambda x.U(t^0_{b_{11}})\overline{[U(w_{1i})/x_{1i}][U(w_{2i})/x_{2i}]}\ldots\overline{[U(w_{mi})/x_{mi}]} \in \mathrm{VALUE}^0_{\mathrm{sub}}$.

Case 5. $(v^i_{b_1} = \langle v^{i+1}_{b_{11}} \rangle)$. Then $t^i_{a_1} = U(\langle v^{i+1}_{b_{11}} \rangle) = \langle U(v^{i+1}_{b_{11}}) \rangle$. Let $t^i_{a_1} = \langle t^{i+1}_{a_{11}} \rangle$. By the induction hypothesis, $t^{i+1}_{a_{11}} \in \mathrm{VALUE}^{i+1}_{\mathrm{sub}}$. Then $\langle t^{i+1}_{a_{11}} \rangle \in \mathrm{VALUE}^i_{\mathrm{sub}}$.

Case 6. $(v^{i+2}_{b_1} = {\sim} v^{i+1}_{b_{11}})$. This case is analogous to the $(v^i_{b_1} = \langle v^{i+1}_{b_{11}} \rangle)$ case.

Case 7. $(v^{i+1}_{b_1} = !v^{i+1}_{b_{11}})$. This case is analogous to the $(v^i_{b_1} = \langle v^{i+1}_{b_{11}} \rangle)$ case.

Case 8. $(v^i_{b_1} = n)$. Then $t^i_{a_1} = U(n) = n$ and $n \in \mathrm{VALUE}^i_{\mathrm{sub}}$.

Case 9. $(v^{i+1}_{b_1} = v^{i+1}_{b_{11}} + v^{i+1}_{b_{12}})$. This case is analogous to the $(v^{i+1}_{b_1} = v^{i+1}_{b_{11}}\ v^{i+1}_{b_{12}})$ case.

□

Lemma 367 (Canonisation of Environmental MetaML)**.** *If* $v^i_{a_1} \simeq c^i_{b_1}$*,* $\mathscr{V}; \mathscr{V} \vdash t^i_{b_1}$ *wb,* $FV(t^i_{b_1}) \subseteq \mathscr{V}$*,* $\mathrm{VAR}(t^i_{b_1}) \subseteq \mathscr{X}$ *and* $\mathscr{V} \subseteq \mathscr{X}$*, then* $\mathscr{V}; \mathscr{X} \vdash c^i_{b_1} \longrightarrow^{i*} v^i_{b_2}$ *and* $v^i_{a_1} \simeq v^i_{b_2}$.

Proof. We proceed by simultaneous induction on the structure of $v^i_{a_1} \in \mathrm{VALUE}^i_{\mathrm{sub}}$ and on the closure descendant relation $\prec^{\mathrm{x}} c^i_{b_1}$.

Case 1. $(v^{i+1}_{a_1} = x)$. We proceed by cases on $c^{i+1}_{b_1} \in \mathrm{CONF}^{i+1}_{\mathrm{env}}$.

Case i. $(c^{i+1}_{b_1} = x)$. Then $x \longrightarrow^* x$ and $x \simeq x$.

Case ii. $(c^{i+1}_{b_1} = \langle\!\langle\ x_0,\ \overline{\rho_i}^{+m}\ \rangle\!\rangle)$. We proceed by cases on m.

Case a. $(m = 1)$. Given $x \simeq \langle\!\langle\ x_0,\ \rho_1\ \rangle\!\rangle$, we have:

$$\begin{array}{lll} & x & \\ = & U(\llcorner\, x_0,\ \rho_1\ \lrcorner) & \\ = & U(\llcorner\, x_0\overline{[w_{1i}/x_{1i}]},\ \varepsilon\ \lrcorner) & \\ = & U(x_0\overline{[w_{1i}/x_{1i}]}) & \\ = & U(w_{1\mathrm{p}}) & \text{where } x_0 \equiv x_{1\mathrm{p}} \end{array}$$

We have $x \simeq w_{1\mathrm{p}}$.

Then by (var-env), we have:

$$\begin{array}{lll} \mathscr{V};\mathscr{X} \vdash & \llcorner\, x_0,\ \rho_1\ \lrcorner & \\ \longrightarrow & w_{1\mathrm{p}} & \text{where } \rho_1(x_0) = w_{1\mathrm{p}} \end{array}$$

Since $x \simeq w_{1\mathrm{p}}$, we have $w_{1\mathrm{p}} = x$. We have $x \in \textsc{Value}^{i+1}_{\mathrm{env}}$.

Case b. ($m > 1$). Given $x \simeq \llcorner\, x_0,\ (\rho_1;\rho_2;...;\rho_m)\ \lrcorner$, we have:

$$\begin{array}{lll} & x & \\ = & U(\llcorner\, x_0,\ (\rho_1;\rho_2;...;\rho_m)\ \lrcorner) & \\ = & U(\llcorner\, x_0\overline{[w_{1i}/x_{1i}]},\ (\rho_2;...;\rho_m)\ \lrcorner) & \\ = & U(\llcorner\, w_{1\mathrm{p}_1},\ (\rho_2;...;\rho_m)\ \lrcorner) & \text{where } x_0 \equiv x_{1\mathrm{p}_1} \end{array}$$

We have $x \simeq \llcorner\, w_{1\mathrm{p}_1},\ (\rho_2;...;\rho_m)\ \lrcorner$.

Then by (var-env), we have:

$$\begin{array}{lll} \mathscr{V};\mathscr{X} \vdash & \llcorner\, x_0,\ (\rho_1;\rho_2;...;\rho_m)\ \lrcorner & \\ \longrightarrow & \llcorner\, w_{1\mathrm{p}_1},\ (\rho_2;...;\rho_m)\ \lrcorner & \text{where } \rho_1(x_0) = w_{1\mathrm{p}_1} \end{array}$$

Then $\llcorner\, w_{1\mathrm{p}_1},\ (\rho_2;...;\rho_m)\ \lrcorner \prec^{\mathrm{x}} \llcorner\, x_0,\ (\rho_1;\rho_2;...;\rho_m)\ \lrcorner$.

By Lemma 357, $\mathscr{V};\mathscr{V} \vdash \llcorner\, w_{1\mathrm{p}_1},\ (\rho_2;...;\rho_m)\ \lrcorner\ wb$.

By Lemma 356, $FV(\llcorner\, w_{1\mathrm{p}_1},\ (\rho_2;...;\rho_m)\ \lrcorner) \subseteq \mathscr{V}$.

By the induction hypothesis, we have $\mathscr{V};\mathscr{X} \vdash \llcorner\, w_{1\mathrm{p}_1},\ (\rho_2;...;\rho_m)\ \lrcorner \longrightarrow v^{i+1}_{b_2}$ and $x \simeq v^{i+1}_{b_2}$.

Case 2. ($v^{i+1}_{a_1} = v^{i+1}_{a_{11}}\ v^{i+1}_{a_{12}}$). We proceed by cases on $c^{i+1}_{b_1} \in \textsc{Conf}^{i+1}_{\mathrm{env}}$.

Case i. ($c^{i+1}_{b_1} = c^{i+1}_{b_{11}}\ c^{i+1}_{b_{12}}$). Given $v^{i+1}_{a_{11}}\ v^{i+1}_{a_{12}} \simeq c^{i+1}_{b_{11}}\ c^{i+1}_{b_{12}}$, we have:

$$\begin{array}{lll} & v^{i+1}_{a_{11}}\ v^{i+1}_{a_{12}} & \\ = & U(c^{i+1}_{b_{11}}\ c^{i+1}_{b_{12}}) & \\ = & U(c^{i+1}_{b_{11}})\,U(c^{i+1}_{b_{12}}) & \end{array}$$

Then $v^{i+1}_{a_{11}} \simeq c^{i+1}_{b_{11}}$ and $v^{i+1}_{a_{12}} \simeq c^{i+1}_{b_{12}}$.

We have $\mathscr{V};\mathscr{V} \vdash c^{i}_{b_{11}}\ wb$, $\mathscr{V};\mathscr{V} \vdash c^{i}_{b_{12}}\ wb$, $FV(c^{i}_{b_{11}}) \subseteq \mathscr{V}$ and $FV(c^{i}_{b_{12}}) \subseteq \mathscr{V}$.

By the induction hypothesis, $\mathscr{V};\mathscr{X} \vdash c^{i+1}_{b_{11}} \longrightarrow^{*} v^{i+1}_{b_{21}}$, $v^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{21}}$, $\mathscr{V};\mathscr{X} \cup \textsc{Var}(v^{i+1}_{b_{21}}) \vdash c^{i+1}_{b_{12}} \longrightarrow v^{i+1}_{b_{22}}$ and $v^{i+1}_{a_{12}} \simeq v^{i+1}_{b_{22}}$.

Hence $\mathscr{V};\mathscr{X} \vdash c^{i+1}_{b_{11}}\ c^{i+1}_{b_{12}} \longrightarrow^{*} v^{i+1}_{b_{21}}\ v^{i+1}_{b_{22}}$ and $v^{i+1}_{a_{11}}\ v^{i+1}_{a_{12}} \simeq v^{i+1}_{b_{21}}\ v^{i+1}_{b_{22}}$.

Case ii. ($c^{i+1}_{b_1} = \llcorner\, t^{i+1}_{b_{11}}\ t^{i+1}_{b_{12}},\ (\rho_1;\rho_2;...;\rho_m)\ \lrcorner$). Given $v^{i+1}_{a_{11}}\ v^{i+1}_{a_{12}} \simeq \llcorner\, t^{i+1}_{b_{11}}\ t^{i+1}_{b_{12}},\ (\rho_1;\rho_2;...;\rho_m)\ \lrcorner$, we have:

$$\begin{array}{rl} & v_{a_{11}}^{i+1}\ v_{a_{12}}^{i+1} \\ = & U(\text{◖}\ t_{b_{11}}^{i+1}\ t_{b_{12}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}) \\ = & U((t_{b_{11}}^{i+1}\ t_{b_{12}}^{i+1})\overline{[w_{1i}/x_{1i}]}...\overline{[w_{mi}/x_{mi}]}) \\ = & U((t_{b_{11}}^{i+1}\overline{[w_{1i}/x_{1i}]}...\overline{[w_{mi}/x_{mi}]})\ (t_{b_{12}}^{i+1}\overline{[w_{1i}/x_{1i}]}...\overline{[w_{mi}/x_{mi}]})) \\ = & U(t_{b_{11}}^{i+1}\overline{[w_{1i}/x_{1i}]}...\overline{[w_{mi}/x_{mi}]})\ U(t_{b_{12}}^{i+1}\overline{[w_{1i}/x_{1i}]}...\overline{[w_{mi}/x_{mi}]}) \\ = & U(\text{◖}\ t_{b_{11}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗})\ U(\text{◖}\ t_{b_{12}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}) \end{array}$$

Then $v_{a_{11}}^{i+1} \simeq \text{◖}\ t_{b_{11}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}$ and $v_{a_{12}}^{i+1} \simeq \text{◖}\ t_{b_{12}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}$.
We have $\mathscr{V};\mathscr{V} \vdash \text{◖}\ t_{b_{11}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}\ wb$, $\mathscr{V};\mathscr{V} \vdash \text{◖}\ t_{b_{12}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}\ wb$, $FV(\text{◖}\ t_{b_{11}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}) \subseteq \mathscr{V}$ and $FV(\text{◖}\ t_{b_{12}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}) \subseteq \mathscr{V}$.
By the induction hypothesis, $\mathscr{V};\mathscr{X} \vdash \text{◖}\ t_{b_{11}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗} \longrightarrow^* v_{b_{21}}^{i+1}$, $v_{a_{11}}^{i+1} \simeq v_{b_{21}}^{i+1}$, $\mathscr{V};\mathscr{X} \cup \{v_{b_{21}}^{i+1}\} \vdash \text{◖}\ t_{b_{12}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗} \longrightarrow^* v_{b_{22}}^{i+1}$ and $v_{a_{12}}^{i+1} \simeq v_{b_{22}}^{i+1}$.
Hence $\mathscr{V};\mathscr{X} \vdash \text{◖}\ t_{b_{11}}^{i+1}\ t_{b_{12}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗} \longrightarrow^* v_{b_{21}}^{i+1}\ v_{b_{22}}^{i+1}$ and $v_{a_{11}}^{i+1}\ v_{a_{12}}^{i+1} \simeq v_{b_{21}}^{i+1}\ v_{b_{22}}^{i+1}$.

Case 3. $(v_{a_1}^0 = \lambda x.t_{a_{11}}^0)$. We proceed by cases on $c_{b_1}^0 \in \text{CONF}_{\text{env}}^0$.

Case i. $(c_{b_1}^0 = \text{◹}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◺})$. Then $\text{◹}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◺} \in \text{VALUE}_{\text{env}}^0$.

Case ii. $(c_{b_1}^0 = \text{◖}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗})$. Given $\lambda x.t_{a_{11}}^0 \simeq \text{◖}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}$, we have:

$$\begin{array}{rl} & \lambda x.t_{a_{11}}^0 \\ = & U(\text{◖}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}) \\ = & U(\text{◹}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◺}) \end{array}$$

Then $\lambda x.t_{a_{11}}^0 \simeq \text{◹}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◺}$.
By (lam-0-env) we have:

$$\text{◖}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗} \longrightarrow \text{◹}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◺}$$

Case iii. $(c_{b_1}^0 = \text{◖}\ \text{◹}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_l)\ \text{◺},\ (\rho_{l+1},...,\rho_m)\ \text{◗})$.
Given $\lambda x.t_{a_{11}}^0 \simeq \text{◖}\ \text{◹}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_l)\ \text{◺},\ (\rho_{l+1},...,\rho_m)\ \text{◗}$, we have:

$$\begin{array}{rl} & \lambda x.t_{a_{11}}^0 \\ = & U(\text{◖}\ \text{◹}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_l)\ \text{◺},\ (\rho_{l+1},...,\rho_m)\ \text{◗}) \\ = & U(\text{◹}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_l;\rho_{l+1},...,\rho_m)\ \text{◺}) \end{array}$$

Then $\lambda x.t_{a_{11}}^0 \simeq \text{◹}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_l;\rho_{l+1},...,\rho_m)\ \text{◺}$.
By (clov-0-env) we have:

$$\begin{array}{c} \text{◖}\ \text{◹}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_l)\ \text{◺},\ (\rho_{l+1},...,\rho_m)\ \text{◗} \longrightarrow \\ \text{◹}\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_l;\rho_{l+1},...,\rho_m)\ \text{◺} \end{array}$$

Case iv. $(c_{b_1}^0 = \text{◖}\ x_0,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗})$ This is analogous to the $(c_{b_1}^{i+1} = \text{◖}\ x_0,\ \overline{\rho_i}^{+m}\ \text{◗})$ subcase of the $(v_{a_1}^{i+1} = x)$ case.

Case 4. $(v_{a_1}^{i+1} = \lambda x.v_{a_{11}}^{i+1})$. We proceed by cases on $c_{b_1}^{i+1} \in \text{CONF}_{\text{env}}^{i+1}$.

Case i. $(c_{b_1}^{i+1} = \lambda x.v_{b_{11}}^{i+1})$. Then $\lambda x.v_{b_{11}}^{i+1} \in \text{VALUE}_{\text{env}}^{i+1}$.

Case ii. $(c_{b_1}^{i+1} = \lambda x.c_{b_{11}}^{i+1}$ where $c_{b_{11}}^{i+1} \notin \text{VALUE}_{\text{env}}^{i+1})$. Given $\lambda x.v_{a_{11}}^{i+1} \simeq \lambda x.c_{b_{11}}^{i+1}$, we have:

$$\begin{array}{rl} & \lambda x.v_{a_{11}}^{i+1} \\ = & U(\lambda x.c_{b_{11}}^{i+1}) \\ = & \lambda x.U(c_{b_{11}}^{i+1}) \end{array}$$

Then $v_{a_{11}}^{i+1} \simeq c_{b_{11}}^{i+1}$.
We have $\mathscr{V} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash c_{b_{11}}^{i+1}$ and $FV(c_b^{i+1}) \subseteq \mathscr{V} \cup \{x\}$.
By the induction hypothesis, we have $\mathscr{V} \cup \{x\}; \mathscr{X} \vdash c_{b_{11}}^{i+1} \longrightarrow^* v_{b_{12}}^{i+1}$ and $v_{a_{11}}^{i+1} \simeq v_{b_{12}}^{i+1}$.
Hence $\mathscr{V}; \mathscr{X} \vdash \lambda x.c_{b_{11}}^{i+1} \longrightarrow^* \lambda x.v_{b_{12}}^{i+1}$ and $\lambda x.v_{a_{11}}^{i+1} \simeq \lambda x.v_{b_{12}}^{i+1}$.

Case iii. $(c_{b_1}^{i+1} = \triangleleft\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_m)\ \triangleright)$. Then $\triangleleft\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_m)\ \triangleright \in \text{VALUE}_{\text{env}}^{i+1}$.

Case iv. $(c_{b_1}^{i+1} = \blacktriangleleft\ \lambda x_0.t_{b_{11}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \blacktriangleright)$.
Let $x_N \notin \mathscr{X} \cup \text{VAR}(v_{a_1}^{i+1})$. Then $\lambda x.v_{a_{11}}^{i+1} \sim_\alpha \lambda x_N.v_{a_{11}}^{i+1}[x_N/x]$.
By (lam-(i+1)-env) we have:

$$\mathscr{V}; \mathscr{X} \vdash \blacktriangleleft\ \lambda x_0.t_{b_{11}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \blacktriangleright \longrightarrow \lambda x_N.\blacktriangleleft\ t_{b_{11}}^{i+1},\ (\rho_1[x_0 \mapsto x_N];\rho_2[x_N \mapsto x_N];...;\rho_m[x_N \mapsto x_N])\ \blacktriangleright$$

By Lemma 357, $\mathscr{V}; \mathscr{V} \vdash \lambda x_N.\blacktriangleleft\ t_{b_{11}}^{i+1},\ (\rho_1[x_0 \mapsto x_N];\rho_2[x_N \mapsto x_N];...;\rho_m[x_N \mapsto x_N])\ \blacktriangleright\ wb$.
By Lemma 356, $FV(\lambda x_N.\blacktriangleleft\ t_{b_{11}}^{i+1},\ (\rho_1[x_0 \mapsto x_N];\rho_2[x_N \mapsto x_N];...;\rho_m[x_N \mapsto x_N])\ \blacktriangleright) \subseteq \mathscr{V}$.
Then $\mathscr{V} \cup \{x_N\}; \mathscr{V} \cup \{x_N\} \vdash \blacktriangleleft\ t_{b_{11}}^{i+1},\ (\rho_1[x_0 \mapsto x_N];\rho_2[x_N \mapsto x_N];...;\rho_m[x_N \mapsto x_N])\ \blacktriangleright\ wb$
and $FV(\blacktriangleleft\ t_{b_{11}}^{i+1},\ (\rho_1[x_0 \mapsto x_N];\rho_2[x_N \mapsto x_N];...;\rho_m[x_N \mapsto x_N])\ \blacktriangleright) \subseteq \mathscr{V} \cup \{x_N\}$.
We have:

$$\begin{array}{rll} & \lambda x_N.v_{a_{11}}^{i+1}[x_N/x] & \\ = & U(\mathscr{X} \mid \blacktriangleleft\ \lambda x_0.t_{b_{11}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \blacktriangleright) & \\ = & \lambda x_N.U(\mathscr{X} \cup \{x_N\} \mid \blacktriangleleft\ t_{b_{11}}^{i+1},\ (\rho_1[x_0 \mapsto x_N];\rho_2[x_N \mapsto x_N];...;\rho_m[x_N \mapsto x_N])\ \blacktriangleright) & \text{where } x_N \notin \mathscr{X} \end{array}$$

Then $v_{a_{11}}^{i+1}[x_N/x] \simeq \blacktriangleleft\ t_{b_{11}}^{i+1},\ (\rho_1[x_0 \mapsto x_N];\rho_2[x_N \mapsto x_N];...;\rho_m[x_N \mapsto x_N])\ \blacktriangleright$.
By the induction hypothesis, we have $\blacktriangleleft\ t_{b_{11}}^{i+1},\ (\rho_1[x_0 \mapsto x_N];\rho_2[x_N \mapsto x_N];...;\rho_m[x_N \mapsto x_N])\ \blacktriangleright \longrightarrow^* v_{b_{12}}^{i+1}$ and $v_{a_{11}}^{i+1} \simeq v_{b_{12}}^{i+1}$.
Hence $\mathscr{V}; \mathscr{X} \vdash \blacktriangleleft\ \lambda x_0.t_{b_{11}}^{i+1},\ (\rho_1;\rho_2;...;\rho_m)\ \blacktriangleright \longrightarrow \lambda x_N.v_{b_{12}}^{i+1}$ and $\lambda x_N.v_{a_{11}}^{i+1}[x_N/x] \simeq \lambda x_N.v_{b_{12}}^{i+1}$.

Case v. $(c_{b_1}^{i+1} = \blacktriangleleft\ \triangleleft\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_l)\ \triangleright,\ (\rho_{l+1},...,\rho_m)\ \blacktriangleright)$.
Given $\lambda x.v_{a_{11}}^{i+1} \simeq \blacktriangleleft\ \triangleleft\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_l)\ \triangleright,\ (\rho_{l+1},...,\rho_m)\ \blacktriangleright$, we have:

$$\begin{array}{rl} & \lambda x.v_{a_{11}}^{i+1} \\ = & U(\blacktriangleleft\ \triangleleft\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_l)\ \triangleright,\ (\rho_{l+1},...,\rho_m)\ \blacktriangleright) \\ = & U(\triangleleft\ \lambda x_0.t_{b_{11}}^0,\ (\rho_1;\rho_2;...;\rho_l;\rho_{l+1};...;\rho_m)\ \triangleright) \end{array}$$

Then $\lambda x.v^{i+1}_{a_{11}} \simeq \mathbb{d}\ \lambda x_0.t^0_{b_{11}},\ (\rho_1;\rho_2;...;\rho_l;\rho_{l+1};...;\rho_m)\ \mathbb{D}$.
By (clov-0-env) we have:

$$\begin{array}{c} \text{◖}\ \mathbb{d}\ \lambda x_0.t^0_{b_{11}},\ (\rho_1;\rho_2;...;\rho_l)\ \mathbb{D},\ (\rho_{l+1},...,\rho_m)\ \text{◗} \longrightarrow \\ \mathbb{d}\ \lambda x_0.t^0_{b_{11}},\ (\rho_1;\rho_2;...;\rho_l;\rho_{l+1};...;\rho_m)\ \mathbb{D} \end{array}$$

Case vi. ($c^{i+1}_{b_1} = \text{◖}\ x_0,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}$). This is analogous to the ($c^{i+1}_{b_1} = \text{◖}\ x_0,\ \overline{\rho_i}^{+m}\ \text{◗}$) subcase of the ($v^{i+1}_{a_1} = x$) case.

Case 5. ($v^i_{a_1} = \langle v^{i+1}_{a_{11}} \rangle$). We proceed by cases on $c^i_{b_1} \in \text{CONF}^i_{\text{env}}$.

Case i. ($c^i_{b_1} = \langle c^{i+1}_{b_{11}} \rangle$). Given $\langle v^{i+1}_{a_{11}} \rangle \simeq \langle c^{i+1}_{b_{11}} \rangle$, we have:

$$\begin{array}{rl} & \langle v^{i+1}_{a_{11}} \rangle \\ = & U(\langle c^{i+1}_{b_{11}} \rangle) \\ = & \langle U(c^{i+1}_{b_{11}}) \rangle \end{array}$$

Then $v^{i+1}_{a_{11}} \simeq c^{i+1}_{b_{11}}$.
We have $\mathscr{V};\mathscr{V} \vdash t^{i+1}_{b_{11}}\ wb$ and $FV(t^{i+1}_{b_{11}}) \subseteq \mathscr{V}$.
By the induction hypothesis, $\mathscr{V};\mathscr{X} \vdash c^{i+1}_{b_{11}} \longrightarrow^* v^{i+1}_{b_{21}}$ and $v^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{21}}$.
Hence $\mathscr{V};\mathscr{X} \vdash \langle c^{i+1}_{b_{11}} \rangle \longrightarrow^* \langle v^{i+1}_{b_{21}} \rangle$ and $\langle v^{i+1}_{a_{11}} \rangle \simeq \langle v^{i+1}_{b_{21}} \rangle$.

Case ii. ($c^i_{b_1} = \text{◖}\ \langle t^{i+1}_{b_{11}} \rangle,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}$). Given $\langle v^{i+1}_{a_{11}} \rangle \simeq \text{◖}\ \langle t^{i+1}_{b_{11}} \rangle,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}$, we have:

$$\begin{array}{rl} & \langle v^{i+1}_{a_{11}} \rangle \\ = & U(\text{◖}\ \langle t^{i+1}_{b_{11}} \rangle,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}) \\ = & \langle U(\text{◖}\ t^{i+1}_{b_{11}},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}) \rangle \end{array}$$

Then $v^{i+1}_{a_{11}} \simeq \text{◖}\ t^{i+1}_{b_{11}},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}$.
By (code-env), we have:

$$\begin{array}{rl} \mathscr{V};\mathscr{X} \vdash & \text{◖}\ \langle t^{i+1}_{b_{11}} \rangle,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗} \\ \longrightarrow & \langle \text{◖}\ t^{i+1}_{b_{11}},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗} \rangle \end{array}$$

We have $\mathscr{V};\mathscr{V} \vdash \langle \text{◖}\ t^{i+1}_{b_{11}},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗} \rangle\ wb$ and $FV(\langle \text{◖}\ t^{i+1}_{b_{11}},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗} \rangle) \subseteq \mathscr{V}$.
By the induction hypothesis, $\mathscr{V};\mathscr{X} \vdash \text{◖}\ t^{i+1}_{b_{11}},\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗} \longrightarrow^* v^{i+1}_{b_{21}}$ and $v^{i+1}_{a_{11}} \simeq v^{i+1}_{b_{21}}$.
Hence $\mathscr{V};\mathscr{X} \vdash \text{◖}\ \langle t^{i+1}_{b_{11}} \rangle,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗} \longrightarrow^* \langle v^{i+1}_{b_{21}} \rangle$ and $\langle v^{i+1}_{a_{11}} \rangle \simeq \langle v^{i+1}_{b_{21}} \rangle$.

Case iii. ($v^0_{a_1} = \langle v^1_{a_{11}} \rangle$ and $c^0_{b_1} = \text{◖}\ x_0,\ (\rho_1;\rho_2;...;\rho_m)\ \text{◗}$). This is analogous to the ($c^{i+1}_{b_1} = \text{◖}\ x_0,\ \overline{\rho_i}^{+m}\ \text{◗}$) subcase of the ($v^{i+1}_{a_1} = x$) case.

Case 6. ($v^{i+2}_{a_1} = \sim v^{i+1}_{a_{11}}$). This case is analogous to the ($v^i_{a_1} = \langle v^{i+1}_{a_{11}} \rangle$) case.

Case 7. ($v^{i+1}_{a_1} = !v^{i+1}_{a_{11}}$). This case is analogous to the ($v^i_{a_1} = \langle v^{i+1}_{a_{11}} \rangle$) case.

Case 8. $(v^i_{a_1} = n)$. We proceed by cases on $c^i_{b_1} \in \textsc{Conf}^i_{\text{env}}$.

Case i. $(c^i_{b_1} = n)$. Then $n \longrightarrow^* n$ and $n \in \textsc{Value}^i_{\text{env}}$.

Case ii. $(c^i_{b_1} = \blacklozenge\ n_0,\ (\rho_1;\rho_2;...;\rho_m)\ \blacklozenge)$. Given $n \simeq \blacklozenge\ n_0,\ (\rho_1;\rho_2;...;\rho_m)\ \blacklozenge$, we have:

$$\begin{array}{rcl} & & n \\ & = & U(\blacklozenge\ n_0,\ (\rho_1;\rho_2;...;\rho_m)\ \blacklozenge) \\ & = & n_0 \end{array}$$

Then $n \simeq n_0$.

By (num-env), we have:

$$\begin{array}{rcl} & & \blacklozenge\ n_0,\ (\rho_1;\rho_2;...;\rho_m)\ \blacklozenge \\ & \longrightarrow & n_0 \end{array}$$

Case iii. $(c^i_{b_1} = \blacklozenge\ x_0,\ (\rho_1;\rho_2;...;\rho_m)\ \blacklozenge)$. This is analogous to the $(c^{i+1}_{b_1} = \blacklozenge\ x_0,\ \overline{\rho_i}^{+m}\ \blacklozenge)$ subcase of the $(v^{i+1}_{a_1} = x)$ case.

Case 9. $(v^{i+1}_{a_1} = v^{i+1}_{a_{11}} + v^{i+1}_{a_{12}})$. This case is analogous to the $(v^{i+1}_{a_1} = v^{i+1}_{a_{11}}\ v^{i+1}_{a_{12}})$ case.

□

D.3.6 Bisimulation

Lemma 368 (Simulation: (Substitutional) MetaML simulates Environmental MetaML.)**.** *If* $t^i_{a_1} \simeq c^i_{b_1}$, $\mathscr{V};\mathscr{X} \vdash c^i_{b_1} \longrightarrow c^i_{b_2}$, $\mathscr{V};\mathscr{V} \vdash c^i_{b_1}$ *wb,* $FV(c^i_{b_1}) \subseteq \mathscr{V}$*;* $\textsc{Var}(c^i_{b_1}) \subseteq \mathscr{X}$ *and* $\mathscr{V} \subseteq \mathscr{X}$*, then* $t^i_{a_1} \longrightarrow^* t^i_{a_2}$ *and* $t^i_{a_2} \simeq c^i_{b_2}$.

Proof. We proceed by structural induction on $c^i_{b_1} \in \textsc{Conf}^i_{\text{env}}$.

Case 1. $(c^i_{b_1} = v^i_{b_1})$. This case is vacuous because $c^i_{b_1} \not\longrightarrow^i$.

Case 2. $(c^i_{b_1} = c^i_{b_{11}}\ c^i_{b_{12}})$. Let $t^i_{a_1} = t^i_{a_{11}}\ t^i_{a_{12}}$. We have $t^i_{a_{11}} \simeq c^i_{b_{11}}$ and $t^i_{a_{12}} \simeq c^i_{b_{12}}$.
We also have $\mathscr{V};\mathscr{V} \vdash c^i_{b_{11}}$ *wb*, $\mathscr{V};\mathscr{V} \vdash c^i_{b_{12}}$ *wb*, $FV(c^i_{b_{11}}) \subseteq \mathscr{V}$ and $FV(c^i_{b_{12}}) \subseteq \mathscr{V}$.
We proceed by cases on $\mathscr{V};\mathscr{X} \vdash c^i_{b_1} \longrightarrow c^i_{b_2}$.

Case i. (appL-i). Let $\mathscr{V};\mathscr{X} \vdash c^i_{b_{11}}\ c^i_{b_{12}} \longrightarrow c^i_{b_{21}}\ c^i_{b_{12}}$ where $\mathscr{V};\mathscr{X} \vdash c^i_{b_{11}} \longrightarrow c^i_{b_{21}}$.
By the induction hypothesis, $t^i_{a_{11}} \longrightarrow^* t^i_{a_{21}}$ and $t^i_{a_{21}} \simeq c^i_{b_{21}}$. We have $t^i_{a_{11}}\ t^i_{a_{12}} \longrightarrow^* t^i_{a_{21}}\ t^i_{a_{12}}$ and $t^i_{a_{21}}\ t^i_{a_{12}} \simeq c^i_{b_{21}}\ c^i_{b_{12}}$.

Case ii. (appR-i). Let $c^i_{b_{11}} = v^i_{b_{11}}$ and $\mathscr{V};\mathscr{X} \vdash v^i_{b_{11}}\ c^i_{b_{12}} \longrightarrow v^i_{b_{11}}\ c^i_{b_{22}}$ where $c^i_{b_{12}} \longrightarrow c^i_{b_{22}}$. By Lemma 366, $t^i_{a_{11}} = v^i_{a_{11}}$. By the induction hypothesis, $t^i_{a_{12}} \longrightarrow^* t^i_{a_{22}}$ and $t^i_{a_{22}} \simeq c^i_{b_{22}}$. We have $v^i_{a_{11}}\ t^i_{a_{12}} \longrightarrow^* v^i_{a_{11}}\ t^i_{a_{22}}$ and $v^i_{a_{11}}\ t^i_{a_{12}} \simeq v^i_{b_{11}}\ c^i_{b_{22}}$.

Case iii. (app). Let $c^0_{b_{11}} = \triangleleft\ \lambda x.t^0_{b_{11}},\ (\rho_1;\rho_2;...;\rho_m)\ \triangleright$, $c^0_{b_{12}} = v^0_{b_{12}}$ and $\mathscr{V};\mathscr{X} \vdash \triangleleft\ \lambda x.t^0_{b_{11}},\ (\rho_1;\rho_2;...;\rho_m)\ \triangleright\ v^0_{b_{12}} \longrightarrow \blacklozenge\ t^0_{b_{11}},\ (\rho_1[x \mapsto v^0_{b_{12}}];\rho_2;...;\rho_m)\ \blacklozenge$.
By Lemma 366, we know $t^0_{a_{12}} = v^0_{a_{12}}$.

Let $x_N \notin \mathscr{X}$. We have $\triangleleft\ \lambda x.t^0_{b_{11}},\ (\rho_1;\rho_2;...;\rho_m)\ \triangleright \sim_\alpha \triangleleft\ \lambda x_N.t^0_{b_{11}}[x_N/x],\ (\rho_1[x_N \mapsto x_N];\rho_2[x_N \mapsto x_N];...;\rho_m[x_N \mapsto x_N])\ \triangleright$.

We have:

$$\begin{array}{rl} & t^0_{a_{11}} \\ = & U(\triangleleft\ \lambda x_N.t^0_{b_{11}}[x_N/x],\ (\rho_1[x_N \mapsto x_N];\rho_2[x_N \mapsto x_N];...;\rho_m[x_N \mapsto x_N])\ \triangleright) \\ = & U(\blacktriangleleft\ \lambda x_N.t^0_{b_{11}}[x_N/x],\ (\rho_1[x_N \mapsto x_N];\rho_2[x_N \mapsto x_N];...;\rho_m[x_N \mapsto x_N])\ \blacktriangleright) \\ = & U((\lambda x_N.t^0_{b_{11}}[x_N/x])\overline{[w_{1i}/x_{1i}][w_{2i}/x_{2i}]...[w_{mi}/x_{mi}]}) \\ = & U(\lambda x_N.t^0_{b_{11}}[x_N/x]\overline{[w_{1i}/x_{1i}][w_{2i}/x_{2i}]...[w_{mi}/x_{mi}]}) \\ = & \lambda x_N.U(t^0_{b_{11}}[x_N/x]\overline{[w_{1i}/x_{1i}][w_{2i}/x_{2i}]...[w_{mi}/x_{mi}]}) \\ = & \lambda x_N.U(t^0_{b_{11}})[x_N/x]\overline{[U(w_{1i})/x_{1i}][U(w_{2i})/x_{2i}]...[U(w_{mi})/x_{mi}]} \end{array}$$

Observe that

$$\begin{array}{rl} & \lambda x_N.U(t^0_{b_{11}})[x_N/x]\overline{[U(w_{1i})/x_{1i}][U(w_{2i})/x_{2i}]...[U(w_{mi})/x_{mi}]}\ v^0_{a_{12}} \\ \longrightarrow & U(t^0_{b_{11}})[x_N/x]\overline{[U(w_{1i})/x_{1i}][U(w_{2i})/x_{2i}]...[U(w_{mi})/x_{mi}]}[v^0_{a_{12}}/x_N] \end{array}$$

and

$$\begin{array}{rl} \mathscr{V};\mathscr{X} \vdash & \triangleleft\ \lambda x_N.t^0_{b_{11}}[x_N/x],\ (\rho_1;\rho_2;...;\rho_m)\ \triangleright\ v^0_{b_{12}} \\ \longrightarrow & \blacktriangleleft\ t^0_{b_{11}}[x_N/x],\ (\rho_1[x_N \mapsto v^0_{b_{12}}];\rho_2;...;\rho_m)\ \blacktriangleright\ . \end{array}$$

Since $\mathscr{V};\mathscr{V} \vdash \triangleleft\ \lambda x.t^0_{b_{11}},\ (\rho_1;\rho_2;...;\rho_m)\ \triangleright\ wb$, we have
$AV(\triangleleft\ \lambda x.t^0_{b_{11}},\ (\rho_1;\rho_2;...;\rho_m)\ \triangleright) \cap \mathscr{V} = \emptyset$.
Hence $AV(\triangleleft\ \lambda x.t^0_{b_{11}},\ (\rho_1;\rho_2;...;\rho_m)\ \triangleright) \cap FV(v^0_{b_{12}}) = \emptyset$.
We have:

$$\begin{array}{rl} & U(\blacktriangleleft\ t^0_{b_{11}}[x_N/x],\ (\rho_1[x_N \mapsto v^0_{b_{12}}];\rho_2;...;\rho_m)\ \blacktriangleright) \\ = & U(t^0_{b_{11}}[x_N/x][v^0_{b_{12}}/x_N]\overline{[w_{1i}/x_{1i}][w_{2i}/x_{2i}]...[w_{mi}/x_{mi}]}) \\ = & U(t^0_{b_{111}}[x_N/x_0]\overline{[w_{1i}/x_{1i}][w_{2i}/x_{2i}]...[w_{mi}/x_{mi}]}[v^0_{b_{12}}/x_N]) \\ = & U(t^0_{b_{111}})[x_N/x_0]\overline{[U(w_{1i})/x_{1i}][U(w_{2i})/x_{2i}]...[U(w_{mi})/x_{mi}]}[U(v^0_{b_{12}})/x_N] \\ = & U(t^0_{b_{111}})[x_N/x_0]\overline{[U(w_{1i})/x_{1i}][U(w_{2i})/x_{2i}]...[U(w_{mi})/x_{mi}]}[v^0_{a_{12}}/x_N] \end{array}$$

We get:

$$\begin{array}{rl} & U(t^0_{b_{111}})[x_N/x_0]\overline{[U(w_{1i})/x_{1i}][U(w_{2i})/x_{2i}]...[U(w_{mi})/x_{mi}]}[v^0_{a_{12}}/x_N] \\ \simeq & \blacktriangleleft\ t^0_{b_{11}}[x_N/x],\ (\rho_1[x_N \mapsto v^0_{b_{12}}];\rho_2;...;\rho_m)\ \blacktriangleright \end{array}$$

Case 3. $(c^{i+1}_{b_1} = \lambda x.c^{i+1}_{b_{11}})$. Let $t^{i+1}_{a_1} = \lambda x.t^{i+1}_{a_{11}}$. We have $t^{i+1}_{a_{11}} \simeq c^{i+1}_{b_{11}}$.
We also have $\mathscr{V} \cup \{x\};\mathscr{V} \cup \{x\} \vdash c^{i+1}_{b_{11}}\ wb$, $x \notin \mathscr{V}$ and $FV(c^{i+1}_{b_{11}}) \subseteq \mathscr{V} \cup \{x\}$.
We proceed by cases on $c^{i+1}_{b_1} \longrightarrow c^{i+1}_{b_2}$. The only case is (lambda-(i+1)).
Let $\mathscr{V};\mathscr{X} \vdash \lambda x.c^{i+1}_{b_{11}} \longrightarrow \lambda x.c^{i+1}_{b_{12}}$ where $\mathscr{V} \cup \{x\};\mathscr{X} \vdash c^{i+1}_{b_{11}} \longrightarrow c^{i+1}_{b_{12}}$. By the induction hypothesis, $t^{i+1}_{a_{11}} \longrightarrow^* t^{i+1}_{a_{12}}$ and $t^{i+1}_{a_{12}} \simeq c^{i+1}_{b_{12}}$. We have $\lambda x.t^{i+1}_{a_{11}} \longrightarrow^* \lambda x.t^{i+1}_{a_{12}}$ and $\lambda x.t^{i+1}_{a_{12}} \simeq \lambda x.c^{i+1}_{b_{12}}$.

Case 4. $(c^i_{b_1} = c^i_{b_{11}} + c^i_{b_{12}})$. This case is analogous to the $(c^i_{b_1} = c^i_{b_{11}}\ c^i_{b_{12}})$ case.

Case 5. $(c^i_{b_1} = \langle c^{i+1}_{b_{11}} \rangle)$. Let $t^i_{a_1} = \langle t^{i+1}_{a_{11}} \rangle$. We have $t^{i+1}_{a_{11}} \simeq c^{i+1}_{b_{11}}$.
We also have $\mathscr{V}; \mathscr{V} \vdash c^{i+1}_{b_{11}}$ *wb* and $FV(c^{i+1}_{b_{11}}) \subseteq \mathscr{V}$.
We proceed by cases on $\mathscr{V}; \mathscr{X} \vdash c^i_{b_1} \longrightarrow c^i_{b_2}$. The only case is (code-i).
Let $\mathscr{V}; \mathscr{X} \vdash \langle c^{i+1}_{b_{11}} \rangle \longrightarrow \langle c^{i+1}_{b_{12}} \rangle$ where $\mathscr{V}; \mathscr{X} \vdash c^{i+1}_{b_{11}} \longrightarrow c^{i+1}_{b_{12}}$.
By the induction hypothesis, $t^{i+1}_{a_{11}} \longrightarrow t^{i+1}_{a_{12}}$ and $t^{i+1}_{a_{12}} \simeq c^{i+1}_{b_{12}}$.
We have $\langle t^{i+1}_{a_{11}} \rangle \longrightarrow \langle t^{i+1}_{a_{12}} \rangle$ and $\langle t^{i+1}_{a_{12}} \rangle \simeq \langle c^{i+1}_{b_{12}} \rangle$.

Case 6. $(c^{i+1}_{b_1} = {\sim} c^i_{b_{11}})$. This case is analogous to the $(c^i_{b_1} = \langle c^{i+1}_{b_{11}} \rangle)$ case.

Case 7. $(c^i_{b_1} = !c^i_{b_{11}})$. This case is analogous to the $(c^i_{b_1} = \langle c^{i+1}_{b_{11}} \rangle)$ case.

Case 8. $(c^i_{b_1} = \blacklozenge\, t^i_{b_1},\ (\rho_1; \rho_2; ...; \rho_m)\, \blacklozenge)$. We proceed by cases on $\mathscr{V}; \mathscr{X} \vdash c^i_{b_1} \longrightarrow c^i_{b_2}$. The cases are (lam-0-env), (lam-(i+1)-env), (clov-env), (den-env), (var-env), (num-env), (app-env), (code-env), (run-env), (splice-env). It is provable that $t^i_{a_1} = t^i_{a_2} = U(c^i_{b_1}) = U(c^i_{b_2})$ holds for each of the above cases. We have $t^i_{a_1} \longrightarrow^* t^i_{a_2}$ and $t^i_{a_2} \simeq c^i_{b_2}$.

□

Lemma 369 (Simulation: Environmental MetaML simulates (Substitutional) MetaML.)**.** *If* $t^i_{a_1} \simeq c^i_{b_1}$, $t^i_{a_1} \longrightarrow t^i_{a_2}$, $\mathscr{V}; \mathscr{V} \vdash c^i_{b_1}$ *wb,* $FV(c^i_{b_1}) \subseteq \mathscr{V}$, $\textsc{Var}(c^i_{b_1}) \subseteq \mathscr{X}$ *and* $\mathscr{V} \subseteq \mathscr{X}$*, then* $\mathscr{V}; \mathscr{X} \vdash c^i_{b_1} \longrightarrow^* c^i_{b_2}$ *and* $t^i_{a_2} \simeq c^i_{b_2}$.

Proof. We proceed by simultaneous induction on the structure of $t^i_{a_1} \in \textsc{Term}^i_{\text{sub}}$ and on the closure descendant relation $\prec^{\text{x}} c^i_{b_1}$.

Case 1. $(t^i_{a_1} = x)$. This case is vacuous because $x \not\longrightarrow$.

Case 2. $(t^i_{a_1} = t^i_{a_{11}}\ t^i_{a_{12}})$. We proceed by cases on $c^i_{b_1} \in \textsc{Conf}^i_{\text{env}}$.

Case i. Let $c^i_{b_1} = c^i_{b_{11}}\ c^i_{b_{12}}$, $t^i_{a_{11}} = U(c^i_{b_{11}})$ and $t^i_{a_{12}} = U(c^i_{b_{12}})$.
We have $t^i_{a_{11}} \simeq c^i_{b_{11}}$, $t^i_{a_{12}} \simeq c^i_{b_{12}}$, $\mathscr{V}; \mathscr{V} \vdash c^i_{b_{11}}$ *wb*, $\mathscr{V}; \mathscr{V} \vdash c^i_{b_{12}}$ *wb*, $\textsc{Var}(c^i_{b_{11}}) \subseteq \mathscr{X}$, $\textsc{Var}(c^i_{b_{12}}) \subseteq \mathscr{X}$, $FV(c^i_{b_{11}}) \subseteq \mathscr{V}$ and $FV(c^i_{b_{12}}) \subseteq \mathscr{V}$.
We proceed by cases on $t^i_{a_1} \longrightarrow t^i_{a_2}$.

Case a. (appL-i). Let $t^i_{a_{11}}\ t^i_{a_{12}} \longrightarrow t^i_{a_{21}}\ t^i_{a_{12}}$ where $t^i_{a_{11}} \longrightarrow t^i_{a_{21}}$.
By the induction hypothesis, $\mathscr{V}; \mathscr{X} \vdash c^i_{b_{11}} \longrightarrow^* c^i_{b_{21}}$ and $t^i_{a_{21}} \simeq c^i_{b_{21}}$.
Hence $\mathscr{V}; \mathscr{X} \vdash c^i_{b_{11}}\ c^i_{b_{12}} \longrightarrow^* c^i_{b_{21}}\ c^i_{b_{12}}$ and $t^i_{a_{21}}\ t^i_{a_{12}} \simeq c^i_{b_{21}}\ c^i_{b_{12}}$.

Case b. (appR-i). Let $t^i_{a_{11}} = v^i_{a_{11}}$ and $v^i_{a_{11}}\ t^i_{a_{12}} \longrightarrow v^i_{a_{11}}\ t^i_{a_{22}}$ where $t^i_{a_{12}} \longrightarrow t^i_{a_{22}}$.
By Lemma 367, $\mathscr{V}; \mathscr{X} \vdash c^i_{b_{11}} \longrightarrow^* v^i_{b_{11}}$.
Then $\mathscr{V}; \mathscr{X} \vdash c^i_{b_{11}}\ c^i_{b_{12}} \longrightarrow^* v^i_{b_{11}}\ c^i_{b_{12}}$.
By the induction hypothesis, $\mathscr{V}; \mathscr{X} \cup \textsc{Var}(v^i_{b_{11}}) \vdash c^i_{b_{12}} \longrightarrow^* c^i_{b_{22}}$ and $t^i_{a_{22}} \simeq c^i_{b_{22}}$.
Then $\mathscr{V}; \mathscr{X} \cup \textsc{Var}(v^i_{b_{11}}) \vdash v^i_{b_{11}}\ c^i_{b_{12}} \longrightarrow^* v^i_{b_{11}}\ c^i_{b_{22}}$
Hence $\mathscr{V}; \mathscr{X} \vdash c^i_{b_{11}}\ c^i_{b_{12}} \longrightarrow^* v^i_{b_{11}}\ c^i_{b_{22}}$ and $t^i_{a_{11}}\ t^i_{a_{22}} \simeq v^i_{b_{11}}\ c^i_{b_{22}}$.

Case c. (app-0). Let $t^0_{a_{11}} = \lambda x.t^0_{a_{111}}$, $t^0_{a_{12}} = v^0_{a_{12}}$ and $\lambda x.t^0_{a_{111}}\ v^0_{a_{12}} \longrightarrow t^0_{a_{111}}[v^0_{a_{12}}/x]$.
Given $\lambda x.t^0_{a_{111}} \simeq c^0_{b_{11}}$, by Lemma 367, $\mathscr{V};\mathscr{X} \vdash c^0_{b_{11}} \longrightarrow^* v^0_{b_{11}}$ and $\lambda x.t^0_{a_{111}} \simeq v^0_{b_{11}}$. Then $\mathscr{V};\mathscr{X} \vdash c^0_{b_{11}}\ c^0_{b_{12}} \longrightarrow^* v^0_{b_{11}}\ c^0_{b_{12}}$.
Given $v^0_{a_{12}} \simeq c^0_{b_{12}}$, by Lemma 367, $\mathscr{V};\mathscr{X} \vdash c^0_{b_{12}} \longrightarrow^* v^0_{b_{12}}$ and $v^0_{a_{12}} \simeq v^0_{b_{12}}$. Then $\mathscr{V};\mathscr{X} \cup \{v^0_{b_{11}}\} \vdash c^0_{b_{12}} \longrightarrow^* v^0_{b_{12}}$.
We have $\mathscr{V};\mathscr{X} \vdash c^0_{b_{11}}\ c^0_{b_{12}} \longrightarrow^* v^0_{b_{11}}\ v^0_{b_{12}}$.
By Lemma 357, $\mathscr{V};\mathscr{V} \vdash v^0_{b_{11}}\ wb$, $\mathscr{V};\mathscr{V} \vdash v^0_{b_{12}}\ wb$. By Lemma 356, $FV(v^0_{b_{11}}) \subseteq \mathscr{V}$ and $FV(v^0_{b_{12}}) \subseteq \mathscr{V}$.
We proceed by cases on $v^0_{b_{11}}$ in $\lambda x.t^0_{a_{111}} \simeq v^0_{b_{11}}$. The only possible case is $v^0_{b_{11}} = \mathbb{d}\ \lambda x_0.t^0_{b_{111}},\ (\rho_1,\rho_2,...,\rho_m)\ \mathbb{D}$.
Let $x_N \notin \mathscr{X} \cup \textsc{Var}(t^0_{a_{11}}\ t^0_{a_{12}}) \cup \textsc{Var}(v^0_{b_{11}}\ v^0_{b_{12}})$.
We have $\lambda x.t^0_{a_{111}} \sim_\alpha \lambda x_N.t^0_{a_{111}}[x_N/x]$ and $\mathbb{d}\ \lambda x_0.t^0_{b_{111}},\ (\rho_1,\rho_2,...,\rho_m)\ \mathbb{D} \sim_\alpha \mathbb{d}\ \lambda x_N.t^0_{b_{111}}[x_N/x_0],\ (\rho_1[x_N \mapsto x_N],\rho_2[x_N \mapsto x_N],...,\rho_m[x_N \mapsto x_N])\ \mathbb{D}$.
We have:

$$
\begin{array}{ll}
 & \lambda x_N.t^0_{a_{111}}[x_N/x] \\
= & U(\mathbb{d}\ \lambda x_N.t^0_{b_{111}}[x_N/x_0],\ (\rho_1[x_N \mapsto x_N],\rho_2[x_N \mapsto x_N],...,\rho_m[x_N \mapsto x_N])\ \mathbb{D}) \\
= & U(\blacklozenge\ \lambda x_N.t^0_{b_{111}}[x_N/x_0],\ (\rho_1[x_N \mapsto x_N],\rho_2[x_N \mapsto x_N],...,\rho_m[x_N \mapsto x_N])\ \blacklozenge) \\
= & U((\lambda x_N.t^0_{b_{111}}[x_N/x_0])\overline{[w_{1i}/x_{1i}][w_{2i}/x_{2i}]}...\overline{[w_{mi}/x_{mi}]}) \\
= & U(\lambda x_N.t^0_{b_{111}}[x_N/x_0]\overline{[w_{1i}/x_{1i}][w_{2i}/x_{2i}]}...\overline{[w_{mi}/x_{mi}]}) \\
= & \lambda x_N.U(t^0_{b_{111}}[x_N/x_0]\overline{[w_{1i}/x_{1i}][w_{2i}/x_{2i}]}...\overline{[w_{mi}/x_{mi}]})
\end{array}
$$

Then $t^0_{a_{111}}[x_N/x] \simeq$
$U(t^0_{b_{111}}[x_N/x_0]\overline{[w_{1i}/x_{1i}]}[x_N/x_N]\overline{[w_{2i}/x_{2i}]}[x_N/x_N]...\overline{[w_{mi}/x_{mi}]}[x_N/x_N])$.
Observe that

$$\lambda x_N.t^0_{a_{111}}[x_N/x]\ v^0_{a_{12}} \longrightarrow t^0_{a_{111}}[x_N/x][v^0_{a_{12}}/x_N]$$

and

$$
\begin{array}{rl}
\mathscr{V};\mathscr{X} \cup \textsc{Var}(v^0_{b_{11}}\ v^0_{b_{12}}) \vdash & \mathbb{d}\ \lambda x_N.t^0_{b_{111}}[x_N/x_0],\ (\rho_1,\rho_2,...,\rho_m)\ \mathbb{D}\ v^0_{b_{12}} \\
\longrightarrow & \blacklozenge\ t^0_{b_{111}}[x_N/x_0],\ (\rho_1[x_N \mapsto v^0_{b_{12}}],\rho_2,...,\rho_m)\ \blacklozenge\ .
\end{array}
$$

Since $FV(\mathbb{d}\ \lambda x_N.t^0_{b_{111}}[x_N/x_0],\ (\rho_1,\rho_2,...,\rho_m)\ \mathbb{D}\ v^0_{b_{12}}) \subseteq \mathscr{V}$, we have $FV(v^0_{b_{12}}) \subseteq \mathscr{V}$.
Since $\mathscr{V};\mathscr{V} \vdash \mathbb{d}\ \lambda x_N.t^0_{b_{111}}[x_N/x_0],\ (\rho_1,\rho_2,...,\rho_m)\ \mathbb{D}\ wb$, we have $AV(\mathbb{d}\ \lambda x_N.t^0_{b_{111}}[x_N/x_0],\ (\rho_1,\rho_2,...,\rho_m)\ \mathbb{D}) \cap \mathscr{V} = \emptyset$.
We have:

$$
\begin{array}{ll}
 & U(\blacklozenge\ t^0_{b_{111}}[x_N/x_0],\ (\rho_1[x_N \mapsto v^0_{b_{12}}],\rho_2,...,\rho_m)\ \blacklozenge) \\
= & U(t^0_{b_{111}}[x_N/x_0][v^0_{b_{12}}/x_N]\overline{[w_{1i}/x_{1i}][w_{2i}/x_{2i}]}...\overline{[w_{mi}/x_{mi}]}) \\
= & U(t^0_{b_{111}}[x_N/x_0]\overline{[w_{1i}/x_{1i}][w_{2i}/x_{2i}]}...\overline{[w_{mi}/x_{mi}]}[v^0_{b_{12}}/x_N]) \\
= & U(t^0_{b_{111}})[x_N/x_0]\overline{[U(w_{1i})/x_{1i}][U(w_{2i})/x_{2i}]}...\overline{[U(w_{mi})/x_{mi}]}[U(v^0_{b_{12}})/x_N] \\
= & U(t^0_{b_{111}}[x_N/x_0]\overline{[w_{1i}/x_{1i}][w_{2i}/x_{2i}]}...\overline{[w_{mi}/x_{mi}]})[U(v^0_{b_{12}})/x_N] \\
= & t^0_{a_{111}}[x_N/x][v^0_{a_{12}}/x_N]
\end{array}
$$

We get:

$$\begin{array}{rl} & t^0_{a_{111}}[x_N/x][v^0_{a_{12}}/x_N] \\ \simeq & \text{◖}\ t^0_{b_{111}}[x_N/x_0],\ (\rho_1[x_N \mapsto v^0_{b_{12}}], \rho_2, ..., \rho_m)\ \text{◗} \end{array}$$

Case ii. Let $c^i_{b_1} = \text{◖}\ (t^i_{b_{11}}\ t^i_{b_{12}}),\ \rho^*\ \text{◗}$.
We have $\mathscr{V}; \mathscr{V} \vdash \text{◖}\ (t^i_{b_{11}}\ t^i_{b_{12}}),\ \rho^*\ \text{◗}\ wb$, $\textsc{Var}(\text{◖}\ (t^i_{b_{11}}\ t^i_{b_{12}}),\ \rho^*\ \text{◗}) \subseteq \mathscr{X}$ and $FV(\text{◖}\ (t^i_{b_{11}}\ t^i_{b_{12}}),\ \rho^*\ \text{◗}) \subseteq \mathscr{V}$.
By (app-env), we have $\text{◖}\ (t^i_{b_{11}}\ t^i_{b_{12}}),\ \rho^*\ \text{◗} \longrightarrow \text{◖}\ t^i_{b_{11}},\ \rho^*\ \text{◗◖}\ t^i_{b_{12}},\ \rho^*\ \text{◗}$.
Then $\text{◖}\ t^i_{b_{11}},\ \rho^*\ \text{◗◖}\ t^i_{b_{12}},\ \rho^*\ \text{◗} \prec^{\mathrm{x}} \text{◖}\ (t^i_{b_{11}}\ t^i_{b_{12}}),\ \rho^*\ \text{◗}$.
Given $t^i_{a_{11}}\ t^i_{a_{12}} \simeq \text{◖}\ (t^i_{b_{11}}\ t^i_{b_{12}}),\ \rho^*\ \text{◗}$, we have $t^i_{a_{11}} \simeq \text{◖}\ t^i_{b_{11}},\ \rho^*\ \text{◗}$ and $t^i_{a_{12}} \simeq \text{◖}\ t^i_{b_{12}},\ \rho^*\ \text{◗}$.
Suppose $t^i_{a_{11}}\ t^i_{a_{12}} \longrightarrow t^i_{a_2}$, by the induction hypothesis, $\mathscr{V}; \mathscr{X} \vdash \text{◖}\ t^i_{b_{11}},\ \rho^*\ \text{◗◖}\ t^i_{b_{12}},\ \rho^*\ \text{◗} \longrightarrow^* c^i_{b_2}$ and $t^i_{a_2} \simeq c^i_{b_2}$.

Case 3. $(t^{i+1}_{a_1} = \lambda x.t^{i+1}_{a_{11}})$. We proceed by cases on $t^{i+1}_{a_1} \longrightarrow t^{i+1}_{a_2}$. The only case is (lambda-(i+1)).
Let $\lambda x.t^{i+1}_{a_{11}} \longrightarrow \lambda x.t^{i+1}_{a_{21}}$ where $t^{i+1}_{a_{11}} \longrightarrow t^{i+1}_{a_{21}}$.
Let $\mathscr{V}; \mathscr{V} \vdash c^{i+1}_{b_1}\ wb$, $FV(c^{i+1}_{b_1}) \subseteq \mathscr{V}$, $\textsc{Var}(c^{i+1}_{b_1}) \subseteq \mathscr{X}$ and $\mathscr{V} \subseteq \mathscr{X}$.
We proceed by cases on $c^{i+1}_{b_1} \in \textsc{Conf}^{i+1}_{\mathrm{env}}$.

Case i. Let $c^{i+1}_{b_1} = \lambda x.c^{i+1}_{b_{11}}$.
We have $\lambda x.t^{i+1}_{a_{11}} = U(\lambda x.c^{i+1}_{b_{11}}) = \lambda x.U(c^{i+1}_{b_{11}})$. Then $t^{i+1}_{a_{11}} \simeq c^{i+1}_{b_{11}}$.
Suppose $c^{i+1}_{b_{11}} \notin \textsc{Value}^{i+1}_{\mathrm{env}}$. Otherwise by Lemma 366, $t^{i+1}_{a_1} \in \textsc{Value}^{i+1}_{\mathrm{sub}}$ which means $t^{i+1}_{a_1} \not\longrightarrow^{i+1}$.
We have $\mathscr{V} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash c^{i+1}_{b_{11}}\ wb$, $x \notin \mathscr{V}$ and $FV(c^{i+1}_{b_{11}}) \subseteq \mathscr{V} \cup \{x\}$.
By the induction hypothesis, $\mathscr{V} \cup \{x\}; \mathscr{V} \cup \{x\} \vdash c^{i+1}_{b_{11}} \longrightarrow^* c^{i+1}_{b_{21}}$ and $t^{i+1}_{a_{21}} \simeq c^{i+1}_{b_{21}}$.
Hence $\mathscr{V}; \mathscr{V} \vdash \lambda x_0.c^{i+1}_{b_{11}} \longrightarrow^* \lambda x_0.c^{i+1}_{b_{11}}$ and $\lambda x.t^{i+1}_{a_{21}} \simeq \lambda x_0.c^{i+1}_{b_{21}}$.

Case ii. Let $c^{i+1}_{b_1} = \text{◖}\ \lambda x_0.t^{i+1}_{b_{11}},\ (\rho_1; \rho_2; ..., \rho_m)\ \text{◗}$.
Let $x_N \notin \mathscr{X}$.
We have $\text{◖}\ \lambda x_0.t^{i+1}_{b_{11}},\ (\rho_1; \rho_2; ..., \rho_m)\ \text{◗} \longrightarrow \lambda x_N.\text{◖}\ t^{i+1}_{b_{11}},\ (\rho_1[x_0 \mapsto x_N]; \rho_2[x_N \mapsto x_N]; ..., \rho_m[x_N \mapsto x_N])\ \text{◗}$.
Then $\lambda x_N.\text{◖}\ t^{i+1}_{b_{11}},\ (\rho_1[x_0 \mapsto x_N]; \rho_2[x_N \mapsto x_N]; ..., \rho_m[x_N \mapsto x_N])\ \text{◗} \prec^{\mathrm{x}} \text{◖}\ \lambda x_0.t^{i+1}_{b_{11}},\ (\rho_1; \rho_2; ..., \rho_m)\ \text{◗}$.
By Lemma 357, $\mathscr{V}; \mathscr{V} \vdash \lambda x_N.\text{◖}\ t^{i+1}_{b_{11}},\ (\rho_1[x_0 \mapsto x_N]; \rho_2[x_N \mapsto x_N]; ..., \rho_m[x_N \mapsto x_N])\ \text{◗}\ wb$.
By Lemma 356, $FV(\lambda x_N.\text{◖}\ t^{i+1}_{b_{11}},\ (\rho_1[x_0 \mapsto x_N]; \rho_2[x_N \mapsto x_N]; ..., \rho_m[x_N \mapsto x_N])\ \text{◗}) \subseteq \mathscr{V}$.
Provably we have $\lambda x.t^{i+1}_{a_{11}} = U(\text{◖}\ \lambda x_0.t^{i+1}_{b_{11}},\ (\rho_1; \rho_2; ..., \rho_m)\ \text{◗}) = U(\lambda x_N.\text{◖}\ t^{i+1}_{b_{11}},\ (\rho_1[x_0 \mapsto x_N]; \rho_2[x_N \mapsto x_N]; ..., \rho_m[x_N \mapsto x_N]))\ \text{◗})$. Given $\lambda x.t^{i+1}_{a_{11}} \longrightarrow \lambda x.t^{i+1}_{a_{21}}$, by the induction hypothesis, $\mathscr{V}; \mathscr{X} \cup \{x_N\} \vdash \lambda x_N.\text{◖}\ t^{i+1}_{b_{11}},\ (\rho_1[x_0 \mapsto x_N]; \rho_2[x_N \mapsto x_N]; ..., \rho_m[x_N \mapsto x_N])\ \text{◗} \longrightarrow^* c^{i+1}_{b_2}$ and $\lambda x.t^{i+1}_{a_{21}} \simeq c^{i+1}_{b_2}$.

Case 4. $(t^i_{a_1} = n)$. This case is vacuous because $n \not\longrightarrow^i$.

Case 5. $(t^i_{a_1} = t^i_{a_{11}} + t^i_{a_{12}})$. This case is analogous to the $(t^i_{a_1} = t^i_{a_{11}}\ t^i_{a_{12}})$ case.

Case 6. $(t^i_{a_1} = \langle t^{i+1}_{a_{11}} \rangle)$. We proceed by cases on $t^i_{a_1} \longrightarrow t^i_{a_2}$. The only case is (code-i).
Then $\langle t^{i+1}_{a_{11}} \rangle \longrightarrow \langle t^{i+1}_{a_{21}} \rangle$ where $t^{i+1}_{a_{11}} \longrightarrow t^{i+1}_{a_{21}}$.
Let $\mathscr{V}; \mathscr{V} \vdash c^i_{b_1}$ *wb*, $FV(c^i_{b_1}) \subseteq \mathscr{V}$, $\textsc{Var}(c^i_{b_1}) \subseteq \mathscr{X}$ and $\mathscr{V} \subseteq \mathscr{X}$.
We proceed by cases on $c^i_{b_1} \in \textsc{Conf}^i_{\text{env}}$.

Case i. Let $c^i_{b_1} = \langle c^{i+1}_{b_{11}} \rangle$. Then $t^{i+1}_{a_{11}} \simeq c^{i+1}_{b_{11}}$.
We have $\mathscr{V}; \mathscr{V} \vdash c^{i+1}_{b_{11}}$ *wb* and $FV(c^{i+1}_{b_{11}}) \subseteq \mathscr{V}$.
By the induction hypothesis, $\mathscr{V}; \mathscr{X} \vdash c^{i+1}_{b_{11}} \longrightarrow^* c^{i+1}_{b_{21}}$ and $t^{i+1}_{a_{21}} \simeq c^{i+1}_{b_{21}}$.
Hence $\mathscr{V}; \mathscr{X} \vdash \langle c^{i+1}_{b_{11}} \rangle \longrightarrow^* \langle c^{i+1}_{b_{11}} \rangle$ and $\langle t^{i+1}_{a_{21}} \rangle \simeq \langle c^{i+1}_{b_{21}} \rangle$.

Case ii. Let $c^{i+1}_{b_1} = \blacklozenge\ \langle t^{i+1}_{b_{11}} \rangle, (\rho_1; \rho_2; ...; \rho_m)\ \blacklozenge$.
We have $\mathscr{V}; \mathscr{V} \vdash \blacklozenge\ t^{i+1}_{b_{11}}, (\rho_1; \rho_2; ...; \rho_m)\ \blacklozenge$ *wb* and $FV(\blacklozenge\ t^{i+1}_{b_{11}}, (\rho_1; \rho_2; ...; \rho_m)\ \blacklozenge) \subseteq \mathscr{V}$.
By (code-env), we have $\mathscr{V}; \mathscr{X} \vdash \blacklozenge\ \langle t^{i+1}_{b_{11}} \rangle, (\rho_1; \rho_2; ...; \rho_m)\ \blacklozenge \longrightarrow \langle \blacklozenge\ t^{i+1}_{b_{11}}, (\rho_1; \rho_2; ...; \rho_m)\ \blacklozenge \rangle$.
Then $\langle \blacklozenge\ t^{i+1}_{b_{11}}, (\rho_1; \rho_2; ...; \rho_m)\ \blacklozenge \rangle \prec^{\mathrm{x}} \blacklozenge\ \langle t^{i+1}_{b_{11}} \rangle, (\rho_1; \rho_2; ...; \rho_m)\ \blacklozenge$.
Provably we have $\langle t^{i+1}_{a_{11}} \rangle = U(\blacklozenge\ \langle t^{i+1}_{b_{11}} \rangle, (\rho_1; \rho_2; ...; \rho_m)\ \blacklozenge) = U(\langle \blacklozenge\ t^{i+1}_{b_{11}}, (\rho_1; \rho_2; ...; \rho_m)\ \blacklozenge \rangle)$.
Given $\langle t^{i+1}_{a_{11}} \rangle \longrightarrow \langle t^{i+1}_{a_{21}} \rangle$, by the induction hypothesis, $\mathscr{V}; \mathscr{X} \vdash \langle \blacklozenge\ t^{i+1}_{b_{11}}, (\rho_1; \rho_2; ...; \rho_m)\ \blacklozenge \rangle \longrightarrow^* c^i_{b_2}$ and $\langle t^{i+1}_{a_{21}} \rangle \simeq c^i_{b_2}$.

Case 7. $(t^{i+1}_{a_1} = {\sim} t^i_{a_{11}})$. This case is analogous to the $(t^i_{a_1} = \langle t^{i+1}_{a_{11}} \rangle)$ case.

Case 8. $(t^i_{a_1} = !t^i_{a_{11}})$. This case is analogous to the $(t^i_{a_1} = \langle t^{i+1}_{a_{11}} \rangle)$ case.

□

D.3.7 Soundness and Completeness

Theorem 370 (Soundness of Environmental MetaML w.r.t. (Substitutional) MetaML)**.** *If* $t^0_{a_1} \simeq c^0_{b_1}$, $\emptyset; \emptyset \vdash c^0_{b_1}$ *wb,* $FV(c^0_{b_1}) = \emptyset$ *and* $t^0_{a_1} \longrightarrow^{0*} v^0_{a_2}$ *in Substitutional MetaML, then* $\triangleright\, c^0_{b_1} \longrightarrow^* v^0_{b_2}$ *in Environmental MetaML and* $v^0_{a_2} \simeq v^0_{b_2}$.

Proof. We proceed by induction on the length of $t^0_{a_1} \longrightarrow^{0*} v^0_{a_2}$.

Case 1. (0). Let $t^0_{a_1} = v^0_{a_2}$. By Lemma 367, $\emptyset; \textsc{Var}(c^0_{b_1}) \vdash c^0_{b_1} \longrightarrow^{0*} v^0_{b_2}$ and $v^0_{a_2} \simeq v^0_{b_2}$. We have $\triangleright\, c^0_{b_1} \longrightarrow^* v^0_{b_2}$.

Case 2. $(n+1)$. Let $t^0_{a_1} \longrightarrow^0 t^0_{a_2} \longrightarrow^{0(n)} v^0_{a_2}$.
We have $\emptyset; \emptyset \vdash c^0_{b_1}$ *wb* and $FV(c^0_{b_1}) = \emptyset$.
By Lemma 369, $\emptyset; \textsc{Var}(c^0_{b_1}) \vdash c^0_{b_1} \longrightarrow^{0*} c^0_{b_2}$ and $t^0_{a_2} \simeq c^0_{b_2}$. Then $\triangleright\, c^0_{b_1} \longrightarrow^* c^0_{b_2}$.
By Lemma 357, $\emptyset; \emptyset \vdash c^0_{b_2}$ *wb*. By Lemma 356, $FV(c^0_{b_2}) = \emptyset$.
By the induction hypothesis, $\triangleright\, c^0_{b_2} \longrightarrow^* v^0_{b_2}$ and $v^0_{a_2} \simeq v^0_{b_2}$.
We have $\triangleright\, c^0_{b_1} \longrightarrow^* v^0_{b_2}$.

□

Theorem 371 (Completeness of Environmental MetaML w.r.t. (Substitutional) MetaML)**.** *If* $t^0_{a_1} \simeq c^0_{b_1}$, $\emptyset;\emptyset \vdash c^0_{b_1}$ *wb,* $FV(c^0_{b_1}) = \emptyset$ *and* $\triangleright\, c^0_{b_1} \longrightarrow^* v^0_{b_2}$ *in Environmental MetaML, then* $t^0_{a_1} \longrightarrow^{0*} v^0_{a_2}$ *in Substitutional MetaML and* $v^0_{a_2} \simeq v^0_{b_2}$.

Proof. We proceed by induction on the length of $\triangleright\, c^0_{b_1} \longrightarrow^* v^0_{b_2}$.

Case 1. (0). Let $c^0_{b_1} = v^0_{b_2}$. By Lemma 366, $t^0_{a_1} \in \text{VALUE}^0_{\text{sub}}$. Let $v^0_{a_2} = t^0_{a_1}$. We have $t^0_{a_1} \longrightarrow^{0*} v^0_{a_2}$ and $v^0_{a_2} \simeq v^0_{b_2}$.

Case 2. $(n+1)$. Let $\triangleright\, c^0_{b_1} \longrightarrow c^0_{b_2} \longrightarrow^{(n)} v^0_{b_2}$.
We have $\emptyset; \text{VAR}(c^0_{b_1}) \vdash c^0_{b_1} \longrightarrow^0 c^0_{b_2}$, $\emptyset;\emptyset \vdash c^0_{b_1}$ *wb* and $FV(c^0_{b_1}) = \emptyset$.
By Lemma 368, $t^0_{a_1} \longrightarrow^{0*} t^0_{a_2}$ and $t^0_{a_2} \simeq c^0_{b_2}$.
By Lemma 357, $\emptyset;\emptyset \vdash c^0_{b_2}$ *wb*. By Lemma 356, $FV(c^0_{b_2}) = \emptyset$.
By the induction hypothesis, $t^0_{a_2} \longrightarrow^{0*} v^0_{a_2}$ and $v^0_{a_2} \simeq v^0_{b_2}$.
We have $t^0_{a_1} \longrightarrow^{0*} v^0_{a_2}$.

□

Theorem 372 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{MetaML}}$, $eval_{\text{MetaML:SubSOS}}(t)$ *is Kleene equal to* $eval_{\text{MetaML:EnvSOS}}(t)$.

Proof. We first show that if $eval_{\text{MetaML:SubSOS}}(t) = a$ where $a \in \text{ANS}_{\text{MetaML}}$, then $eval_{\text{MetaML:EnvSOS}}(t) = a$.

Case 1. If $eval_{\text{MetaML:SubSOS}}(t) = \texttt{function}$, then $t \longrightarrow^{0*} \lambda x.t'^0$ in Substitutional MetaML. Observe that $t \simeq t$. By Theorem 370, $\triangleright \langle\!\langle\, t,\ \rho_{\text{init}}^{\text{VAR}(t)} \,\rangle\!\rangle \longrightarrow^* v$ in Environmental MetaML and $\lambda x.t'^0 \simeq v$. Then $v = \langle\!\langle\, \lambda x_0.t''^0,\ (\rho_1;...;\rho_m) \,\rangle\!\rangle$ and $U(\langle\!\langle\, \lambda x_0.t''^0,\ (\rho_1;...;\rho_m) \,\rangle\!\rangle) = \lambda x.t'^0$. We have $eval_{\text{MetaML:EnvSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:SubSOS}}(t) = \texttt{code}$, then $t \longrightarrow^{0*} \langle v^1 \rangle$ in Substitutional MetaML. Observe that $t \simeq t$. By Theorem 370, $\triangleright \langle\!\langle\, t,\ \rho_{\text{init}}^{\text{VAR}(t)} \,\rangle\!\rangle \longrightarrow^* v'$ in Environmental MetaML and $\langle v^1 \rangle \simeq v'$. Then $v' = \langle v''^1 \rangle$ and $U(\langle v''^1 \rangle) = \langle v^1 \rangle$. We have $eval_{\text{MetaML:EnvSOS}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:Sub:}\longrightarrow}(t) = n$, then $t \longrightarrow^{0*} n$ in Substitutional MetaML. Observe that $t \simeq t$. By Theorem 370, $\triangleright \langle\!\langle\, t,\ \rho_{\text{init}}^{\text{VAR}(t)} \,\rangle\!\rangle \longrightarrow^* v$ in Environmental MetaML and $n \simeq v$. Then $v = n$. We have $eval_{\text{MetaML:SusSOS}}(t) = n$.

We then show that if $eval_{\text{MetaML:EnvSOS}}(t) = a$ where $a \in \text{ANS}_{\text{MetaML}}$, then $eval_{\text{MetaML:SubSOS}}(t) = a$.

Case 1. If $eval_{\text{MetaML:EnvSOS}}(t) = \texttt{function}$, then $\triangleright \langle\!\langle\, t,\ \rho_{\text{init}}^{\text{VAR}(t)} \,\rangle\!\rangle \longrightarrow^* \langle\!\langle\, \lambda x.t'^0,\ (\rho_1;...;\rho_m) \,\rangle\!\rangle$ in Environmental MetaML. Observe that $t \simeq t$. By Theorem 371, $t \longrightarrow^{0*} v$ in Substitutional MetaML and $v \simeq \langle\!\langle\, \lambda x.t'^0,\ (\rho_1;...;\rho_m) \,\rangle\!\rangle$. Then $v = U(\langle\!\langle\, \lambda x.t'^0,\ (\rho_1;...;\rho_m) \,\rangle\!\rangle) = (\lambda x.U(t'^0))\overline{[w_{1i}/x_{1i}]}...\overline{[w_{mi}/x_{mi}]}$. We have $eval_{\text{MetaML:SubSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:EnvSOS}}(t) = \texttt{code}$, then $\triangleright \langle\!\langle\, t,\ \rho_{\text{init}}^{\text{VAR}(t)} \,\rangle\!\rangle \longrightarrow^* \langle v^1 \rangle$ in Environmental MetaML. Observe that $t \simeq t$. By Theorem 371, $t \longrightarrow^{0*} v'$ in (Substitutional) MetaML and $v' \simeq \langle v^1 \rangle$. Then $v' = \langle v''^1 \rangle$ and $v' = U(\langle v^1 \rangle)$. We have $eval_{\text{MetaML:SubSOS}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:EnvSOS}}(t) = n$, then $\triangleright \blacklozenge\, t,\ \rho_{\text{init}}^{\text{VAR}(t)} \blacklozenge \longrightarrow^* n$ in Environmental MetaML. Observe that $t \simeq t$. By Theorem 371, $t \longrightarrow^{0*} v$ in (Substitutional) MetaML and $v \simeq n$. Then $v = n$. We have $eval_{\text{MetaML:SubSOS}}(t) = n$.

We observe that $eval_{\text{MetaML:SubSOS}}(t)$ is undefined if and only if $eval_{\text{MetaML:EnvSOS}}(t)$ is undefined. Therefore, $eval_{\text{MetaML:SubSOS}}(t)$ is Kleene equal to $eval_{\text{MetaML:EnvSOS}}(t)$. □

D.4 Equivalence of Structural Operational Semantics and Reduction Semantics of Environmental MetaML

We demonstrate the equivalence of the structural operational semantics of Environmental MetaML and the reduction semantics of Environmental MetaML.

Lemma 373. *If* $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$ *and* $E \in \text{EXCT}^{i\multimap j}$, *then* $\mathscr{X} \vdash E^{i\multimap j}[c_1^i] \longmapsto^j E^{i\multimap j}[c_2^i]$.

Proof. Suppose that $c_1^i \longrightarrow^i c_2^i$ and $E \in \text{EXCT}^{i\multimap j}$. We show there exists some $E_0 \in \text{ECXT}^{k\multimap j}$ and $c_{01}^k, c_{02}^k \in \text{CONF}^k$ such that $E^{i\multimap j}[c_1^i] = E_0^{k\multimap j}[c_{01}^k]$ and $E^{i\multimap j}[c_2^i] = E_0^{k\multimap j}[c_{02}^k]$ where $c_{01}^k \longrightarrow^k c_{02}^k$. We proceed by induction on the structure of the derivation of $c_1^i \longrightarrow^i c_2^i$.

Case 1. (lambda-(i+1)). Let $c_1^{i+1} = \lambda x.c_{11}^{i+1}$, $c_2^{i+1} = \lambda x.c_{21}^{i+1}$ and $\mathscr{V} \cup \{x\} \mid \mathscr{X} \vdash c_{11}^{i+1} \longrightarrow^{i+1} c_{21}^{i+1}$. Let $E_0^{(i+1)\multimap j} = E^{(i+1)\multimap j}[\lambda x.\square]$. By the induction hypothesis, $\mathscr{X} \vdash E_0^{(i+1)\multimap j}[c_{11}^{i+1}] \longmapsto^j E_0^{(i+1)\multimap j}[c_{21}^{i+1}]$. Thus $\mathscr{X} \vdash E^{(i+1)\multimap j}[\lambda x.c_{11}^{i+1}] \longmapsto^j E^{(i+1)\multimap j}[\lambda x.c_{21}^{i+1}]$.

Case 2. (appL-i). Let $c_1^i = c_{11}^i\ c_{12}^i$, $c_2^i = c_{21}^i\ c_{12}^i$ and $\mathscr{V} \mid \mathscr{X} \vdash c_{11}^i \longrightarrow^i c_{21}^i$. Let $E_0^{i\multimap j} = E^{i\multimap j}[\square\ c_{12}^i]$. By the induction hypothesis, $\mathscr{X} \vdash E_0^{i\multimap j}[c_{11}^i] \longmapsto^j E_0^{i\multimap j}[c_{21}^i]$. Thus $\mathscr{X} \vdash E^{i\multimap j}[c_{11}^i\ c_{12}^i] \longmapsto^j E^{i\multimap j}[c_{21}^i\ c_{12}^i]$.

Case 3. (appR-i). Let $c_1^i = v_{11}^i\ c_{12}^i$, $c_2^i = v_{11}^i\ c_{22}^i$ and $\mathscr{V} \mid \mathscr{X} \vdash c_{11}^i \longrightarrow^i c_{22}^i$. Let $E_0^{i\multimap j} = E^{i\multimap j}[v_{11}^i\ \square]$. By the induction hypothesis, $\mathscr{X} \vdash E_0^{i\multimap j}[c_{12}^i] \longmapsto^j E_0^{i\multimap j}[c_{22}^i]$. Thus $\mathscr{X} \vdash E^{i\multimap j}[v_{11}^i\ c_{12}^i] \longmapsto^j E^{i\multimap j}[v_{11}^i\ c_{22}^i]$.

Case 4. (app-0). Let $c_1^0 = \triangleleft\ (\lambda x.t^0),\ (\rho, \rho^*) \triangleright\ v^0$ and $c_2^0 = \blacklozenge\, t^0,\ (\rho[x \mapsto v^0]; \rho^*) \blacklozenge$. We have $\mathscr{X} \vdash E^{0\multimap j}[c_1^0] \longmapsto^j E^{0\multimap j}[c_2^0]$ because $\mathscr{X} \vdash c_1^0\ \mathscr{R}^0\ c_2^0$.

Case 5. (run-0). Let $c_1^0 = !\langle v^1 \rangle$ and $c_2^0 = \blacklozenge\, v^1,\ (\rho_{\text{init}}^{\text{VAR}(\mathscr{X})}; \varepsilon) \blacklozenge$. We have $E^{0\multimap j}[c_1^0] \longmapsto^j E^{0\multimap j}[c_2^0]$ because $\mathscr{X} \vdash c_1^0\ \mathscr{R}^0\ c_2^0$.

Case 6. (run-i). Let $c_1^i = !c_{11}^i$, $c_2^i = !c_{21}^i$ and $\mathscr{V} \mid \mathscr{X} \vdash c_{11}^i \longrightarrow^i c_{21}^i$. Let $E_0^{i\multimap j} = E^{i\multimap j}[!\square]$. By the induction hypothesis, $\mathscr{X} \vdash E_0^{i\multimap j}[c_{11}^i] \longmapsto^j E_0^{i\multimap j}[c_{21}^i]$. Thus $\mathscr{X} \vdash E^{i\multimap j}[!c_{11}^i] \longmapsto^j E^{i\multimap j}[!c_{21}^i]$.

Case 7. (code-i). Let $c_1^i = \langle c_{11}^{i+1} \rangle$, $c_2^i = \langle c_{21}^{i+1} \rangle$ and $\mathscr{V} \mid \mathscr{X} \vdash c_{11}^{i+1} \longrightarrow^{i+1} c_{21}^{i+1}$. Let $E_0^{(i+1)\multimap j} = E^{i\multimap j}[\langle \square \rangle]$. By the induction hypothesis, $\mathscr{X} \vdash E_0^{(i+1)\multimap j}[c_{11}^{i+1}] \longmapsto^j E_0^{(i+1)\multimap j}[c_{21}^{i+1}]$. Thus $\mathscr{X} \vdash E^{i\multimap j}[\langle c_{11}^{i+1} \rangle] \longmapsto^j E^{i\multimap j}[\langle c_{21}^{i+1} \rangle]$.

Case 8. (splice-1). Let $c_1^1 = {\sim}\langle v^1 \rangle$ and $c_2^1 = v^1$. We have $\mathscr{X} \vdash E^{1-\circ j}[c_1^1] \longmapsto^j E^{1-\circ j}[c_2^1]$ because $\mathscr{X} \vdash c_1^1 \mathscr{R}^1 c_2^1$.

Case 9. (splice-(i+1)). Let $c_1^{i+1} = {\sim}c_{11}^i$, $c_2^{i+1} = {\sim}c_{21}^i$ and $\mathscr{V} \mid \mathscr{X} \vdash c_{11}^i \longrightarrow^i c_{21}^i$. Let $E_0^{i-\circ j} = E^{(i+1)-\circ j}[{\sim}\Box]$. By the induction hypothesis, $\mathscr{X} \vdash E_0^{i-\circ j}[c_{11}^i] \longmapsto^j E_0^{i-\circ j}[c_{21}^i]$. Thus $\mathscr{X} \vdash E^{(i+1)-\circ j}[{\sim}c_{11}^i] \longmapsto^j E^{(i+1)-\circ j}[{\sim}c_{21}^i]$.

Case 10. (plusL-i). Let $c_1^i = c_{11}^i + c_{12}^i$, $c_2^i = c_{21}^i + c_{12}^i$ and $\mathscr{V} \mid \mathscr{X} \vdash c_{11}^i \longrightarrow^i c_{21}^i$. Let $E_0^{i-\circ j} = E^{i-\circ j}[\Box + c_{12}^i]$. By the induction hypothesis, $\mathscr{X} \vdash E_0^{i-\circ j}[c_{11}^i] \longmapsto^j E_0^{i-\circ j}[c_{21}^i]$. Thus $\mathscr{X} \vdash E^{i-\circ j}[c_{11}^i + c_{12}^i] \longmapsto^j E^{i-\circ j}[c_{21}^i + c_{12}^i]$.

Case 11. (plusR-i). Let $c_1^i = v_{11}^i + c_{12}^i$, $c_2^i = v_{11}^i + c_{22}^i$ and $\mathscr{V} \mid \mathscr{X} \vdash c_{11}^i \longrightarrow^i c_{22}^i$. Let $E_0^{i-\circ j} = E^{i-\circ j}[v_{11}^i + \Box]$. By the induction hypothesis, $\mathscr{X} \vdash E_0^{i-\circ j}[c_{12}^i] \longmapsto^j E_0^{i-\circ j}[c_{22}^i]$. Thus $\mathscr{X} \vdash E^{i-\circ j}[v_{11}^i + c_{12}^i] \longmapsto^j E^{i-\circ j}[v_{11}^i + c_{22}^i]$.

Case 12. (plus-0). Let $c_1^0 = n_1 + n_2$ and $c_2^0 = n$ where $n = n_1 + n_2$. We have $\mathscr{X} \vdash E^{0-\circ j}[c_1^0] \longmapsto^j E^{0-\circ j}[c_2^0]$ because $\mathscr{X} \vdash c_1^0 \mathscr{R}^0 c_2^0$.

Case 13. (lam-0-env). Let $c_1^0 = \blacktriangleleft\ \lambda x.t^0,\ \rho^*\ \blacktriangleright$, $c_2^0 = \triangleleft\ \lambda x.t^0,\ \rho^*\ \triangleright$ and $\mathscr{V} \mid \mathscr{X} \vdash c_1^0 \longrightarrow^0 c_2^0$. We have $\mathscr{X} \vdash E^{0-\circ j}[c_1^0] \longmapsto^j E^{0-\circ j}[c_2^0]$ because $\mathscr{X} \vdash c_1^0 \mathscr{R}^0 c_2^0$.

Case 14. (lam-(i+1)-env). Let $c_1^{i+1} = \blacktriangleleft\ \lambda x.t^{i+1},\ (\rho;\rho^*)\ \blacktriangleright$, $c_2^{i+1} = \lambda x_N.\blacktriangleleft\ t^{i+1},\ (\rho[x \mapsto x_N][x_N \mapsto x_N];(\rho[x_N \mapsto x_N])^*)\ \blacktriangleright$ where $x_N \notin \mathscr{X}$, and $\mathscr{V} \mid \mathscr{X} \vdash c_1^{i+1} \longrightarrow^{i+1} c_2^{i+1}$. We have $\mathscr{X} \vdash E^{(i+1)-\circ j}[c_1^{i+1}] \longmapsto^j E^{(i+1)-\circ j}[c_2^{i+1}]$ because $\mathscr{X} \vdash c_1^{i+1} \mathscr{R}^{i+1} c_2^{i+1}$.

Case 15. (clov-env). Let $c_1^i = \blacktriangleleft\ \triangleleft\ \lambda x.t,\ \rho_1^*\ \triangleright,\ \rho_2^*\ \blacktriangleright$, $c_2^i = \triangleleft\ \lambda x.t,\ (\rho_1^*;\rho_2^*)\ \triangleright$ and $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$. We have $\mathscr{X} \vdash E^{i-\circ j}[c_1^i] \longmapsto^j E^{i-\circ j}[c_2^i]$ because $\mathscr{X} \vdash c_1^i \mathscr{R}^i c_2^i$.

Case 16. (den-env). Let $c_1^i = \blacktriangleleft\ \omega,\ \varepsilon\ \blacktriangleright$, $c_2^i = w$ and $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$. We have $\mathscr{X} \vdash E^{i-\circ j}[c_1^i] \longmapsto^j E^{i-\circ j}[c_2^i]$ because $\mathscr{X} \vdash c_1^i \mathscr{R}^i c_2^i$.

Case 17. (var-env). Let $c_1^i = \blacktriangleleft\ x,\ (\rho;\rho^*)\ \blacktriangleright$, $c_2^i = \blacktriangleleft\ \rho(x),\ \rho^*\ \blacktriangleright$ and $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$. We have $\mathscr{X} \vdash E^{i-\circ j}[c_1^i] \longmapsto^j E^{i-\circ j}[c_2^i]$ because $\mathscr{X} \vdash c_1^i \mathscr{R}^i c_2^i$.

Case 18. (num-env). Let $c_1^i = \blacktriangleleft\ n,\ (\rho;\rho^*)\ \blacktriangleright$, $c_2^i = n$ and $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$. We have $\mathscr{X} \vdash E^{i-\circ j}[c_1^i] \longmapsto^j E^{i-\circ j}[c_2^i]$ because $\mathscr{X} \vdash c_1^i \mathscr{R}^i c_2^i$.

Case 19. (app-env). Let $c_1^i = \blacktriangleleft\ t_1\ t_2,\ \rho^*\ \blacktriangleright$, $c_2^i = \blacktriangleleft\ t_1,\ \rho^*\ \blacktriangleright \blacktriangleleft\ t_2,\ \rho^*\ \blacktriangleright$ and $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$. We have $\mathscr{X} \vdash E^{i-\circ j}[c_1^i] \longmapsto^j E^{i-\circ j}[c_2^i]$ because $\mathscr{X} \vdash c_1^i \mathscr{R}^i c_2^i$.

Case 20. (code-env). Let $c_1^i = \blacktriangleleft\ \langle t^{i+1} \rangle,\ \rho^*\ \blacktriangleright$, $c_2^i = \langle \blacktriangleleft\ t^{i+1},\ \rho^*\ \blacktriangleright \rangle$ and $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$. We have $\mathscr{X} \vdash E^{i-\circ j}[c_1^i] \longmapsto^j E^{i-\circ j}[c_2^i]$ because $\mathscr{X} \vdash c_1^i \mathscr{R}^i c_2^i$.

Case 21. (run-env). Let $c_1^i = \blacktriangleleft\ !t^i,\ \rho^*\ \blacktriangleright$, $c_2^i = !\blacktriangleleft\ t^i,\ \rho^*\ \blacktriangleright$ and $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$. We have $\mathscr{X} \vdash E^{i-\circ j}[c_1^i] \longmapsto^j E^{i-\circ j}[c_2^i]$ because $\mathscr{X} \vdash c_1^i \mathscr{R}^i c_2^i$.

Case 22. (splice-env). Let $c_1^{i+1} = \blacklozenge\, {\sim} t^i,\ \rho^*\, \blacklozenge$, $c_2^{i+1} = {\sim}\blacklozenge\, t^i,\ \rho^*\, \blacklozenge$ and $\mathscr{V} \mid \mathscr{X} \vdash c_1^{i+1} \longrightarrow^{i+1} c_2^{i+1}$. We have $\mathscr{X} \vdash E^{(i+1)\multimap j}[c_1^{i+1}] \longmapsto^j E^{(i+1)\multimap j}[c_2^{i+1}]$ because $\mathscr{X} \vdash c_1^{i+1}\ \mathscr{R}^{i+1}\ c_2^{i+1}$.

Therefore, if $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$ and $E \in \textsc{Exct}^{i\multimap j}$, then $E^{i\multimap j}[c_1^i] \longmapsto^j E^{i\multimap j}[c_2^i]$. □

Corollary 374. *If $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$, then $\mathscr{X} \vdash c_1^i \longmapsto^i c_2^i$.*

Proof. Suppose $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$. Let $E = \Box^{i\multimap i}$ in Lemma 373. We get $\mathscr{X} \vdash \Box^{i\multimap i}[c_1^i] \longmapsto^i \Box^{i\multimap i}[c_2^i]$. Hence $\mathscr{X} \vdash c_1^i \longmapsto^i c_2^i$. □

Lemma 375. *If $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$ and $E \in \textsc{Ecxt}^{i\multimap j}$, then $\mathscr{X} \cup \textsc{Var}(E) \vdash E^{i\multimap j}[c_1^i] \longrightarrow^j E^{i\multimap j}[c_2^i]$.*

Proof. Suppose $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$ and $E \in \textsc{Ecxt}^{i\multimap j}$. We proceed by induction on the structure of the derivation $E \in \textsc{Ecxt}^{i\multimap j}$.

Case 1. ($E = \Box$). Observe that $c_1^i = \Box^{i\multimap i}[c_1^i]$ and $c_2^i = \Box^{i\multimap i}[c_2^i]$. We have $\mathscr{X} \cup \textsc{Var}(\Box) \vdash \Box^{i\multimap i}[c_1^i] \longrightarrow^i \Box^i[c_1^i]$

Case 2. ($E = E_0^{i\multimap j}[\Box\ c_0^i]$). By (appL-i), $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(c_0^i) \vdash c_1^i\ c_0^i \longrightarrow^i c_2^i\ c_0^i$. By the induction hypothesis, $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(c_0^i) \cup \textsc{Var}(E_0) \vdash E_0^{i\multimap j}[c_1^i\ c_0^i] \longrightarrow^j E_0^{i\multimap j}[c_2^i\ c_0^i]$. Thus $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(c_0^i) \cup \textsc{Var}(E_0) \vdash E_0^{i\multimap j}[\Box\ c_0^i][c_1^i] \longrightarrow^j E_0^{i\multimap j}[\Box\ c_0^i][c_2^i]$.

Case 3. ($E = E_0^{i\multimap j}[v_0^i\ \Box]$). By (appR-i), $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(v_0^i) \vdash v_0^i\ c_1^i \longrightarrow^i v_0^i\ c_2^i$. By the induction hypothesis, $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(v_0^i) \cup \textsc{Var}(E_0) \vdash E_0^{i\multimap j}[v_0^i\ c_1^i] \longrightarrow^j E_0^{i\multimap j}[v_0^i\ c_2^i]$. Thus $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(v_0^i) \cup \textsc{Var}(E_0) \vdash E_0^{i\multimap j}[v_0^i\ \Box][c_1^i] \longrightarrow^j E_0^{i\multimap j}[v_0^i\ \Box][c_2^i]$.

Case 4. ($E = E_0^{(i+1)\multimap j}[\lambda x.\Box]$). By (lambda-(i+1)), $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(x) \vdash \lambda x.c_1^{i+1} \longrightarrow^{i+1} \lambda x.c_2^{i+1}$.
By the induction hypothesis, $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(x) \cup \textsc{Var}(E_0) \vdash E_0^{(i+1)\multimap j}[\lambda x.c_1^{i+1}] \longrightarrow^j E_0^{(i+1)\multimap j}[\lambda x.c_2^{i+1}]$.
Thus $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(x) \cup \textsc{Var}(E_0) \vdash E_0^{(i+1)\multimap j}[\lambda x.\Box][c_1^{i+1}] \longrightarrow^j E_0^{(i+1)\multimap j}[\lambda x.\Box][c_2^{i+1}]$.

Case 5. ($E = E_0^{i\multimap j}[\langle\Box\rangle]$). By (code-i), $\mathscr{V} \mid \mathscr{X} \vdash \langle c_1^{i+1}\rangle \longrightarrow^i \langle c_2^{i+1}\rangle$. By the induction hypothesis, $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(E_0) \vdash E_0^{i\multimap j}[\langle c_1^{i+1}\rangle] \longrightarrow^j E_0^{i\multimap j}[\langle c_2^{i+1}\rangle]$. Thus $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(E_0) \vdash E_0^{i\multimap j}[\langle\Box\rangle][c_1^{i+1}] \longrightarrow^j E_0^{i\multimap j}[\langle\Box\rangle][c_2^{i+1}]$.

Case 6. ($E = E_0^{(i+1)\multimap j}[{\sim}\Box]$). By (splice-(i+1)), $\mathscr{V} \mid \mathscr{X} \vdash {\sim}c_1^i \longrightarrow^{i+1} {\sim}c_2^i$. By the induction hypothesis, $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(E_0) \vdash E_0^{(i+1)\multimap j}[{\sim}c_1^i] \longrightarrow^j E_0^{(i+1)\multimap j}[{\sim}c_2^i]$. Thus $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(E_0) \vdash E_0^{(i+1)\multimap j}[{\sim}\Box][c_1^i] \longrightarrow^j E_0^{(i+1)\multimap j}[{\sim}\Box][c_2^i]$.

Case 7. ($E = E_0^{i\multimap j}[!\Box]$). By (run-i), $\mathscr{V} \mid \mathscr{X} \vdash !c_1^i \longrightarrow^i !c_2^i$. By the induction hypothesis, $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(E_0) \vdash E_0^{i\multimap j}[!c_1^i] \longrightarrow^j E_0^{i\multimap j}[!c_2^i]$. Thus $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(E_0) \vdash E_0^{i\multimap j}[!\Box][c_1^i] \longrightarrow^j E_0^{i\multimap j}[!\Box][c_2^i]$.

Case 8. ($E = E_0^{i\multimap j}[\Box + c_0^i]$). By (plusL-i), $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(c_0^i) \vdash c_1^i + c_0^i \longrightarrow^i c_2^i + c_0^i$. By the induction hypothesis, $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(c_0^i) \cup \textsc{Var}(E_0) \vdash E_0^{i\multimap j}[c_1^i + c_0^i] \longrightarrow^j E_0^{i\multimap j}[c_2^i + c_0^i]$. Thus $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(c_0^i) \cup \textsc{Var}(E_0) \vdash E_0^{i\multimap j}[\Box + c_0^i][c_1^i] \longrightarrow^j E_0^{i\multimap j}[\Box + c_0^i][c_2^i]$.

Case 9. $(E = E_0^{i-\circ j}[v_0^i + \Box])$. By (plusR-i), $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(c_0^i) \vdash v_0^i + c_1^i \longrightarrow^i v_0^i + c_2^i$. By the induction hypothesis, $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(c_0^i) \cup \textsc{Var}(E_0) \vdash E_0^{i-\circ j}[v_0^i + c_1^i] \longrightarrow^j E_0^{i-\circ j}[v_0^i + c_2^i]$. Thus $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(c_0^i) \cup \textsc{Var}(E_0) \vdash E_0^{i-\circ j}[v_0^i + \Box][c_1^i] \longrightarrow^j E_0^{i-\circ j}[v_0^i + \Box][c_2^i]$.

Therefore, if $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$ and $E \in \textsc{Ecxt}^{i-\circ j}$, then $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(E) \vdash E^{i-\circ j}[c_1^i] \longrightarrow^j E^{i-\circ j}[c_2^i]$. □

Corollary 376. *If $\mathscr{X} \vdash c_1^i \longmapsto^i c_2^i$, then $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$.*

Proof. Suppose $\mathscr{X} \vdash c_1^i \longmapsto^i c_2^i$. Let $c_1^i = E_0^{j-\circ i}[c_{01}^j]$, $c_2^i = E_0^{j-\circ i}[c_{02}^j]$ and $\mathscr{X} \vdash c_{01}^j \mathscr{R}^j c_{02}^j$. Observe that $\mathscr{X} \vdash c_{01}^j \mathscr{R}^j c_{02}^j$ implies $\mathscr{V}; \mathscr{X} \vdash c_{01}^j \longrightarrow^j c_{02}^j$. Let $E = E_0^{j-\circ i}$ in Lemma 375. We get $\mathscr{V}; \mathscr{X} \vdash E_0^{j-\circ i}[c_{01}^j] \longrightarrow^i E_0^{j-\circ i}[c_{02}^j]$. Hence $\mathscr{V}; \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$. □

Theorem 377. *$\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_2^i$ if and only if $\mathscr{X} \vdash c_1^i \longmapsto^i c_2^i$.*

Proof. This theorem follows Corollaries 374 and 376 directly. □

Theorem 378. *$\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^{i*} c_2^i$ if and only if $\mathscr{X} \vdash c_1^i \longmapsto^{i*} c_2^i$.*

Proof. We first show that if $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^{i*} c_2^i$ then $\mathscr{X} \vdash c_1^i \longmapsto^{i*} c_2^i$. Suppose $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^{i(n)} c_2^i$. We proceed by induction on n.

Case 1. When $n = 0$, $c_1^i = c_2^i$. We have $\mathscr{X} \vdash c_1^i \longmapsto^{i*} c_2^i$ immediately.

Case 2. Let $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_3^i \longrightarrow^{i(n)} c_2^i$.
Given $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_3^i$, by Corollary 374, $\mathscr{X} \vdash c_1^i \longmapsto^i c_3^i$.
Given $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(c_3^i) \vdash c_3^i \longrightarrow^{i(n)} c_2^i$, by the induction hypothesis, $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(c_3^i) \vdash c_3^i \longmapsto^{i*} c_2^i$.
We get $\mathscr{X} \vdash c_1^i \longmapsto^i c_3^i \longmapsto^{i*} c_2^i$. Hence $\mathscr{X} \vdash c_1^i \longmapsto^{i*} c_2^i$.

Now we show that if $\mathscr{X} \vdash c_1^i \longmapsto^{i*} c_2^i$ then $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^{i*} c_2^i$. Suppose $\mathscr{X} \vdash c_1^i \longmapsto^{i(n)} c_2^i$. We proceed by induction on n.

Case 1. When $n = 0$, $c_1^i = c_2^i$. We have $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^{i*} c_2^i$ immediately.

Case 2. Let $\mathscr{X} \vdash c_1^i \longmapsto^i c_3^i \longmapsto^{i(n)} c_2^i$.
Given $\mathscr{X} \vdash c_1^i \longmapsto^i c_3^i$, by Corollary 376, $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_3^i$.
Given $\mathscr{X} \cup \textsc{Var}(c_3^i) \vdash c_3^i \longmapsto^{i(n)} c_2^i$, by the induction hypothesis, $\mathscr{V} \mid \mathscr{X} \cup \textsc{Var}(c_3^i) \vdash c_3^i \longrightarrow^{i*} c_2^i$.
We get $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^i c_3^i \longrightarrow^{i*} c_2^i$. Hence $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^{i*} c_2^i$.

Therefore, $\mathscr{V} \mid \mathscr{X} \vdash c_1^i \longrightarrow^{i*} c_2^i$ if and only if $\mathscr{X} \vdash c_1^i \longmapsto^{i*} c_2^i$. □

Theorem 379 (Kleene Equality of Evaluators)**.** *For any $t \in \textsc{Prgm}_{\text{MetaML}}$, $eval_{\text{MetaML:EnvSOS}}(t)$ is Kleene equal to $eval_{\text{MetaML:EnvRed}}(t)$.*

Proof. We first show if $eval_{\text{MetaML:EnvSOS}}(t) = a$ where $a \in \textsc{Ans}_{\text{MetaML}}$, then $eval_{\text{MetaML:EnvRed}}(t) = a$.

Case 1. If $eval_{\text{MetaML:EnvSOS}}(t) = \texttt{function}$, then $\emptyset \mid \text{VAR}(t) \vdash t \longrightarrow^{0*} \langle\!\langle\, \lambda x.t'^{0},\ \rho^{*} \,\rangle\!\rangle$. By Theorem 378, $\text{VAR}(t) \vdash t \longmapsto^{0*} \langle\!\langle\, \lambda x.t'^{0},\ \rho^{*} \,\rangle\!\rangle$.
We have $eval_{\text{MetaML:EnvRed}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:EnvSOS}}(t) = \texttt{code}$, then $\emptyset \mid \text{VAR}(t) \vdash t \longrightarrow^{0*} \langle v^{1} \rangle$. By Theorem 378, $\text{VAR}(t) \vdash t \longmapsto^{0*} \langle v^{1} \rangle$.
We have $eval_{\text{MetaML:EnvRed}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:EnvSOS}}(t) = n$, then $\emptyset \mid \text{VAR}(t) \vdash t \longrightarrow^{0*} n$. By Theorem 378, $\text{VAR}(t) \vdash t \longmapsto^{0*} n$.
We have $eval_{\text{MetaML:EnvRed}}(t) = n$.

We then show if $eval_{\text{MetaML:EnvRed}}(t) = a$ where $a \in \text{ANS}_{\text{MetaML}}$, then $eval_{\text{MetaML:EnvSOS}}(t) = a$.

Case 1. If $eval_{\text{MetaML:EnvRed}}(t) = \texttt{function}$, then $\text{VAR}(t) \vdash t \longmapsto^{0*} \langle\!\langle\, \lambda x.t'^{0},\ \rho^{*} \,\rangle\!\rangle$. By Theorem 378, $\emptyset \mid \text{VAR}(t) \vdash t \longrightarrow^{0*} \langle\!\langle\, \lambda x.t'^{0},\ \rho^{*} \,\rangle\!\rangle$.
We have $eval_{\text{MetaML:EnvSOS}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:EnvRed}}(t) = \texttt{code}$, then $\text{VAR}(t) \vdash t \longmapsto^{0*} \langle v^{1} \rangle$. By Theorem 378, $\emptyset \mid \text{VAR}(t) \vdash t \longrightarrow^{0*} \langle v^{1} \rangle$.
We have $eval_{\text{MetaML:EnvSOS}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:EnvRed}}(t) = n$, then $\text{VAR}(t) \vdash t \longmapsto^{0*} n$. By Theorem 378, $\emptyset \mid \text{VAR}(t) \vdash t \longrightarrow^{0*} n$.
We have $eval_{\text{MetaML:EnvSOS}}(t) = n$.

We observe that $eval_{\text{MetaML:EnvSOS}}(t)$ is undefined if and only if $eval_{\text{MetaML:EnvRed}}(t)$ is undefined. Therefore, $eval_{\text{MetaML:EnvSOS}}(t)$ is Kleene equal to $eval_{\text{MetaML:EnvRed}}(t)$. □

D.5 Equivalence of Reduction Semantics and Abstract Machine (MEK Machine) of Environmental MetaML

We demonstrate the equivalence of the reduction semantics of Environmental MetaML and the abstract machine (the MEK machine) of Environmental MetaML.

Lemma 380. $\langle i,\ E^{i-\circ 0},\ v^{i} \rangle_{\text{f}} \longmapsto^{*}_{\text{mek}} \langle i,\ E^{i-\circ 0},\ v^{i} \rangle_{\text{b}}$.

Proof. We proceed by induction on the structure of the derivation of $v^{i} \in \text{VALUE}^{i}$.

Case 1. ($x \in \text{VALUE}^{i+1}$). We immediately have $\langle i+1,\ E^{(i+1)-\circ 0},\ x \rangle_{\text{f}} \longmapsto_{\text{mek}} \langle i,\ E^{(i+1)-\circ 0},\ x \rangle_{\text{b}}$.

Case 2. ($v_1^{i+1}\ v_2^{i+1} \in \text{VALUE}^{i+1}$). We have:

$$
\begin{array}{ll}
 & \langle i+1,\, E^{(i+1)-\circ 0},\, v_1^{i+1}\, v_2^{i+1}\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mek}} & \langle i+1,\, E^{(i+1)-\circ 0}[\Box\, v_2^{i+1}],\, v_1^{i+1}\rangle_{\mathrm{f}} \\
\longmapsto^{*}_{\mathrm{mek}} & \langle i+1,\, E^{(i+1)-\circ 0}[\Box\, v_2^{i+1}],\, v_1^{i+1}\rangle_{\mathrm{b}} \quad \text{by the induction hypothesis} \\
\longmapsto_{\mathrm{mek}} & \langle i+1,\, E^{(i+1)-\circ 0}[v_1^{i+1}\, \Box],\, v_2^{i+1}\rangle_{\mathrm{f}} \\
\longmapsto^{*}_{\mathrm{mek}} & \langle i+1,\, E^{(i+1)-\circ 0}[v_1^{i+1}\, \Box],\, v_2^{i+1}\rangle_{\mathrm{b}} \quad \text{by the induction hypothesis} \\
\longmapsto_{\mathrm{mek}} & \langle i+1,\, E^{(i+1)-\circ 0},\, v_1^{i+1}\, v_2^{i+1}\rangle_{\mathrm{b}}
\end{array}
$$

Case 3. $(\triangleleft\ \lambda x.t_1^0,\, \rho^*\ \triangleright \in \mathrm{VALUE}^0)$. We immediately have $\langle 0,\, E^{0-\circ 0},\, \triangleleft\ \lambda x.t_1^0,\, \rho^*\ \triangleright\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mek}} \langle 0,\, E^{0-\circ 0},\, \triangleleft\ \lambda x.t_1^0,\, \rho^*\ \triangleright\rangle_{\mathrm{b}}$.

Case 4. $(\lambda x.v_1^{i+1} \in \mathrm{VALUE}^{i+1})$. We have:

$$
\begin{array}{ll}
 & \langle i+1,\, E^{(i+1)-\circ 0},\, \lambda x.v_1^{i+1}\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mek}} & \langle i+1,\, E^{(i+1)-\circ 0}[\lambda x.\Box],\, v_1^{i+1}\rangle_{\mathrm{f}} \\
\longmapsto^{*}_{\mathrm{mek}} & \langle i+1,\, E^{(i+1)-\circ 0}[\lambda x.\Box],\, v_1^{i+1}\rangle_{\mathrm{b}} \quad \text{by the induction hypothesis} \\
\longmapsto_{\mathrm{mek}} & \langle i+1,\, E^{(i+1)-\circ 0},\, \lambda x.v_1^{i+1}\rangle_{\mathrm{b}}
\end{array}
$$

Case 5. $(\langle v_1^{i+1}\rangle \in \mathrm{VALUE}^i)$. We have:

$$
\begin{array}{ll}
 & \langle i,\, E^{i-\circ 0},\, \langle v_1^{i+1}\rangle\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mek}} & \langle i+1,\, E^{i-\circ 0}[\langle\Box\rangle],\, v_1^{i+1}\rangle_{\mathrm{f}} \\
\longmapsto^{*}_{\mathrm{mek}} & \langle i+1,\, E^{i-\circ 0}[\langle\Box\rangle],\, v_1^{i+1}\rangle_{\mathrm{b}} \quad \text{by the induction hypothesis} \\
\longmapsto_{\mathrm{mek}} & \langle i,\, E^{i-\circ 0},\, \langle v_1^{i+1}\rangle\rangle_{\mathrm{b}}
\end{array}
$$

Case 6. $(\sim v_1^{i+1} \in \mathrm{VALUE}^{i+2})$. We have:

$$
\begin{array}{ll}
 & \langle i+2,\, E^{(i+2)-\circ 0},\, \sim v_1^{i+1}\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mek}} & \langle i+1,\, E^{(i+2)-\circ 0}[\sim\Box],\, v_1^{i+1}\rangle_{\mathrm{f}} \\
\longmapsto^{*}_{\mathrm{mek}} & \langle i+1,\, E^{(i+2)-\circ 0}[\sim\Box],\, v_1^{i+1}\rangle_{\mathrm{b}} \quad \text{by the induction hypothesis} \\
\longmapsto_{\mathrm{mek}} & \langle i+2,\, E^{(i+2)-\circ 0},\, \sim v_1^{i+1}\rangle_{\mathrm{b}}
\end{array}
$$

Case 7. $(!v_1^{i+1} \in \mathrm{VALUE}^{i+1})$. We have:

$$
\begin{array}{ll}
 & \langle i+1,\, E^{(i+1)-\circ 0},\, !v_1^{i+1}\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mek}} & \langle i+1,\, E^{(i+1)-\circ 0}[!\Box],\, v_1^{i+1}\rangle_{\mathrm{f}} \\
\longmapsto^{*}_{\mathrm{mek}} & \langle i+1,\, E^{(i+1)-\circ 0}[!\Box],\, v_1^{i+1}\rangle_{\mathrm{b}} \quad \text{by the induction hypothesis} \\
\longmapsto_{\mathrm{mek}} & \langle i+1,\, E^{(i+1)-\circ 0},\, !v_1^{i+1}\rangle_{\mathrm{b}}
\end{array}
$$

Case 8. $(n \in \mathrm{VALUE}^i)$. We immediately have $\langle i,\, E^{i-\circ 0},\, n\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mek}} \langle i,\, E^{i-\circ 0},\, n\rangle_{\mathrm{b}}$.

□

Lemma 381. *If* $\mathscr{X} \vdash c_1^i \mathscr{R}^i c_2^i$ *and* $\textsc{Var}(E[c_1]) = \mathscr{X}$*, then* $\langle i,\, E^{i-\circ 0},\, c_1^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} \langle i,\, E^{i-\circ 0},\, c_1^i\rangle_{\mathrm{r}} \longmapsto_{\mathrm{mek}} \langle i,\, E^{i-\circ 0},\, c_2^i\rangle_{\mathrm{f}}$.

Proof. We proceed by cases on $c_1^i \mathscr{R}^i c_2^i$.

Case 1. (app-0). Let $i = 0$, $c_1^0 = \triangleleft\ \lambda x.t_{11}^0,\ (\rho;\rho^*)\ \triangleright\ v_{12}^0$ and $c_2^0 = \blacktriangleleft\ t_{11}^0,\ (\rho[x \mapsto v_{12}^0];\rho^*)\ \blacktriangleright$. We have:

$$\begin{array}{ll}
 & \langle 0,\, E^{0-\circ 0},\, \triangleleft\ \lambda x.t_{11}^0,\ (\rho;\rho^*)\ \triangleright\ v_{12}^0\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0}[\square\ v_{12}^0],\, \triangleleft\ \lambda x.t_{11}^0,\ (\rho;\rho^*)\ \triangleright\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0}[\square\ v_{12}^0],\, \triangleleft\ \lambda x.t_{11}^0,\ (\rho;\rho^*)\ \triangleright\rangle_{\mathrm{b}} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0}[\triangleleft\ \lambda x.t_{11}^0,\ (\rho;\rho^*)\ \triangleright\ \square],\, v_{12}^0\rangle_{\mathrm{f}} \\
\longmapsto^*_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0}[\triangleleft\ \lambda x.t_{11}^0,\ (\rho;\rho^*)\ \triangleright\ \square],\, v_{12}^0\rangle_{\mathrm{b}} \quad \text{by Lemma 380} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0},\, \triangleleft\ \lambda x.t_{11}^0,\ (\rho;\rho^*)\ \triangleright\ v_{12}^0\rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0},\, \blacktriangleleft\ t_{11}^0,\ (\rho[x \mapsto v_{12}^0];\rho^*)\ \blacktriangleright\rangle_{\mathrm{f}}
\end{array}$$

Case 2. (run-0). Let $i = 0$, $c_1^0 = !\langle v_{11}^1\rangle$ and $c_2^0 = \blacktriangleleft\ v^1,\ (\rho_{\mathrm{init}}^{\textsc{Var}(\mathscr{X})};\varepsilon)\ \blacktriangleright$. We have:

$$\begin{array}{ll}
 & \langle 0,\, E^{0-\circ 0},\, !\langle v_{11}^1\rangle\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0}[!\square],\, \langle v_{11}^1\rangle\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mek}} & \langle 1,\, E^{0-\circ 0}[!\square][\langle\square\rangle],\, v_{11}^1\rangle_{\mathrm{f}} \\
\longmapsto^*_{\mathrm{mek}} & \langle 1,\, E^{0-\circ 0}[!\square][\langle\square\rangle],\, v_{11}^1\rangle_{\mathrm{b}} \quad \text{by Lemma 380} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0}[!\square],\, \langle v_{11}^1\rangle\rangle_{\mathrm{b}} \\
\longmapsto^*_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0},\, !\langle v_{11}^1\rangle\rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0},\, \blacktriangleleft\ v^1,\ (\rho_{\mathrm{init}}^{\textsc{Var}(E[!\langle v_{11}\rangle])};\varepsilon)\ \blacktriangleright\rangle_{\mathrm{f}}
\end{array}$$

Case 3. (splice-1). Let $i = 1$, $c_1^1 = \sim\langle v_{11}^1\rangle$ and $c_2^1 = v_{11}^1$. We have:

$$\begin{array}{ll}
 & \langle 1,\, E^{0-\circ 0},\, \sim\langle v_{11}^1\rangle\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0}[\sim\square],\, \langle v_{11}^1\rangle\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mek}} & \langle 1,\, E^{0-\circ 0}[\sim\square][\langle\square\rangle],\, v_{11}^1\rangle_{\mathrm{f}} \\
\longmapsto^*_{\mathrm{mek}} & \langle 1,\, E^{0-\circ 0}[\sim\square][\langle\square\rangle],\, v_{11}^1\rangle_{\mathrm{b}} \quad \text{by Lemma 380} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0}[\sim\square],\, \langle v_{11}^1\rangle\rangle_{\mathrm{b}} \\
\longmapsto^*_{\mathrm{mek}} & \langle 1,\, E^{0-\circ 0},\, \sim\langle v_{11}^1\rangle\rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{mek}} & \langle 1,\, E^{0-\circ 0},\, v_{11}^1\rangle_{\mathrm{f}}
\end{array}$$

Case 4. (plus-0). Let $i = 0$, $c_1^0 = n_1 + n_2$ and $c_2^0 = n$ where $n = n_1 + n_2$. We have:

$$\begin{array}{ll}
 & \langle 0,\, E^{0-\circ 0},\, n_1 + n_2\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0}[\square + n_2],\, n_1\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0}[\square + n_2],\, n_1\rangle_{\mathrm{b}} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0}[n_1 + \square],\, n_2\rangle_{\mathrm{f}} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0}[n_1 + \square],\, n_2\rangle_{\mathrm{b}} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0},\, n_1 + n_2\rangle_{\mathrm{r}} \\
\longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0},\, n\rangle_{\mathrm{f}} \quad \text{where } n = n_1 + n_2
\end{array}$$

Case 5. (other cases). Let $c_1^i = \blacklozenge\, t_{11}^i,\ \rho^*\, \blacklozenge$ and $\mathscr{X} \vdash \blacklozenge\, t_{11}^i,\ \rho^*\, \blacklozenge\, \mathscr{R}^i\ c_2^i$. We have:

$$\begin{array}{rl} & \langle i,\ E^{i-\circ 0},\ \blacklozenge\, t_{11}^i,\ \rho^*\, \blacklozenge\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mek}} & \langle i,\ E^{i-\circ 0},\ \blacklozenge\, t_{11}^i,\ \rho^*\, \blacklozenge\rangle_{\mathrm{r}} \\ \longmapsto_{\mathrm{mek}} & \langle i,\ E^{i-\circ 0},\ c_2^i\rangle_{\mathrm{f}} \end{array}$$

□

Lemma 382. *If* $c^i = E_1^{j-\circ i}[c_1^j]$ *and* $c_1^j\ \mathscr{R}^j\ c_2^j$, *then* $\langle i,\ E^{i-\circ 0},\ c^i\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mek}}^* \langle j,\ E^{i-\circ 0}E_1^{j-\circ i},\ c_1^j\rangle_{\mathrm{f}}$.

Proof. Suppose $c^i = E_1^{j-\circ i}[c_1^j]$ and $c_1^j\ \mathscr{R}^j\ c_2^j$. We want to show $\langle i,\ E^{i-\circ 0},\ E_1^{j-\circ i}[c_1^j]\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mek}}^* \langle j,\ E^{i-\circ 0}E_1^{j-\circ i},\ c_1^j\rangle$. We proceed by induction on the structure of the derivation of $E_1^{j-\circ i}$.

Case 1. ($E_1^{i-\circ i} = \square^{i-\circ i}$). We know $\langle i,\ E^{i-\circ 0},\ c^i\rangle_{\mathrm{f}} = \langle i,\ E^{i-\circ 0},\ c_1^i\rangle_{\mathrm{f}}$ and $\langle i,\ E^{i-\circ 0},\ c_1^i\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mek}}^* \langle i,\ E^{i-\circ 0},\ c_1^i\rangle_{\mathrm{f}}$.

Case 2. ($E_1^{j-\circ i} = E_{11}^{j-\circ i}\ c_{11}^i$). We know $\langle i,\ E^{i-\circ 0},\ c^i\rangle_{\mathrm{f}} = \langle i,\ E^{i-\circ 0},\ E_1^{j-\circ i}[c_1^j]\rangle_{\mathrm{f}} = \langle i,\ E^{i-\circ 0},\ (E_{11}^{j-\circ i}\ c_{11}^i)[c_1^j]\rangle_{\mathrm{f}}$. We have:

$$\begin{array}{rl} & \langle i,\ E^{i-\circ 0},\ (E_{11}^{j-\circ i}\ c_{11}^i)[c_1^j]\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mek}} & \langle i,\ E^{i-\circ 0}[\square\ c_{11}^i],\ E_{11}^{j-\circ i}[c_1^j]\rangle_{\mathrm{f}} \end{array}$$

Since $E_{11}^{j-\circ i}$ is a component of $E_1^{j-\circ i}$, by the induction hypothesis, we have $\langle i,\ E^{i-\circ 0}[\square\ c_{11}^i],\ E_{11}^{j-\circ i}[c_1^j]\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mek}}^*$ $\langle i,\ E^{i-\circ 0}[E_{11}^{j-\circ i}\ c_{11}^i],\ c_1^j\rangle_{\mathrm{f}}$.

Case 3. ($E_1^{j-\circ i} = v_{11}^i\ E_{11}^{j-\circ i}$). We know $\langle i,\ E^{i-\circ 0},\ c^i\rangle_{\mathrm{f}} = \langle i,\ E^{i-\circ 0},\ E_1^{j-\circ i}[c_1^j]\rangle_{\mathrm{f}} = \langle i,\ E^{i-\circ 0},\ (v_{11}^i\ E_{11}^{j-\circ i})[c_1^j]\rangle_{\mathrm{f}}$. We have:

$$\begin{array}{rll} & \langle i,\ E^{i-\circ 0},\ (v_{11}^i\ E_{11}^{j-\circ i})[c_1^j]\rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mek}} & \langle i,\ E^{i-\circ 0}[\square\ E_{11}^{j-\circ i}[c_1^j]],\ v_{11}^i\rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mek}}^* & \langle i,\ E^{i-\circ 0}[\square\ E_{11}^{j-\circ i}[c_1^j]],\ v_{11}^i\rangle_{\mathrm{b}} & \text{by Lemma 380} \\ \longmapsto_{\mathrm{mek}} & \langle i,\ E^{i-\circ 0}[v_{11}^i\ \square],\ E_{11}^{j-\circ i}[c_1^j]\rangle_{\mathrm{f}} & \end{array}$$

Since $E_{11}^{j-\circ i}$ is a component of $E_1^{j-\circ i}$, by the induction hypothesis, we have $\langle i,\ E^{i-\circ 0}[v_{11}^i\ \square],\ E_{11}^{j-\circ i}[c_1^j]\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mek}}^*$ $\langle i,\ E^{i-\circ 0}[v_{11}^i\ E_{11}^{j-\circ i}],\ c_1^j\rangle_{\mathrm{f}}$.

Case 4. ($E_1^{j-\circ(i+1)} = \lambda x.E_{11}^{j-\circ(i+1)}$). We know $\langle i+1,\ E^{(i+1)-\circ 0},\ c^{i+1}\rangle_{\mathrm{f}} = \langle i+1,\ E^{i+1-\circ 0},\ E_1^{j-\circ(i+1)}[c_1^j]\rangle_{\mathrm{f}} =$ $\langle i+1,\ E^{(i+1)-\circ 0},\ (\lambda x.E_{11}^{j-\circ(i+1)})[c_1^j]\rangle_{\mathrm{f}}$. We have:

$$\begin{array}{rl} & \langle i+1,\ E^{(i+1)-\circ 0},\ (\lambda x.E_{11}^{j-\circ(i+1)})[c_1^j]\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\ E^{(i+1)-\circ 0}[\lambda x.\square],\ E_{11}^{j-\circ(i+1)}[c_1^j]\rangle_{\mathrm{f}} \end{array}$$

Since $E_{11}^{j-\circ(i+1)}$ is a component of $E_1^{j-\circ(i+1)}$, by the induction hypothesis,
we have $\langle i+1,\ E^{(i+1)-\circ 0}[\lambda x.\square],\ E_{11}^{j-\circ(i+1)}[c_1^j]\rangle_{\mathrm{f}} \longmapsto_{\mathrm{mek}}^* \langle i+1,\ E^{(i+1)-\circ 0}[\lambda x.E_{11}^{j-\circ(i+1)}],\ c_1^j\rangle_{\mathrm{f}}$.

Case 5. $(E_1^{j-\circ i} = \langle E_{11}^{j-\circ(i+1)} \rangle)$. We know $\langle i,\, E^{i-\circ 0},\, c^i \rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, E_1^{j-\circ i}[c_1^j] \rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, \langle E_{11}^{j-\circ(i+1)} \rangle [c_1^j] \rangle_{\mathrm{f}}$. We have:

$$\begin{array}{rl} & \langle i,\, E^{i-\circ 0},\, \langle E_{11}^{j-\circ(i+1)} \rangle [c_1^j] \rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mek}} & \langle i,\, E^{i-\circ 0}[\langle \Box \rangle],\, E_{11}^{j-\circ(i+1)}[c_1^j] \rangle_{\mathrm{f}} \end{array}$$

Since $E_{11}^{j-\circ(i+1)}$ is a component of $E_1^{j-\circ i}$, by the induction hypothesis, we have $\langle i,\, E^{i-\circ 0}[\langle \Box \rangle],\, E_{11}^{j-\circ(i+1)}[c_1^j] \rangle_{\mathrm{f}} \longmapsto_{\mathrm{mek}}^* \langle i,\, E^{i-\circ 0}[\langle E_{11}^{j-\circ(i+1)} \rangle],\, c_1^j \rangle_{\mathrm{f}}$.

Case 6. $(E_1^{j-\circ(i+1)} = {\sim} E_{11}^{j-\circ i})$. We know:

$$\begin{array}{rl} & \langle i+1,\, E^{(i+1)-\circ 0},\, c^{i+1} \rangle_{\mathrm{f}} \\ = & \langle i+1,\, E^{(i+1)-\circ 0},\, E_1^{j-\circ(i+1)}[c_1^j] \rangle_{\mathrm{f}} \\ = & \langle i+1,\, E^{(i+1)-\circ 0},\, {\sim} E_{11}^{j-\circ i}[c_1^j] \rangle_{\mathrm{f}} \end{array}$$

We have:

$$\begin{array}{rl} & \langle i+1,\, E^{(i+1)-\circ 0},\, {\sim} E_{11}^{j-\circ i}[c_1^j] \rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\, E^{(i+1)-\circ 0}[{\sim}\Box],\, E_{11}^{j-\circ i}[c_1^j] \rangle_{\mathrm{f}} \end{array}$$

Since $E_{11}^{j-\circ i}$ is a component of $E_1^{j-\circ(i+1)}$, by the induction hypothesis, we have $\langle i+1,\, E^{(i+1)-\circ 0}[{\sim}\Box],\, E_{11}^{j-\circ i}[c_1^j] \rangle_{\mathrm{f}} \longmapsto_{\mathrm{mek}}^* \langle i+1,\, E^{(i+1)-\circ 0}[{\sim} E_{11}^{j-\circ i}],\, c_1^j \rangle_{\mathrm{f}}$.

Case 7. $(E_1^{j-\circ i} = {!} E_{11}^{j-\circ i})$. We know $\langle i,\, E^{i-\circ 0},\, c^i \rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, E_1^{j-\circ i}[c_1^j] \rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, {!} E_{11}^{j-\circ i}[c_1^j] \rangle_{\mathrm{f}}$. We have:

$$\begin{array}{rl} & \langle i,\, E^{i-\circ 0},\, {!} E_{11}^{j-\circ i}[c_1^j] \rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mek}} & \langle i,\, E^{i-\circ 0}[{!}\Box],\, E_{11}^{j-\circ i}[c_1^j] \rangle_{\mathrm{f}} \end{array}$$

Since $E_{11}^{j-\circ i}$ is a component of $E_1^{j-\circ i}$, by the induction hypothesis, we have $\langle i,\, E^{i-\circ 0}[{!}\Box],\, E_{11}^{j-\circ i}[c_1^j] \rangle_{\mathrm{f}} \longmapsto_{\mathrm{mek}}^* \langle i,\, E^{i-\circ 0}[{!} E_{11}^{j-\circ i}],\, c_1^j \rangle_{\mathrm{f}}$.

Case 8. $(E_1^{j-\circ i} = E_{11}^{j-\circ i} + c_{11}^i)$. We know $\langle i,\, E^{i-\circ 0},\, c^i \rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, E_1^{j-\circ i}[c_1^j] \rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, (E_{11}^{j-\circ i} + c_{11}^i)[c_1^j] \rangle_{\mathrm{f}}$. We have:

$$\begin{array}{rl} & \langle i,\, E^{i-\circ 0},\, (E_{11}^{j-\circ i} + c_{11}^i)[c_1^j] \rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mek}} & \langle i,\, E^{i-\circ 0}[\Box + c_{11}^i],\, E_{11}^{j-\circ i}[c_1^j] \rangle_{\mathrm{f}} \end{array}$$

Since $E_{11}^{j-\circ i}$ is a component of $E_1^{j-\circ i}$, by the induction hypothesis, we have $\langle i,\, E^{i-\circ 0}[\Box + c_{11}^i],\, E_{11}^{j-\circ i}[c_1^j] \rangle_{\mathrm{f}} \longmapsto_{\mathrm{mek}}^* \langle i,\, E^{i-\circ 0}[E_{11}^{j-\circ i} + c_{11}^i],\, c_1^j \rangle_{\mathrm{f}}$.

Case 9. $(E_1^{j-\circ i} = v_{11}^i + E_{11}^{j-\circ i})$. We know $\langle i,\, E^{i-\circ 0},\, c^i \rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, E_1^{j-\circ i}[c_1^j] \rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0},\, (v_{11}^i + E_{11}^{j-\circ i})[c_1^j] \rangle_{\mathrm{f}}$. We have:

$$\begin{array}{ll} & \langle i,\, E^{i-\circ 0},\, (v^i_{11} + E^{j-\circ i}_{11})[c^j_1]\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mek}} & \langle i,\, E^{i-\circ 0}[\Box + E^{j-\circ i}_{11}[c^j_1]],\, v^i_{11}\rangle_{\mathrm{f}} \\ \longmapsto^*_{\mathrm{mek}} & \langle i,\, E^{i-\circ 0}[\Box + E^{j-\circ i}_{11}[c^j_1]],\, v^i_{11}\rangle_{\mathrm{b}} \quad \text{by Lemma 380} \\ \longmapsto_{\mathrm{mek}} & \langle i,\, E^{i-\circ 0}[v^i_{11} + \Box],\, E^{j-\circ i}_{11}[c^j_1]\rangle_{\mathrm{f}} \end{array}$$

Since $E^{j-\circ i}_{11}$ is a component of $E^{j-\circ i}_1$, by the induction hypothesis, we have $\langle i,\, E^{i-\circ 0}[v^i_{11} + \Box],\, E^{j-\circ i}_{11}[c^j_1]\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} \langle i,\, E^{i-\circ 0}[v^i_{11} + E^{j-\circ i}_{11}],\, c^j_1\rangle_{\mathrm{f}}$.

□

Lemma 383. *If* $E^{i-\circ 0}_0[c^i_0] = E^{j-\circ 0}_1[c^j_1]$ *and* $\mathscr{X} \vdash E^{j-\circ 0}_1[c^j_1] \longmapsto E^{j-\circ 0}_1[c^j_2]$ *where* $\mathscr{X} \vdash c^j_1 \mathscr{R}^j c^j_2$ *and* $\mathrm{VAR}(E_1[E^{i-\circ 0}_0[c^i_0]]) = \mathscr{X}$*, then* $\langle i,\, E^{i-\circ 0}_0,\, c^i_0\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} \langle j,\, E^{j-\circ 0}_1,\, c^j_2\rangle_{\mathrm{f}}$.

Proof. If c^j_1 is inside c^i_0 (or the same as c^i_0), $E^{j-\circ 0}_1$ extends $E^{i-\circ 0}_0$ (or is the same as $E^{i-\circ 0}_0$). Otherwise, because c^i_0 is not reduced, it must be a value.

Case 1. Suppose c^j_1 is inside c^i_0 (or the same as c^i_0). Let $c^i_0 = E^{j-\circ i}_2[c^j_1]$. Then $E^{j-\circ 0}_1 = E^{i-\circ 0}_0 E^{j-\circ i}_2$. We have $\langle i,\, E^{i-\circ 0}_0,\, c^i_0\rangle_{\mathrm{f}} = \langle i,\, E^{i-\circ 0}_0,\, E^{j-\circ i}_2[c^j_1]\rangle_{\mathrm{f}}$. By Lemma 382, $\langle i,\, E^{i-\circ 0}_0,\, E^{j-\circ i}_2[c^j_1]\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} \langle j,\, E^{i-\circ 0}_0 E^{j-\circ i}_2,\, c^j_1\rangle_{\mathrm{f}}$. By Lemma 381, $\langle j,\, E^{i-\circ 0}_0 E^{j-\circ i}_2,\, c^j_1\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} \langle j,\, E^{i-\circ 0}_0 E^{j-\circ i}_2,\, c^j_1\rangle_{\mathrm{r}} \longmapsto_{\mathrm{mek}} \langle j,\, E^{i-\circ 0}_0 E^{j-\circ i}_2,\, c^j_2\rangle_{\mathrm{f}}$.

Case 2. Otherwise, $c^i_0 \in \mathrm{VALUE}^i$. By Lemma 380, $\mathscr{X} \vdash \langle i,\, E^{i-\circ 0}_0,\, c^i_0\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} \langle i,\, E^{i-\circ 0}_0,\, c^i_0\rangle_{\mathrm{b}}$. We prove the following statement by induction on the structure of the derivation of $E^{i-\circ 0}_0 \in \mathrm{ECXT}^{i-\circ 0}$.

Statement: If $E^{i-\circ 0}_0[c^i_0] = E^{j-\circ 0}_1[c^j_1]$ where $c^i_0 \in \mathrm{VALUE}^i$ and $\mathscr{X} \vdash E^{j-\circ 0}_1[c^j_1] \longmapsto E^{j-\circ 0}_1[c^j_2]$ where $\mathscr{X} \vdash c^j_1 \mathscr{R}^j c^j_2$, then $\langle i,\, E^{i-\circ 0}_0,\, c^i_0\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{mek}} \langle j,\, E^{j-\circ 0}_1,\, c^j_2\rangle_{\mathrm{f}}$.

Case i. ($E^{0-\circ 0}_0 = \Box^{0-\circ 0}$). This case is vacuous.

Case ii. ($E^{i-\circ 0}_0 = E^{i-\circ 0}_2[\Box\, c^i_{22}]$). We have:

$$\begin{array}{ll} & \langle i,\, E^{i-\circ 0}_0,\, c^i_0\rangle_{\mathrm{b}} \\ = & \langle i,\, E^{i-\circ 0}_2[\Box\, c^i_{22}],\, c^i_0\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mek}} & \langle i,\, E^{i-\circ 0}_2[c^i_0\, \Box],\, c^i_{22}\rangle_{\mathrm{f}} \end{array}$$

Case a. If $\mathscr{X} \vdash c^i_{22}\ \mathscr{R}^i\ c^i_{23}$, then $c^i_1 = c^i_{22}$ and $c^i_2 = c^i_{23}$.
By Lemma 381, $\langle i,\, E^{i-\circ 0}_2[c^i_0\, \Box],\, c^i_{22}\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} \langle i,\, E^{i-\circ 0}_2[c^i_0\, \Box],\, c^i_2\rangle_{\mathrm{f}}$.

Case b. If $\mathscr{X} \vdash c^i_{22}\ \mathscr{R}^i$ and $c^i_{22} \in \mathrm{VALUE}^i$.

Case 1. If $c_{22} \in \mathrm{VALUE}^0$. We have:

$$\begin{array}{ll} & \langle 0,\, E^{0-\circ 0}_2[c^0_0\, \Box],\, c^0_{22}\rangle_{\mathrm{f}} \\ \longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0}_2[c^0_0\, \Box],\, c^0_{22}\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mek}} & \langle 0,\, E^{0-\circ 0}_2,\, c^0_0\, c^0_{22}\rangle_{\mathrm{r}} \end{array}$$

Then $c^0_1 = c^0_0\, c^0_{22}$. By Lemma 381, $\langle 0,\, E^{0-\circ 0}_2,\, c^0_0\, c^0_{22}\rangle_{\mathrm{r}} \longmapsto_{\mathrm{mek}} \langle 0,\, E^{0-\circ 0}_2,\, c^0_2\rangle_{\mathrm{f}}$.

Case 2. If $c_{22} \in \text{VALUE}^{i+1}$. We have:

$$\begin{array}{lll} & \langle i+1,\ E_2^{(i+1)\multimap 0}[c_0^{i+1}\ \Box],\ c_{22}^{i+1}\rangle_{\mathrm{f}} & \\ \longmapsto^*_{\mathrm{mek}} & \langle i+1,\ E_2^{(i+1)\multimap 0}[c_0^{i+1}\ \Box],\ c_{22}^{i+1}\rangle_{\mathrm{b}} & \text{by Lemma 380} \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\ E_2^{(i+1)\multimap 0},\ c_0^{i+1}\ c_{22}^{i+1}\rangle_{\mathrm{b}} & \end{array}$$

We know $c_0^{i+1}\ c_{22}^{i+1} \in \text{VALUE}^{i+1}$. Since $E_2^{(i+1)\multimap 0}$ is a component of $E_0^{(i+1)\multimap 0}$, by the induction hypothesis, $\langle i+1,\ E_2^{(i+1)\multimap 0},\ c_0^{i+1}\ c_{22}^{i+1}\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{mek}} \langle j,\ E_1^{j\multimap 0},\ c_2^j\rangle_{\mathrm{f}}$.

Case c. If $\mathscr{X} \vdash c_{22}^i \not\mathscr{R}^i$ and $c_{22}^i \notin \text{VALUE}^i$, then $c_{22}^i = E_3^{j\multimap i}[c_1^j]$. Hence $E_1^{j\multimap 0} = E_2^{i\multimap 0}[c_0^i\ E_3^{j\multimap i}]$. We have:

$$\begin{array}{lll} & \langle i+1,\ E_2^{(i+1)\multimap 0}[c_0^{i+1}\ \Box],\ c_{22}^{i+1}\rangle_{\mathrm{f}} & \\ & \langle i,\ E_2^{i\multimap 0}[c_0^i\ \Box],\ c_{22}^i\rangle_{\mathrm{f}} & \\ = & \langle i,\ E_2^{i\multimap 0}[c_0^i\ \Box],\ E_3^{j\multimap i}[c_1^j]\rangle_{\mathrm{f}} & \\ \longmapsto^*_{\mathrm{mek}} & \langle j,\ E_2^{i\multimap 0}[c_0^i\ E_3^{j\multimap i}],\ c_1^j\rangle_{\mathrm{f}} & \text{by Lemma 382} \\ \longmapsto^*_{\mathrm{mek}} & \langle j,\ E_2^{i\multimap 0}[c_0^i\ E_3^{j\multimap i}],\ c_2^j\rangle_{\mathrm{f}} & \text{by Lemma 381} \end{array}$$

Case iii. $(E_0^{i\multimap 0} = E_2^{i\multimap 0}[v_{21}^i\ \Box])$.

Case a. If $c_0 \in \text{VALUE}^0$. We have:

$$\begin{array}{lll} & \langle 0,\ E_0^{0\multimap 0},\ c_0^0\rangle_{\mathrm{b}} & \\ = & \langle 0,\ E_2^{0\multimap 0}[v_{21}^0\ \Box],\ c_0^0\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{mek}} & \langle 0,\ E_2^{0\multimap 0},\ v_{21}^0\ c_0^0\rangle_{\mathrm{r}} & \end{array}$$

Then $c_1^0 = v_{21}^0\ c_0^0$. By Lemma 381, $\langle 0,\ E_2^{0\multimap 0},\ v_{21}^0\ c_0^0\rangle_{\mathrm{r}} \longmapsto_{\mathrm{mek}} \langle 0,\ E_2^{0\multimap 0},\ c_2^0\rangle_{\mathrm{f}}$.

Case b. If $c_0 \in \text{VALUE}^{i+1}$. We have:

$$\begin{array}{lll} & \langle i+1,\ E_0^{(i+1)\multimap 0},\ c_0^{i+1}\rangle_{\mathrm{b}} & \\ = & \langle i+1,\ E_2^{(i+1)\multimap 0}[v_{21}^{i+1}\ \Box],\ c_0^{i+1}\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\ E_2^{(i+1)\multimap 0},\ v_{21}^{i+1}\ c_0^{i+1}\rangle_{\mathrm{b}} & \end{array}$$

We know $v_{21}^{i+1}\ c_0^{i+1} \in \text{VALUE}^{i+1}$. Since $E_2^{(i+1)\multimap 0}$ is a component of $E_0^{(i+1)\multimap 0}$, by the induction hypothesis, $\langle i+1,\ E_2^{(i+1)\multimap 0},\ v_{21}^{i+1}\ c_0^{i+1}\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{mek}} \langle j,\ E_1^{j\multimap 0},\ c_2^j\rangle_{\mathrm{f}}$.

Case iv. $(E_0^{(i+1)\multimap 0} = E_2^{(i+1)\multimap 0}[\lambda x.\Box])$. We have:

$$\begin{array}{lll} & \langle i+1,\ E_0^{(i+1)\multimap 0},\ c_0^{i+1}\rangle_{\mathrm{b}} & \\ = & \langle i+1,\ E_2^{(i+1)\multimap 0}[\lambda x.\Box],\ c_0^{i+1}\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\ E_2^{(i+1)\multimap 0},\ \lambda x.c_0^{i+1}\rangle_{\mathrm{b}} & \end{array}$$

We know $\lambda x.c_0^{i+1} \in \text{VALUE}^{i+1}$. Since $E_2^{(i+1)\multimap 0}$ is a component of $E_0^{(i+1)\multimap 0}$, by the induction hypothesis, $\langle i+1,\ E_2^{(i+1)\multimap 0},\ \lambda x.c_0^{i+1}\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{mek}} \langle j,\ E_1^{j\multimap 0},\ c_2^j\rangle_{\mathrm{f}}$.

Case v. $(E_0^{(i+1)\multimap 0} = E_2^{i\multimap 0}[\langle\Box\rangle])$. We have:

$$\begin{array}{lll} & \langle i+1,\ E_0^{(i+1)\multimap 0},\ c_0^{i+1}\rangle_{\mathrm{b}} & \\ = & \langle i+1,\ E_2^{i\multimap 0}[\langle\Box\rangle],\ c_0^{i+1}\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{mek}} & \langle i,\ E_2^{i\multimap 0},\ \langle c_0^{i+1}\rangle\rangle_{\mathrm{b}} & \end{array}$$

We know $\langle c_0^{i+1} \rangle \in \text{VALUE}^i$. Since $E_2^{i\multimap 0}$ is a component of $E_0^{(i+1)\multimap 0}$, by the induction hypothesis, $\langle i,\, E_2^{i\multimap 0},\, \langle c_0^{i+1} \rangle \rangle_{\text{b}} \longmapsto_{\text{mek}}^* \langle j,\, E_1^{j\multimap 0},\, c_2^j \rangle_{\text{f}}$.

Case vi. $(E_0^{i\multimap 0} = E_2^{(i+1)\multimap 0}[\sim\Box])$.

Case a. If $c_0 \in \text{VALUE}^0$. We have:

$$\begin{array}{ll} & \langle 0,\, E_0^{0\multimap 0},\, c_0^0 \rangle_{\text{b}} \\ = & \langle 0,\, E_2^{1\multimap 0}[\sim\Box],\, c_0^0 \rangle_{\text{b}} \\ \longmapsto_{\text{mek}} & \langle 1,\, E_2^{1\multimap 0},\, \sim c_0^0 \rangle_{\text{r}} \end{array}$$

Then $c_1^1 = \sim c_0^0$. By Lemma 381, $\langle 1,\, E_2^{1\multimap 0},\, \sim c_0^0 \rangle_{\text{r}} \longmapsto_{\text{mek}} \langle 1,\, E_2^{1\multimap 0},\, c_2^1 \rangle_{\text{f}}$.

Case b. If $c_0 \in \text{VALUE}^{i+1}$. We have:

$$\begin{array}{ll} & \langle i+1,\, E_0^{(i+1)\multimap 0},\, c_0^{i+1} \rangle_{\text{b}} \\ = & \langle i+1,\, E_2^{(i+2)\multimap 0}[\sim\Box],\, c_0^{i+1} \rangle_{\text{b}} \\ \longmapsto_{\text{mek}} & \langle i+2,\, E_2^{(i+2)\multimap 0},\, \sim c_0^{i+1} \rangle_{\text{b}} \end{array}$$

We know $\sim c_0^{i+1} \in \text{VALUE}^{i+2}$. Since $E_2^{(i+2)\multimap 0}$ is a component of $E_0^{(i+1)\multimap 0}$, by the induction hypothesis, $\langle i+2,\, E_2^{(i+2)\multimap 0},\, \sim c_0^{i+1} \rangle_{\text{b}} \longmapsto_{\text{mek}}^* \langle j,\, E_1^{j\multimap 0},\, c_2^j \rangle_{\text{f}}$.

Case vii. $(E_0^{i\multimap 0} = E_2^{i\multimap 0}[!\Box])$.

Case a. If $c_0 \in \text{VALUE}^0$. We have:

$$\begin{array}{ll} & \langle 0,\, E_0^{0\multimap 0},\, c_0^0 \rangle_{\text{b}} \\ = & \langle 0,\, E_2^{0\multimap 0}[!\Box],\, c_0^0 \rangle_{\text{b}} \\ \longmapsto_{\text{mek}} & \langle 0,\, E_2^{0\multimap 0},\, !c_0^0 \rangle_{\text{r}} \end{array}$$

Then $c_1^0 = !c_0^0$. By Lemma 381, $\langle 0,\, E_2^{0\multimap 0},\, !c_0^0 \rangle_{\text{r}} \longmapsto_{\text{mek}} \langle 0,\, E_2^{0\multimap 0},\, c_2^0 \rangle_{\text{f}}$.

Case b. If $c_0 \in \text{VALUE}^{i+1}$. We have:

$$\begin{array}{ll} & \langle i+1,\, E_0^{(i+1)\multimap 0},\, c_0^{i+1} \rangle_{\text{b}} \\ = & \langle i+1,\, E_2^{(i+1)\multimap 0}[!\Box],\, c_0^{i+1} \rangle_{\text{b}} \\ \longmapsto_{\text{mek}} & \langle i+1,\, E_2^{(i+1)\multimap 0},\, !c_0^{i+1} \rangle_{\text{b}} \end{array}$$

We know $!c_0^{i+1} \in \text{VALUE}^{i+1}$. Since $E_2^{(i+1)\multimap 0}$ is a component of $E_0^{(i+1)\multimap 0}$, by the induction hypothesis, $\langle i+1,\, E_2^{(i+1)\multimap 0},\, !c_0^{i+1} \rangle_{\text{b}} \longmapsto_{\text{mek}}^* \langle j,\, E_1^{j\multimap 0},\, c_2^j \rangle_{\text{f}}$.

Case viii. $(E_0^{i\multimap 0} = E_2^{i\multimap 0}[\Box + c_{22}^i])$. We have:

$$\begin{array}{ll} & \langle i,\, E_0^{i\multimap 0},\, c_0^i \rangle_{\text{b}} \\ = & \langle i,\, E_2^{i\multimap 0}[\Box + c_{22}^i],\, c_0^i \rangle_{\text{b}} \\ \longmapsto_{\text{mek}} & \langle i,\, E_2^{i\multimap 0}[c_0^i + \Box],\, c_{22}^i \rangle_{\text{f}} \end{array}$$

Case a. If $\mathscr{X} \vdash c_{22}^i \; \mathscr{R}^i \; c_{23}^i$, then $c_1^i = c_{22}^i$ and $c_2^i = c_{23}^i$.
By Lemma 381, $\langle i,\, E_2^{i\multimap 0}[c_0^i + \Box],\, c_{22}^i \rangle_{\text{f}} \longmapsto_{\text{mek}}^* \langle i,\, E_2^{i\multimap 0}[c_0^i + \Box],\, c_2^i \rangle_{\text{f}}$.

Case b. If $\mathscr{X} \vdash c_{22}^i \;\; \mathscr{R}^i$ and $c_{22}^i \in \text{VALUE}^i$.

Case 1. If $c_{22} \in \text{VALUE}^0$. We have:

$$\begin{array}{ll} & \langle 0,\, E_2^{0\multimap 0}[c_0^0 + \Box],\, c_{22}^0 \rangle_{\text{f}} \\ \longmapsto_{\text{mek}} & \langle 0,\, E_2^{0\multimap 0}[c_0^0 + \Box],\, c_{22}^0 \rangle_{\text{b}} \\ \longmapsto_{\text{mek}} & \langle 0,\, E_2^{0\multimap 0},\, c_0^0 + c_{22}^0 \rangle_{\text{r}} \end{array}$$

Then $c_1^0 = c_0^0 + c_{22}^0$. By Lemma 381, $\langle 0,\, E_2^{0\multimap 0},\, c_0^0 + c_{22}^0\rangle_{\mathrm{r}} \longmapsto_{\mathrm{mek}} \langle 0,\, E_2^{0\multimap 0},\, c_2^0\rangle_{\mathrm{f}}$.

Case 2. If $c_{22} \in \textsc{Value}^{i+1}$. We have:

$$\begin{array}{lll} & \langle i+1,\, E_2^{(i+1)\multimap 0}[c_0^{i+1}+\Box],\, c_{22}^{i+1}\rangle_{\mathrm{f}} & \\ \longmapsto^*_{\mathrm{mek}} & \langle i+1,\, E_2^{(i+1)\multimap 0}[c_0^{i+1}+\Box],\, c_{22}^{i+1}\rangle_{\mathrm{b}} & \text{by Lemma 380} \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\, E_2^{(i+1)\multimap 0},\, c_0^{i+1}+c_{22}^{i+1}\rangle_{\mathrm{b}} & \end{array}$$

We know $c_0^{i+1} + c_{22}^{i+1} \in \textsc{Value}^{i+1}$. Since $E_2^{(i+1)\multimap 0}$ is a component of $E_0^{(i+1)\multimap 0}$, by the induction hypothesis, $\langle i+1,\, E_2^{(i+1)\multimap 0},\, c_0^{i+1} + c_{22}^{i+1}\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{mek}} \langle j,\, E_1^{j\multimap 0},\, c_2^j\rangle_{\mathrm{f}}$.

Case c. If $\mathscr{X} \vdash c_{22}^i \;\not\!\!\mathscr{R}^i$ and $c_{22}^i \notin \textsc{Value}^i$, then $c_{22}^i = E_3^{j\multimap i}[c_1^j]$. Hence $E_1^{j\multimap 0} = E_2^{i\multimap 0}[c_0^i + E_3^{j\multimap i}]$. We have:

$$\begin{array}{lll} & \langle i+1,\, E_2^{(i+1)\multimap 0}[c_0^{i+1}+\Box],\, c_{22}^{i+1}\rangle_{\mathrm{f}} & \\ & \langle i,\, E_2^{i\multimap 0}[c_0^i+\Box],\, c_{22}^i\rangle_{\mathrm{f}} & \\ = & \langle i,\, E_2^{i\multimap 0}[c_0^i+\Box],\, E_3^{j\multimap i}[c_1^j]\rangle_{\mathrm{f}} & \\ \longmapsto^*_{\mathrm{mek}} & \langle j,\, E_2^{i\multimap 0}[c_0^i+E_3^{j\multimap i}],\, c_1^j\rangle_{\mathrm{f}} & \text{by Lemma } \mathbf{??} \\ \longmapsto^*_{\mathrm{mek}} & \langle j,\, E_2^{i\multimap 0}[c_0^i+E_3^{j\multimap i}],\, c_2^j\rangle_{\mathrm{f}} & \text{by Lemma 381} \end{array}$$

Case ix. $(E_0^{i\multimap 0} = E_2^{i\multimap 0}[v_{21}^i + \Box])$.

Case a. If $c_0 \in \textsc{Value}^0$. We have:

$$\begin{array}{lll} & \langle 0,\, E_0^{0\multimap 0},\, c_0^0\rangle_{\mathrm{b}} & \\ = & \langle 0,\, E_2^{0\multimap 0}[v_{21}^0+\Box],\, c_0^0\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{mek}} & \langle 0,\, E_2^{0\multimap 0},\, v_{21}^0+c_0^0\rangle_{\mathrm{r}} & \end{array}$$

Then $c_1^0 = v_{21}^0 + c_0^0$. By Lemma 381, $\langle 0,\, E_2^{0\multimap 0},\, v_{21}^0 + c_0^0\rangle_{\mathrm{r}} \longmapsto_{\mathrm{mek}} \langle 0,\, E_2^{0\multimap 0},\, c_2^0\rangle_{\mathrm{f}}$.

Case b. If $c_0 \in \textsc{Value}^{i+1}$. We have:

$$\begin{array}{lll} & \langle i+1,\, E_0^{(i+1)\multimap 0},\, c_0^{i+1}\rangle_{\mathrm{b}} & \\ = & \langle i+1,\, E_2^{(i+1)\multimap 0}[v_{21}^{i+1}+\Box],\, c_0^{i+1}\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\, E_2^{(i+1)\multimap 0},\, v_{21}^{i+1}+c_0^{i+1}\rangle_{\mathrm{b}} & \end{array}$$

We know $v_{21}^{i+1} + c_0^{i+1} \in \textsc{Value}^{i+1}$. Since $E_2^{(i+1)\multimap 0}$ is a component of $E_0^{(i+1)\multimap 0}$, by the induction hypothesis, $\langle i+1,\, E_2^{(i+1)\multimap 0},\, v_{21}^{i+1}+c_0^{i+1}\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{mek}} \langle j,\, E_1^{j\multimap 0},\, c_2^j\rangle_{\mathrm{f}}$.

□

Lemma 384. *If* $v^0 = E^{i\multimap 0}[c^i]$, *then* $\langle i,\, E^{i\multimap 0},\, c^i\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{mek}} v^0$.

Proof. Suppose $v^0 = E^{i\multimap 0}[c^i]$. We know $c^i \in \textsc{Value}^i$. We proceed by induction on the structure of the derivation of $E^{i\multimap 0} \in \textsc{Ecxt}^{i\multimap 0}$. □

Case 1. $(E^{0\multimap 0} = \Box^{0\multimap 0})$. Then $v^0 = c^0$. We have $\langle 0,\, \Box^{0\multimap 0},\, v^0\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mek}} v^0$.

Case 2. ($E^{i-\circ 0} = E_1^{i-\circ 0}[\Box\ c_{12}^i]$). Then $v^0 = E_1^{i-\circ 0}[c^i\ c_{12}^i]$. We know $c^i \in \text{VALUE}^i$, $c_{12}^i \in \text{VALUE}^i$ and $i \geq 1$. Let's use $i+1$ instead of i. We have:

$$\begin{array}{rll} & \langle i+1,\ E_1^{(i+1)-\circ 0}[\Box\ c_{12}^{i+1}],\ c^{i+1}\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\ E_1^{(i+1)-\circ 0}[c^{i+1}\ \Box],\ c_{12}^{i+1}\rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mek}}^* & \langle i+1,\ E_1^{(i+1)-\circ 0}[c^{i+1}\ \Box],\ c_{12}^{i+1}\rangle_{\mathrm{b}} & \text{by Lemma 380} \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\ E_1^{(i+1)-\circ 0},\ c^{i+1}\ c_{12}^{i+1}\rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mek}}^* & \langle i+1,\ E_1^{(i+1)-\circ 0},\ c^{i+1}\ c_{12}^{i+1}\rangle_{\mathrm{b}} & \text{by Lemma 380} \end{array}$$

Since $E_1^{(i+1)-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i+1,\ E_1^{(i+1)-\circ 0},\ c^{i+1}\ c_{12}^{i+1}\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mek}}^* v^0$.

Case 3. ($E^{i-\circ 0} = E_1^{i-\circ 0}[v_{11}^i\ \Box]$). Then $v^0 = E_1^{i-\circ 0}[v_{11}^i\ c^i]$. We know $c^i \in \text{VALUE}^i$ and $i \geq 1$. Let's use $i+1$ instead of i. We have:

$$\begin{array}{rll} & \langle i+1,\ E_1^{(i+1)-\circ 0}[v_{11}^{i+1}\ \Box],\ c^{i+1}\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\ E_1^{(i+1)-\circ 0},\ v_{11}^{i+1}\ c^{i+1}\rangle_{\mathrm{f}} & \\ \longmapsto_{\mathrm{mek}}^* & \langle i+1,\ E_1^{(i+1)-\circ 0},\ v_{11}^{i+1}\ c^{i+1}\rangle_{\mathrm{b}} & \text{by Lemma 380} \end{array}$$

Since $E_1^{(i+1)-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i+1,\ E_1^{(i+1)-\circ 0},\ v_{11}^{i+1}\ c^{i+1}\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mek}}^* v^0$.

Case 4. ($E^{(i+1)-\circ 0} = E_1^{(i+1)-\circ 0}[\lambda x.\Box]$). Then $v^0 = E_1^{(i+1)-\circ 0}[\lambda x.c^{i+1}]$. We know $c^{i+1} \in \text{VALUE}^{i+1}$. We have:

$$\begin{array}{rl} & \langle i+1,\ E_1^{(i+1)-\circ 0}[\lambda x.\Box],\ c^{i+1}\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\ E_1^{(i+1)-\circ 0},\ \lambda x.c^{i+1}\rangle_{\mathrm{b}} \end{array}$$

Since $E_1^{(i+1)-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i+1,\ E_1^{(i+1)-\circ 0},\ \lambda x.c^{i+1}\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mek}}^* v^0$.

Case 5. ($E^{(i+1)-\circ 0} = E_1^{i-\circ 0}[\langle\Box\rangle]$). Then $v^0 = E_1^{i-\circ 0}[\langle c^{i+1}\rangle]$. We know $c^{i+1} \in \text{VALUE}^{i+1}$. We have:

$$\begin{array}{rl} & \langle i+1,\ E_1^{i-\circ 0}[\langle\Box\rangle],\ c^{i+1}\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mek}} & \langle i,\ E_1^{i-\circ 0},\ \langle c^{i+1}\rangle\rangle_{\mathrm{b}} \end{array}$$

Since $E_1^{i-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i,\ E_1^{i-\circ 0},\ \langle c^{i+1}\rangle\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mek}}^* v^0$.

Case 6. ($E^{i-\circ 0} = E_1^{(i+1)-\circ 0}[\sim\Box]$). Then $v^0 = E_1^{(i+1)-\circ 0}[\sim c^i]$. We know $c^i \in \text{VALUE}^i$ and $i \geq 1$. Let use $i+1$ instead of i and $i+2$ instead of $i+1$. We have:

$$\begin{array}{rl} & \langle i+1,\, E_1^{(i+2)-\circ 0}[\sim\Box],\, c^{i+1}\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mek}} & \langle i+2,\, E_1^{(i+2)-\circ 0},\, \sim c^{i+1}\rangle_{\mathrm{b}} \end{array}$$

Since $E_1^{(i+2)-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i+2,\, E_1^{(i+2)-\circ 0},\, \sim c^{i+1}\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{mek}} v^0$.

Case 7. ($E^{i-\circ 0} = E_1^{i-\circ 0}[!\Box]$). Then $v^0 = E_1^{i-\circ 0}[!c^i]$. We know $c^i \in \textsc{Value}^i$ and $i \geq 1$. Let use $i+1$ instead of i. We have:

$$\begin{array}{rl} & \langle i+1,\, E_1^{(i+1)-\circ 0}[!\Box],\, c^{i+1}\rangle_{\mathrm{b}} \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\, E_1^{(i+1)-\circ 0},\, !c^{i+1}\rangle_{\mathrm{b}} \end{array}$$

Since $E_1^{(i+1)-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i+1,\, E_1^{(i+1)-\circ 0},\, !c^{i+1}\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{mek}} v^0$.

Case 8. ($E^{i-\circ 0} = E_1^{i-\circ 0}[\Box + c_{12}^i]$). Then $v^0 = E_1^{i-\circ 0}[c^i + c_{12}^i]$. We know $c^i \in \textsc{Value}^i$, $c_{12}^i \in \textsc{Value}^i$ and $i \geq 1$. Let's use $i+1$ instead of i. We have:

$$\begin{array}{rll} & \langle i+1,\, E_1^{(i+1)-\circ 0}[\Box + c_{12}^{i+1}],\, c^{i+1}\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\, E_1^{(i+1)-\circ 0}[c^{i+1} + \Box],\, c_{12}^{i+1}\rangle_{\mathrm{f}} & \\ \longmapsto^*_{\mathrm{mek}} & \langle i+1,\, E_1^{(i+1)-\circ 0}[c^{i+1} + \Box],\, c_{12}^{i+1}\rangle_{\mathrm{b}} & \text{by Lemma 380} \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\, E_1^{(i+1)-\circ 0},\, c^{i+1} + c_{12}^{i+1}\rangle_{\mathrm{f}} & \\ \longmapsto^*_{\mathrm{mek}} & \langle i+1,\, E_1^{(i+1)-\circ 0},\, c^{i+1} + c_{12}^{i+1}\rangle_{\mathrm{b}} & \text{by Lemma 380} \end{array}$$

Since $E_1^{(i+1)-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i+1,\, E_1^{(i+1)-\circ 0},\, c^{i+1} + c_{12}^{i+1}\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{mek}} v^0$.

Case 9. ($E^{i-\circ 0} = E_1^{i-\circ 0}[v_{11}^i + \Box]$). Then $v^0 = E_1^{i-\circ 0}[v_{11}^i + c^i]$. We know $c^i \in \textsc{Value}^i$ and $i \geq 1$. Let's use $i+1$ instead of i. We have:

$$\begin{array}{rll} & \langle i+1,\, E_1^{(i+1)-\circ 0}[v_{11}^{i+1} + \Box],\, c^{i+1}\rangle_{\mathrm{b}} & \\ \longmapsto_{\mathrm{mek}} & \langle i+1,\, E_1^{(i+1)-\circ 0},\, v_{11}^{i+1} + c^{i+1}\rangle_{\mathrm{f}} & \\ \longmapsto^*_{\mathrm{mek}} & \langle i+1,\, E_1^{(i+1)-\circ 0},\, v_{11}^{i+1} + c^{i+1}\rangle_{\mathrm{b}} & \text{by Lemma 380} \end{array}$$

Since $E_1^{(i+1)-\circ 0}$ is a component of $E^{(i+1)-\circ 0}$, by the induction hypothesis, we have $\langle i+1,\, E_1^{(i+1)-\circ 0},\, v_{11}^{i+1} + c^{i+1}\rangle_{\mathrm{b}} \longmapsto^*_{\mathrm{mek}} v^0$.

Lemma 385. *If* $v^0 = E^{i-\circ 0}[c^i]$, *then* $\langle i,\, E^{i-\circ 0},\, c^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} v^0$.

Proof. Suppose $v^0 = E^{i-\circ 0}[c^i]$. Then $c^i \in \textsc{Value}^i$. By Lemma 380, $\langle i,\, E^{i-\circ 0},\, c^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} \langle i,\, E^{i-\circ 0},\, c^i\rangle_{\mathrm{b}}$. By Lemma 384, $\langle i,\, E^{i-\circ 0},\, c^i\rangle_{\mathrm{b}} \longmapsto_{\mathrm{mek}} v^0$. Hence $\langle i,\, E^{i-\circ 0},\, c^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} v^0$. □

Lemma 386. *If* $\triangleright\, E^{i-\circ 0}[c_1^i] \longmapsto^* v_2^0$, *then* $\langle i,\, E^{i-\circ 0},\, c_1^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} v_2^0$.

Proof. Suppose $\triangleright\, E^{i\multimap 0}[c_1^i] \longmapsto^{(n)} v_2^0$. We proceed by induction on n.

Case 1. When $n = 0$, $v_2^0 = E^{i\multimap 0}[c_1^i]$. By Lemma 385, $\langle i,\, E^{i\multimap 0},\, c_1^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} v_2^0$.

Case 2. Let $E^{i\multimap 0}[c_1^i] \triangleright \longmapsto E_1^{j\multimap 0}[c_2^j] \triangleright \longmapsto^{(n)} v_2^0$, where $E^{i\multimap 0}[c_1^i] = E_1^{j\multimap 0}[c_{11}^j]$ and $\textsc{Var}(E^{i\multimap 0}[c_1^i]) \vdash c_{11}^j \mathrel{\mathscr{R}^j} c_2^j$. By Lemma 383, $\langle i,\, E^{i\multimap 0},\, c_1^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} \langle j,\, E_1^{j\multimap 0},\, c_2^j\rangle_{\mathrm{f}}$. Given $E_1^{j\multimap 0}[c_2^j] \triangleright \longmapsto^{(n)} v_2^0$, by the induction hypothesis, $\langle j,\, E_1^{j\multimap 0},\, c_2^j\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} v_2^0$. Hence we have $\langle i,\, E^{i\multimap 0},\, c_1^i\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} v_2^0$.

□

Theorem 387 (Soundness of MEK Machine w.r.t. Reduction Semantics of Environmental MetaML)**.** *For any* $t_1^0 \in \textsc{Prgm}_{\mathrm{MetaML}}$, *if* $\triangleright \llparenthesis\, t_1^0,\, (\rho_{\mathrm{init}}^{\textsc{Var}(t_1^0)}; \varepsilon) \,\rrparenthesis \longmapsto^* v_2^0$, *then* $\langle 0,\, \Box^{0\multimap 0},\, \llparenthesis\, t_1^0,\, (\rho_{\mathrm{init}}^{\textsc{Var}(t_1^0)}; \varepsilon) \,\rrparenthesis\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} v_2^0$.

Proof. Suppose $\triangleright\, \Box^{0\multimap 0}[\llparenthesis\, t_1^0,\, (\rho_{\mathrm{init}}^{\textsc{Var}(t_1^0)}; \varepsilon) \,\rrparenthesis] \longmapsto^* v_2^0$, by Lemma 386, $\langle 0,\, \Box^{0\multimap 0},\, \llparenthesis\, t_1^0,\, (\rho_{\mathrm{init}}^{\textsc{Var}(t_1^0)}; \varepsilon) \,\rrparenthesis\rangle_{\mathrm{f}} \longmapsto^*_{\mathrm{mek}} v_2^0$. □

Any machine configuration in the MEK machine can be translated to its corresponding representation as a configuration at level 0 in Environmental MetaML.

Definition 388 (Translator)**.** Define the translator $\mathscr{T}_{\mathrm{mek}\to\mathrm{env}}$ to be a total function from the set of machine configurations $\textsc{Cfg}$ to the set of level 0 configurations $\textsc{Conf}^0$.

$$\mathscr{T}_{\mathrm{mek}\to\mathrm{env}} \;:\; \textsc{Cfg} \to \textsc{Conf}^0$$

$$\begin{aligned}
\mathscr{T}_{\mathrm{mek}\to\mathrm{env}}(\langle i,\, E^{i\multimap 0},\, c^i\rangle_{\mathrm{f}}) &= E^{i\multimap 0}[c^i] \\
\mathscr{T}_{\mathrm{mek}\to\mathrm{env}}(\langle i,\, E^{i\multimap 0},\, v^i\rangle_{\mathrm{b}}) &= E^{i\multimap 0}[v^i] \\
\mathscr{T}_{\mathrm{mek}\to\mathrm{env}}(\langle i,\, E^{i\multimap 0},\, c^i\rangle_{\mathrm{r}}) &= E^{i\multimap 0}[c^i] \\
\mathscr{T}_{\mathrm{mek}\to\mathrm{env}}(v^0) &= v^0
\end{aligned}$$

Lemma 389. *If* $C_1 \longmapsto_{\mathrm{mek}} C_2$, *then* $\triangleright\, \mathscr{T}_{\mathrm{mek}\to\mathrm{env}}(C_1) \longmapsto^{0*} \mathscr{T}_{\mathrm{mek}\to\mathrm{env}}(C_2)$.

Proof. We proceed by cases on $C_1 \longmapsto_{\mathrm{mek}} C_2$. □

Case 1. Reduce rules: Let $C_1 = \langle i,\, E^{i\multimap 0},\, c_1^i\rangle_{\mathrm{r}}$ and $C_2 = \langle i,\, E^{i\multimap 0},\, c_2^i\rangle_{\mathrm{f}}$. Then $\triangleright\, E^{i\multimap 0}[c_1] \longmapsto E^{i\multimap 0}[c_2]$ where $\textsc{Var}(C_1) \vdash c_1^i \mathrel{\mathscr{R}^i} c_2^i$. Hence $\triangleright\, \mathscr{T}_{\mathrm{mek}\to\mathrm{env}}(C_1) \longmapsto^0 \mathscr{T}_{\mathrm{mek}\to\mathrm{env}}(C_2)$.

Case 2. Focus rules: Let $C_1 = \langle i,\, E^{i\multimap 0},\, c_1^i\rangle_{\mathrm{f}}$ and $C_2 = \langle i,\, E^{i\multimap 0},\, c_2^i\rangle_{?}$. Then $E^{i\multimap 0}[c_1] = E^{i\multimap 0}[c_2]$. Hence $\triangleright\, \mathscr{T}_{\mathrm{mek}\to\mathrm{env}}(C_1) \longmapsto^{0*} \mathscr{T}_{\mathrm{mek}\to\mathrm{env}}(C_2)$.

Case 3. Build rules:

Case i. (b-value-0). Let $C_1 = \langle 0,\, \Box,\, v^0\rangle_{\mathrm{b}}$ and $C_2 = v^0$. Then $\Box[v^0] = v^0$. Hence $\triangleright\, \mathscr{T}_{\mathrm{mek}\to\mathrm{env}}(C_1) \longmapsto^{0*} \mathscr{T}_{\mathrm{mek}\to\mathrm{env}}(C_2)$.

Case ii. (other rules). Let $C_1 = \langle i,\, E^{i\multimap 0},\, c_1^i\rangle_{\mathrm{b}}$ and $C_2 = \langle i,\, E^{i\multimap 0},\, c_2^i\rangle_{?}$. Then $E^{i\multimap 0}[c_1] = E^{i\multimap 0}[c_2]$. Hence $\triangleright\, \mathscr{T}_{\mathrm{mek}\to\mathrm{env}}(C_1) \longmapsto^{0*} \mathscr{T}_{\mathrm{mek}\to\mathrm{env}}(C_2)$.

Lemma 390. *If* $C_1 \longmapsto^*_{\text{mek}} C_2$, *then* $\triangleright\, \mathscr{T}_{\text{mek}\to\text{env}}(C_1) \longmapsto^{0*} \mathscr{T}_{\text{mek}\to\text{env}}(C_2)$.

Proof. Suppose $C_1 \longmapsto^{(n)}_{\text{mek}} C_2$. We proceed by induction on n.

Case 1. When $n = 0$, $C_1 = C_2$. Then $\mathscr{T}_{\text{mek}\to\text{env}}(C_1) = \mathscr{T}_{\text{mek}\to\text{env}}(C_2)$. We have $\triangleright\, \mathscr{T}_{\text{mek}\to\text{env}}(C_1) \longmapsto^{0*} \mathscr{T}_{\text{mek}\to\text{env}}(C_2)$ immediately.

Case 2. Let $C_1 \longmapsto_{\text{mek}} C_3 \longmapsto^{(n)}_{\text{mek}} C_2$.
Given $C_1 \longmapsto_{\text{mek}} C_3$, by Lemma 389, $\triangleright\, \mathscr{T}_{\text{mek}\to\text{env}}(C_1) \longmapsto^{0*} \mathscr{T}_{\text{mek}\to\text{env}}(C_2)$.
Given $C_3 \longmapsto^{(n)}_{\text{mek}} C_2$, by the induction hypothesis, $\triangleright\, \mathscr{T}_{\text{mek}\to\text{env}}(C_3) \longmapsto^{0*} \mathscr{T}_{\text{mek}\to\text{env}}(C_2)$.
Hence $\triangleright\, \mathscr{T}_{\text{mek}\to\text{env}}(C_1) \longmapsto^{0*} \mathscr{T}_{\text{mek}\to\text{env}}(C_2)$.

□

Theorem 391 (Completeness of MEK Machine w.r.t. Reduction Semantics of Environmental MetaML)**.** *For any* $t_1^0 \in \text{PRGM}_{\text{MetaML}}$, *if* $\langle 0, \square, \blacklozenge\, t_1^0,\, (\rho_{\text{init}}^{\text{VAR}(t_1^0)}; \varepsilon)\, \blacklozenge \rangle_{\text{f}} \longmapsto^*_{\text{mek}} v_2^0$, *then* $\triangleright \blacklozenge\, t_1^0,\, (\rho_{\text{init}}^{\text{VAR}(t_1^0)}; \varepsilon)\, \blacklozenge \longmapsto^* v_2^0$.

Proof. If $\langle 0, \square, \blacklozenge\, t_1^0,\, (\rho_{\text{init}}^{\text{VAR}(t_1^0)}; \varepsilon)\, \blacklozenge \rangle_{\text{f}} \longmapsto^*_{\text{mek}} v_2^0$, by Lemma 390, $\triangleright\, \mathscr{T}_{\text{mek}\to\text{env}}(\langle 0, \square, \blacklozenge\, t_1^0,\, (\rho_{\text{init}}^{\text{VAR}(t_1^0)}; \varepsilon)\, \blacklozenge \rangle_{\text{f}}) \longmapsto^* \mathscr{T}_{\text{mek}\to\text{env}}(v_2^0)$. We have $\triangleright \blacklozenge\, t_1^0,\, (\rho_{\text{init}}^{\text{VAR}(t_1^0)}; \varepsilon)\, \blacklozenge \longmapsto^{0*} v_2^0$. □

Theorem 392 (Kleene Equality of Evaluators)**.** *For any* $t \in \text{PRGM}_{\text{MetaML}}$, $eval_{\text{MetaML:EnvRed}}(t)$ *is Kleene equal to* $eval_{\text{MetaML:MEK}}(t)$.

Proof. We first show if $eval_{\text{MetaML:EnvRed}}(t) = a$ where $a \in \text{ANS}_{\text{MetaML}}$, then $eval_{\text{MetaML:MEK}}(t) = a$.

Case 1. If $eval_{\text{MetaML:EnvRed}}(t) = \texttt{function}$, then $\triangleright \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon)\, \blacklozenge \longmapsto^* \langle\!|\, \lambda x.t'^0,\, \rho^*\, |\!\rangle$.
By Theorem 387, $\langle 0, \square, \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon)\, \blacklozenge \rangle_{\text{f}} \longmapsto^*_{\text{mek}} \langle\!|\, \lambda x.t'^0,\, \rho^*\, |\!\rangle$. We have $eval_{\text{MetaML:SubAbs}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:EnvRed}}(t) = \texttt{code}$, then $\triangleright \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon)\, \blacklozenge \longmapsto^* \langle v^1 \rangle$.
By Theorem 387, $\langle 0, \square, \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon)\, \blacklozenge \rangle_{\text{f}} \longmapsto^*_{\text{mek}} \langle v^1 \rangle$. We have $eval_{\text{MetaML:MEK}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:EnvRed}}(t) = n$, then $\triangleright \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon)\, \blacklozenge \longmapsto^* n$.
By Theorem 387, $\langle 0, \square, \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon)\, \blacklozenge \rangle_{\text{f}} \longmapsto^*_{\text{mek}} n$. We have $eval_{\text{MetaML:MEK}}(t) = n$.

We then show if $eval_{\text{MetaML:MEK}}(t) = a$ where $a \in \text{ANS}_{\text{MetaML}}$, then $eval_{\text{MetaML:EnvRed}}(t) = a$.

Case 1. If $eval_{\text{MetaML:MEK}}(t) = \texttt{function}$, then $\langle 0, \square, \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon)\, \blacklozenge \rangle_{\text{f}} \longmapsto^*_{\text{mek}} \langle\!|\, \lambda x.t'^0,\, \rho^*\, |\!\rangle$.
By Theorem 387, $\triangleright \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon)\, \blacklozenge \longmapsto^* \langle\!|\, \lambda x.t'^0,\, \rho^*\, |\!\rangle$. We have $eval_{\text{MetaML:EnvRed}}(t) = \texttt{function}$.

Case 2. If $eval_{\text{MetaML:MEK}}(t) = \texttt{code}$, then $\langle 0, \square, \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon)\, \blacklozenge \rangle_{\text{f}} \longmapsto^*_{\text{mek}} \langle v^1 \rangle$.
By Theorem 391, $\triangleright \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon)\, \blacklozenge \longmapsto^* \langle v^1 \rangle$. We have $eval_{\text{MetaML:EnvRed}}(t) = \texttt{code}$.

Case 3. If $eval_{\text{MetaML:MEK}}(t) = n$, then $\langle 0, \square, \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon)\, \blacklozenge \rangle_{\text{f}} \longmapsto^*_{\text{mek}} n$.
By Theorem 391, $\triangleright \blacklozenge\, t,\, (\rho_{\text{init}}^{\text{VAR}(t)}; \varepsilon)\, \blacklozenge \longmapsto^* n$. We have $eval_{\text{MetaML:EnvRed}}(t) = n$.

We observe that $eval_{\text{MetaML:EnvRed}}(t)$ is undefined if and only if $eval_{\text{MetaML:MEK}}(t)$ is undefined. Therefore, $eval_{\text{MetaML:EnvRed}}(t)$ is Kleene equal to $eval_{\text{MetaML:MEK}}(t)$. □