Strictness and Totality Analysis with Conjunction

Kirsten Lackner Solberg

Computer Science Department
Aarhus University, Denmark
e-mail: kls@daimi.aau.dk

Abstract. We extend the strictness and totality analysis of [12] by allowing conjunction at all levels rather than at the top-level. We prove the strictness and totality analysis correct with respect to a denotational semantics and finally construct an algorithm for inferring the strictness and totality properties.

1 Introduction

Strictness analysis has proved useful in the implementation of lazy functional languages like Miranda, Lazy ML and Haskell: when a function is strict it is safe to evaluate its argument before performing the function call. Totality analysis has not been adopted so widely: if the argument to a function is known to terminate then it is safe to evaluate it before performing the function call [9].

In the literature there are several approaches to the specification of strictness analysis: abstract interpretation (e.g. [10, 3]), projection analysis (e.g. [14]) and inference based methods (e.g. [2, 6, 7, 8, 15]). Totality analysis has received much less attention and has primarily been specified using abstract interpretation [10, 1]. It can be regarded as an approximation to time complexity analysis; most literature performing such developments consider eager languages but [11] considers lazy languages.

The paper [12] presents an inference system for performing strictness *and* totality analysis. The inference system is an extension of the usual type system in that we introduce three annotations on types σ:

- $\mathbf{b}^{\sigma}$: the value has type σ and is definitely $\perp$,
- $\mathbf{n}^{\sigma}$: the value has type σ and is definitely *not* $\perp$, and
- $\top^{\sigma}$: the value has type σ and it can be any value.

Annotated types can be constructed using the function type constructor and (top-level) conjunction. As an example a function may have the annotated type $(\mathbf{n}^{\mathtt{Int}} \rightarrow \mathbf{n}^{\mathtt{Int}}) \wedge (\mathbf{b}^{\mathtt{Int}} \rightarrow \mathbf{b}^{\mathtt{Int}})$ which means that given a terminating argument the function will definitely terminate and given a non-terminating argument it will definitely not terminate. Thus we capture the strictness as well as the totality of the function. Strictness and totality information can also be combined as in $(\mathbf{n}^{\mathtt{Int}} \rightarrow \mathbf{n}^{\mathtt{Int}} \rightarrow \mathbf{n}^{\mathtt{Int}}) \wedge (\mathbf{b}^{\mathtt{Int}} \rightarrow \mathbf{n}^{\mathtt{Int}} \rightarrow \mathbf{n}^{\mathtt{Int}}) \wedge (\mathbf{n}^{\mathtt{Int}} \rightarrow \mathbf{b}^{\mathtt{Int}} \rightarrow \mathbf{n}^{\mathtt{Int}}) \wedge (\mathbf{b}^{\mathtt{Int}} \rightarrow \mathbf{b}^{\mathtt{Int}} \rightarrow \mathbf{b}^{\mathtt{Int}})$ which will be the annotated type of McCarthy's ambiguity operator.

The strictness and totality analysis in [12] is defined by an inference system and proven sound with respect to a natural-style operational semantics. The proof is long and detailed because in order to reason about fixpoints it is necessary to introduce new terms which include information about how many times the fixpoint was allowed to be unfolded. In a denotational semantics it is easy to reason about fixpoints since there is no need for new terms; all the denotations are already included. In this paper we will extend the analysis of [12] with unrestricted conjunction and we will prove this analysis sound with respect to a denotational semantics. Finally we will construct an algorithm for inferring the strictness and totality properties. There are two different ways of constructing inference algorithms: one is to calculate all the information that can be inferred; the other is to check if a given property can be inferred. We choose the latter since it is often the case that we are only interested in knowing if a term possesses one particular property and we thus expect this approach to lead to more efficient implementations.

Overview Section 2 presents the standard type inference rules and the denotational semantics for our simply-typed lazy lambda calculus. In section 3 we define the strictness and totality properties and the analysis of [12] is restated. In section 4 the analysis is proven correct with respect to the denotational semantics. Finally in Section 5 we construct an algorithm for strictness and totality property inference.

2 Syntax and Semantics

This section introduces the simply-typed lazy λ-calculus with constants and fixpoints. We first define the syntax and typing rules for the language, and then we define the lazy semantics by means of a denotational semantics.

2.1 The Language

The types, $\tau \in \mathrm{T}$, are either base types or function types

$$\tau ::= \mathtt{B} \mid \tau \rightarrow \tau$$

where the base types (i.e. the `B`'s) include `Bool` and `Int`. The terms, $\mathtt{e} \in \mathrm{E}$, of the simply-typed λ-calculus are

$$\mathtt{e} ::= \mathtt{x}^{\tau} \mid \lambda \mathtt{x}^{\tau}.\mathtt{e} \mid \mathtt{e}\ \mathtt{e} \mid \mathtt{fix}\ \mathtt{e} \mid \mathtt{if}\ \mathtt{e}\ \mathtt{then}\ \mathtt{e}\ \mathtt{else}\ \mathtt{e} \mid \mathtt{c}$$

where the constants (i.e. the `c`'s) include `true` and `false` of type `Bool`, and all the integers of type `Int`. In this paper we have chosen to annotate the variables with their type.

We are only considering terms that are typeable according to the type inference rules defined in Figure 1. To all the constants `c` there is a unique type $\tau_{\mathtt{c}}$. The free variables in the term `e` is the set FV(`e`) and substitution on terms $\mathtt{e}[\mathtt{e}_2/\mathtt{x}]$ is defined as usual.

$$[\mathrm{var}]_T\ \frac{}{\vdash \mathtt{x}^{\tau} : \tau} \qquad [\mathrm{const}]_T\ \frac{}{\vdash \mathtt{c} : \tau_{\mathtt{C}}}$$

$$[\mathrm{abs}]_T\ \frac{\vdash \mathtt{e} : \tau}{\vdash \lambda \mathtt{x}^{\sigma}.\mathtt{e} : \sigma \to \tau} \qquad [\mathrm{app}]_T\ \frac{\vdash \mathtt{e}_1 : \sigma \to \tau \quad \vdash \mathtt{e}_2 : \sigma}{\vdash \mathtt{e}_1\ \mathtt{e}_2 : \tau}$$

$$[\mathrm{if}]_T\ \frac{\vdash \mathtt{e}_1 : \mathtt{Bool} \quad \vdash \mathtt{e}_2 : \tau \quad \vdash \mathtt{e}_3 : \tau}{\vdash \mathtt{if}\ \mathtt{e}_1\ \mathtt{then}\ \mathtt{e}_2\ \mathtt{else}\ \mathtt{e}_3 : \tau} \qquad [\mathrm{fix}]_T\ \frac{\vdash \mathtt{e} : \sigma \to \sigma}{\vdash \mathtt{fix}\ \mathtt{e} : \sigma}$$

Fig. 1. Type inference

2.2 The Semantics

The reason for choosing a lazy semantics is to capture the semantics of "real-life" lazy functional programming languages like Miranda [13] in contrast to most other papers on strictness analysis ([3, 5]). The semantics in [12] is a natural-style operational semantics where the terms are evaluated to weak head normal form (abbreviated WHNF), i.e. to constants or lambda-abstractions. Here we will define the semantics as a denotational semantics. We have a type-indexed family of domains:

$$\mathrm{D}_{\mathrm{B}} = \mathrm{B}_{\perp}$$
$$\mathrm{D}_{\sigma \to \tau} = (\mathrm{D}_{\sigma} \to_{\mathrm{cont}} \mathrm{D}_{\tau})_{\perp}$$

We need the two functions up and dn to get from a domain to the lifted domain and back again. We also need an environment ρ that assigns denotations to variables. This is a partial function from variables to the disjoint union of the domains. The environment is type-preserving, that is if $\rho\,(\mathtt{x}^{\tau})$ is defined then it is a member of D_{τ}. Now the semantics assigns denotations to terms, meaning that if we have $\vdash \mathtt{e} : \tau$, then $[\![\mathtt{e}]\!]$ is a partial function from environments to D_{τ} (Figure 2). For each constant, $\mathtt{c}$, there is a unique predefined denotation c.

3 The Analysis

We will assume that the terms are annotated with their types. We define the strictness and totality logic $\mathcal{L}_{\tau}^{\mathrm{ST}}$ as follows:

$$\mathbf{n}^{\tau}, \mathbf{b}^{\tau}, \top^{\tau} \in \mathcal{L}_{\tau}^{\mathrm{ST}} \qquad \frac{\phi^{\tau} \in \mathcal{L}_{\tau}^{\mathrm{ST}} \quad \psi^{\tau} \in \mathcal{L}_{\tau}^{\mathrm{ST}}}{\phi^{\tau} \wedge \psi^{\tau} \in \mathcal{L}_{\tau}^{\mathrm{ST}}}$$

$$\frac{\psi^{\tau} \in \mathcal{L}_{\tau}^{\mathrm{ST}}}{\downarrow(\psi^{\tau}) \in \mathcal{L}_{\tau}^{\mathrm{ST}}}\,. \qquad \frac{\phi^{\tau} \in \mathcal{L}_{\tau}^{\mathrm{ST}} \quad \psi^{\sigma} \in \mathcal{L}_{\sigma}^{\mathrm{ST}}}{\phi^{\tau} \to \psi^{\sigma} \in \mathcal{L}_{\tau \to \sigma}^{\mathrm{ST}}}$$

The idea is that $\mathcal{L}_{\tau}^{\mathrm{ST}}$ is the properties that a term of type τ may possess. We relate a subset of D_{τ} to each property ϕ^{τ} (Figure 3). For this we need to take the downwards closure of a subset of the domains:

$$\mathsf{dc}(X) = \{d' \mid \exists d \in X : d' \leq d\}$$

$$
\begin{aligned}
[\![\mathtt{x}^{\tau}]\!]\,\rho &= \rho\,(\mathtt{x}^{\tau})\\
[\![\lambda\mathtt{x}^{\tau}.\mathtt{e}]\!]\,\rho &= \mathtt{up}(\lambda d.[\![\mathtt{e}]\!]\ \rho[d/\mathtt{x}^{\tau}])\\
[\![\mathtt{e}_1\ \mathtt{e}_2]\!]\,\rho &= \mathtt{dn}([\![\mathtt{e}_1]\!]\,\rho)([\![\mathtt{e}_2]\!]\,\rho)\\
[\![\mathtt{fix}\ \ \mathtt{e}]\!]\,\rho &= \sqcup_n d_n \text{where}\\
& \qquad d_0 = \bot\\
& \qquad d_{n+1} = \mathtt{dn}([\![\mathtt{e}]\!]\,\rho)d_n\\
[\![\mathtt{if}\ \mathtt{e}_1\ \mathtt{then}\ \mathtt{e}_2\ \mathtt{else}\ \mathtt{e}_3]\!]\,\rho &= \begin{cases} \bot, & \text{if } [\![\mathtt{e}_1]\!]\,\rho = \bot_{\mathrm{D}_{\mathtt{Bool}}}\\ [\![\mathtt{e}_2]\!]\,\rho, & \text{if } [\![\mathtt{e}_1]\!]\,\rho = \mathtt{true}\\ [\![\mathtt{e}_3]\!]\,\rho, & \text{if } [\![\mathtt{e}_1]\!]\,\rho = \mathtt{false}\end{cases}\\
[\![\mathtt{c}]\!]\,\rho &= c
\end{aligned}
$$

Fig. 2. Denotational semantics for the λ-calculus

$$
\begin{aligned}
[\![\mathbf{n}^{\tau}]\!] &= \mathrm{D}_{\tau} \setminus \{\bot_{\mathrm{D}_{\tau}}\}\\
[\![\mathbf{b}^{\tau}]\!] &= \{\bot_{\mathrm{D}_{\tau}}\}\\
[\![\top^{\tau}]\!] &= \mathrm{D}_{\tau}\\
[\![\phi^{\tau} \to \psi^{\sigma}]\!] &= \{f \in \mathrm{D}_{\tau \to \sigma} \mid \mathtt{dn}(f)\,[\![\phi^{\tau}]\!] \subseteq [\![\psi^{\sigma}]\!]\}\\
[\![\psi^{\sigma} \wedge \phi^{\sigma}]\!] &= [\![\psi^{\sigma}]\!] \cap [\![\phi^{\sigma}]\!]\\
[\![\downarrow\!(\psi^{\sigma})]\!] &= \mathtt{dc}([\![\psi^{\sigma}]\!])
\end{aligned}
$$

Fig. 3. The meaning of the properties

The idea is that a term with the strictness and totality property $\mathbf{n}^{\tau}$ is a term of type τ and it has a denotation in $\mathrm{D}_{\tau}\setminus\{\bot_{\mathrm{D}_{\tau}}\}$ i.e. the term has a non-bottom denotation. A term with the strictness and totality property $\mathbf{b}^{\tau}$ has the type τ and it has the denotation $\bot_{\mathrm{D}_{\tau}}$, i.e. it is bottom. For terms with the strictness and totality property $\top^{\tau}$ we only know that it has the type τ. A term with strictness and totality property $\phi^{\tau} \to \psi^{\sigma}$ will have the type $\tau \to \sigma$ and will, when applied to another term with the strictness and totality property ϕ^{τ}, yield a term with strictness and totality property ψ^{σ}. A term with the strictness and totality property $\phi^{\tau} \wedge \psi^{\tau}$ is a term with both properties, ϕ^{τ} and ψ^{τ}. The set of terms with property $\downarrow\!(\phi^{\tau})$ is the set of term with property ϕ^{τ} and including bottom; but whenever ϕ^{τ} is empty so is $\downarrow\!(\phi^{\tau})$. The $\downarrow\!(\phi^{\tau})$-property is used to express the fact that the functions we are considering are monotonic, see the rule [monotone] in Figure 4.

An empty property is a property possessed by no term. One example is $(\mathbf{n}^{\tau} \wedge \mathbf{b}^{\tau})$ since a term cannot both terminate and not terminate.

Most terms possess more than one strictness and totality property; as an example consider the strictness and totality properties of $\lambda \mathtt{x}^{\mathtt{Int}} . 7$ which includes

$$\top^{\mathtt{Int}} \to \mathtt{Int}, \mathbf{n}^{\mathtt{Int}} \to \mathtt{Int}, \top^{\mathtt{Int}} \to \mathbf{n}^{\mathtt{Int}}, \mathbf{n}^{\mathtt{Int}} \to \mathbf{n}^{\mathtt{Int}}$$

among others. Some of them are redundant and to express this we define coercions between the strictness and totality properties: $\psi^{\tau} \leq_{\mathrm{ST}} \phi^{\tau}$ may only hold if all terms of strictness and totality property ψ^{τ} also have the strictness and totality property ϕ^{τ} (assuming the types are the same).

The coercion relation $\leq_{\mathrm{ST}}$ defined in Figure 4 is reflexive, transitive, and anti-monotone in contravariant position. We will write $\equiv$ for the equivalence induced by $\leq_{\mathrm{ST}}$, i.e. $\psi^{\tau} \equiv \phi^{\tau}$ if and only if $\psi^{\tau} \leq_{\mathrm{ST}} \phi^{\tau}$ and $\phi^{\tau} \leq_{\mathrm{ST}} \psi^{\tau}$. The rule [top1] expresses that the strictness and totality property $\top^{\tau}$ includes all the terms of type τ. One axiom derived from the rule [top1] is

$$\top^{\tau} \to \top^{\sigma} \leq_{\mathrm{ST}} \top^{\tau \to \sigma} \tag{1}$$

Axiom (1) then motivates rule [top2] because when combined they yield

$$\top^{\tau \to \sigma} \equiv \top^{\tau} \to \top^{\sigma}$$

The rule [monotone] ensures that we live in a universe of monotone function: if we know less about the argument to a function, then we should know less about the result as well. We can infer

$$\mathbf{n}^{\tau} \to \mathbf{b}^{\tau} \leq_{\mathrm{ST}} \top^{\tau} \to \mathbf{b}^{\tau}$$

using the [monotone]-rule. This is used in order to infer the type

$$(\mathbf{n}^{\tau} \to \mathbf{b}^{\tau}) \to \top^{\tau} \to \mathbf{b}^{\tau}$$

for `twice`.

We would like the $\downarrow(\psi)$ properties to be equivalent to properties without any $\downarrow(\psi')$. We have

$$\begin{array}{ll} \downarrow(\mathbf{n}^{\sigma}) \equiv \top^{\sigma} & \downarrow(\top^{\sigma}) \equiv \top^{\sigma} \\ \downarrow(\mathbf{b}^{\sigma}) \equiv \mathbf{b}^{\sigma} & \downarrow(\psi \to \phi) \equiv \psi \to \downarrow(\phi) \end{array}$$

but only

$$\downarrow(\psi \wedge \phi) \leq_{\mathrm{ST}} \downarrow(\psi) \wedge \downarrow(\phi)$$

The reason for introducing $\downarrow(\psi)$ in the first place was to be able to express the monotonicity-rule. The advantage of having an equivalence between properties with $\downarrow$ and properties without $\downarrow$ is that we can forget all about $\downarrow(\phi)$-properties since they are equivalent to another property without any $\downarrow(\phi)$-properties and the world will look nicer.

Now we can turn to the analysis defined in Figure 5. The list A of assumptions gives strictness and totality properties to the free variables. We shall assume that all the variables in the list are distinct. For each constant `c`, we assume that a strictness and

$$[\text{ref}]\ \frac{}{\psi \leq_{\mathrm{ST}} \psi} \qquad [\text{trans}]\ \frac{\psi \leq_{\mathrm{ST}} \phi \quad \phi \leq_{\mathrm{ST}} \chi}{\psi \leq_{\mathrm{ST}} \chi}$$

$$[\rightarrow]\ \frac{\psi' \leq_{\mathrm{ST}} \psi \quad \phi \leq_{\mathrm{ST}} \phi'}{\psi \rightarrow \phi \leq_{\mathrm{ST}} \psi' \rightarrow \phi'}$$

$$[\text{top1}]\ \frac{}{\phi^{\tau} \leq_{\mathrm{ST}} \top^{\tau}} \qquad [\text{top2}]\ \frac{}{\top^{\tau \rightarrow \sigma} \leq_{\mathrm{ST}} \top^{\tau} \rightarrow \top^{\sigma}}$$

$$[\text{bot}]\ \frac{}{\mathbf{b}^{\tau \rightarrow \sigma} \leq_{\mathrm{ST}} \top^{\tau} \rightarrow \mathbf{b}^{\sigma}}$$

$$[\text{notbot}]\ \frac{}{\mathbf{n}^{\tau} \rightarrow \mathbf{n}^{\sigma} \leq_{\mathrm{ST}} \mathbf{n}^{\tau \rightarrow \sigma}}$$

$$[\downarrow 1]\ \frac{}{\phi \leq_{\mathrm{ST}} \downarrow(\phi)} \qquad [\downarrow 2]\ \frac{\psi \leq_{\mathrm{ST}} \phi}{\downarrow(\psi) \leq_{\mathrm{ST}} \downarrow(\phi)}$$

$$[\downarrow 3]\ \frac{}{\top^{\tau} \leq_{\mathrm{ST}} \downarrow(\mathbf{n}^{\tau})} \qquad [\downarrow 4]\ \frac{}{\downarrow(\mathbf{b}^{\tau}) \leq_{\mathrm{ST}} \mathbf{b}^{\tau}}$$

$$[\downarrow 5]\ \frac{}{\downarrow(\downarrow(\psi)) \leq_{\mathrm{ST}} \downarrow(\psi)} \qquad [\downarrow 6]\ \frac{}{\downarrow(\psi \wedge \phi) \leq_{\mathrm{ST}} \downarrow(\psi) \wedge \downarrow(\phi)}$$

$$[\downarrow 7]\ \frac{}{\downarrow(\psi \rightarrow \phi) \equiv \psi \rightarrow \downarrow(\phi)}$$

$$[\text{monotone}]\ \frac{}{\psi \rightarrow \phi \leq_{\mathrm{ST}} \downarrow(\psi) \rightarrow \downarrow(\phi)}$$

$$[\wedge 1]\ \frac{}{\psi \wedge \phi \leq_{\mathrm{ST}} \psi} \qquad [\wedge 2]\ \frac{}{\psi \wedge \phi \leq_{\mathrm{ST}} \phi}$$

$$[\wedge]\ \frac{\chi \leq_{\mathrm{ST}} \psi \quad \chi \leq_{\mathrm{ST}} \phi}{\chi \leq_{\mathrm{ST}} \psi \wedge \phi}$$

$$[\rightarrow \wedge]\ \frac{}{(\psi \rightarrow \phi') \wedge (\psi \rightarrow \phi'') \leq_{\mathrm{ST}} \psi \rightarrow (\phi' \wedge \phi'')}$$

Fig. 4. Coercions between strictness and totality properties

totality property is specified ϕ_{c}; as an example for the successor function we have $\phi_{\mathtt{succ}} = (\mathbf{n}^{\mathtt{Int}} \rightarrow \mathbf{n}^{\mathtt{Int}}) \wedge (\mathbf{b}^{\mathtt{Int}} \rightarrow \mathbf{b}^{\mathtt{Int}})$. The remaining properties can be coerced from this strictness and totality property:

$$\begin{array}{l} \phi_{\mathtt{succ}} \leq_{\mathrm{ST}} \top^{\mathtt{Int} \rightarrow \mathtt{Int}} \\ \phi_{\mathtt{succ}} \leq_{\mathrm{ST}} \top^{\mathtt{Int}} \rightarrow \top^{\mathtt{Int}} \\ \phi_{\mathtt{succ}} \leq_{\mathrm{ST}} \mathbf{n}^{\mathtt{Int}} \rightarrow \top^{\mathtt{Int}} \\ \quad \text{etc.} \end{array}$$

The rules [var], [abs], [app], and [const] are straightforward. There are three rules for conditional — depending on whether the test is of strictness and totality property $\mathbf{b}^{\mathtt{Bool}}$, $\mathbf{n}^{\mathtt{Bool}}$, or $\top^{\mathtt{Bool}}$

$$[\text{var}]\ \frac{}{\mathrm{A} \vdash \mathtt{x}^{\tau} : \phi^{\tau}}\ \text{if } \mathtt{x}^{\tau} : \phi^{\tau} \in \mathrm{A}$$

$$[\text{abs}]\ \frac{\mathrm{A}, \mathtt{x} : \phi^{\sigma} \vdash \mathtt{e} : \psi^{\tau}}{\mathrm{A} \vdash \lambda \mathtt{x}^{\sigma}.\mathtt{e} : (\phi^{\sigma} \rightarrow \psi^{\tau}) \wedge \mathbf{n}^{\sigma \rightarrow \tau}}$$

$$[\text{app}]\ \frac{\mathrm{A} \vdash \mathtt{e}_1 : \psi^{\sigma} \rightarrow \phi^{\tau} \quad \mathrm{A} \vdash \mathtt{e}_2 : \psi^{\sigma}}{\mathrm{A} \vdash \mathtt{e}_1\ \mathtt{e}_2 : \phi^{\tau}}$$

$$[\text{if1}]\ \frac{\mathrm{A} \vdash \mathtt{e}_1 : \mathbf{b}^{\mathtt{Bool}} \quad \mathrm{A} \vdash \mathtt{e}_2 : \psi^{\sigma} \quad \mathrm{A} \vdash \mathtt{e}_3 : \psi^{\sigma}}{\mathrm{A} \vdash \mathtt{if}\ \mathtt{e}_1\ \mathtt{then}\ \mathtt{e}_2\ \mathtt{else}\ \mathtt{e}_3 : \mathbf{b}^{\sigma}}$$

$$[\text{if2}]\ \frac{\mathrm{A} \vdash \mathtt{e}_1 : \mathbf{n}^{\mathtt{Bool}} \quad \mathrm{A} \vdash \mathtt{e}_2 : \psi^{\sigma} \quad \mathrm{A} \vdash \mathtt{e}_3 : \psi^{\sigma}}{\mathrm{A} \vdash \mathtt{if}\ \mathtt{e}_1\ \mathtt{then}\ \mathtt{e}_2\ \mathtt{else}\ \mathtt{e}_3 : \psi^{\sigma}}$$

$$[\text{if3}]\ \frac{\mathrm{A} \vdash \mathtt{e}_1 : \top^{\mathtt{Bool}} \quad \mathrm{A} \vdash \mathtt{e}_2 : \psi^{\sigma} \quad \mathrm{A} \vdash \mathtt{e}_3 : \psi^{\sigma}}{\mathrm{A} \vdash \mathtt{if}\ \mathtt{e}_1\ \mathtt{then}\ \mathtt{e}_2\ \mathtt{else}\ \mathtt{e}_3 : \psi^{\sigma}}\ \text{if BOT}(\psi^{\sigma})$$

$$[\text{fix}]\ \frac{\mathrm{A} \vdash \mathtt{e} : \psi_1^{\sigma} \rightarrow \psi_2^{\sigma} \wedge \psi_2^{\sigma} \rightarrow \psi_3^{\sigma} \wedge \cdots \wedge \psi_{n-1}^{\sigma} \rightarrow \psi_n^{\sigma}}{\mathrm{A} \vdash \mathtt{fix}\ \mathtt{e} : \psi_n^{\sigma}}$$
$$\text{if} \begin{cases} \text{BOT}(\psi_1^{\sigma}), \\ \exists p, q : p < q \wedge \\ \psi_q^{\sigma} \leq_{\text{ST}} \psi_p^{\sigma} \end{cases}$$

$$[\text{const}]\ \frac{}{\mathrm{A} \vdash \mathtt{c} : \psi_{\mathtt{C}}}$$

$$[\text{coer}]\ \frac{\mathrm{A} \vdash \mathtt{e} : \psi^{\sigma}}{\mathrm{A} \vdash \mathtt{e} : \phi^{\sigma}}\ \text{if } \psi^{\sigma} \leq_{\text{ST}} \phi^{\sigma} \qquad [\text{conj}]\ \frac{\mathrm{A} \vdash \mathtt{e} : \psi^{\sigma} \quad \mathrm{A} \vdash \mathtt{e} : \phi^{\sigma}}{\mathrm{A} \vdash \mathtt{e} : \psi^{\sigma} \wedge \phi^{\sigma}}$$

Fig. 5. Strictness and Totality Property Inference

The predicate BOT defined by

$$\begin{array}{ll} \text{BOT}(\mathbf{n}^{\sigma}) = \mathtt{ff} & \text{BOT}(\top^{\sigma}) = \mathtt{tt} \\ \text{BOT}(\mathbf{b}^{\sigma}) = \mathtt{tt} & \text{BOT}(\downarrow(\phi)) = \text{BOT}(\phi) \\ \text{BOT}(\psi \rightarrow \phi) = \text{BOT}(\phi) & \\ \text{BOT}(\psi \wedge \phi) = \text{BOT}(\psi) \wedge \text{BOT}(\phi) & \end{array}$$

is true for the strictness and totality property ψ^{τ} whenever it is guaranteed that bottom can be described by the property ψ^{τ}. The reason for *not* taking BOT($\downarrow(\phi)$) be to true is that ϕ may be empty and therefore also $\downarrow(\phi)$ may be empty. However, the definition of BOT($\downarrow(\phi)$) that we have adopted is not as precise as one would wish. (An example of which is when $\phi = \mathbf{n}^{\sigma}$.)

The rule [coer] allows to change the strictness and totality property to a greater property. It is quite useful as a preparation for applying the rule [if3], i.e. whenever BOT(ϕ) is not true we can coerce ϕ to a property ϕ' so that BOT(ϕ') is true.

From the rule [fix] the two more intuitive rules

$$[\text{fix1}]\ \frac{\mathrm{A} \vdash \mathtt{e} : \psi^{\sigma} \rightarrow \psi^{\sigma}}{\mathrm{A} \vdash \mathtt{fix}\ \mathtt{e} : \psi^{\sigma}}\ \text{if BOT}(\psi^{\sigma})$$

and

$$\text{[fix2]}\ \frac{\mathrm{A} \vdash \mathtt{e} : \psi^\sigma \rightarrow \phi^\sigma}{\mathrm{A} \vdash \mathtt{fix\ e} : \phi^\sigma}\ \text{if BOT}(\psi^\sigma) \text{ and } \phi^\sigma \leq_{\mathrm{ST}} \psi^\sigma$$

are easily derived. The rule [fix] itself can be explained as follows: first we have to ensure that the strictness and totality property ψ_1^σ can describe bottom in order to be able to start the iteration towards the fixpoint. After the first iteration the term has the property ψ_2^σ and after the second the strictness and totality property ψ_3^σ, etc. When we reach iteration number $q-1$ we can apply the rule [coer] because we have $\psi_q^\sigma \leq_{\mathrm{ST}} \psi_p^\sigma$ and the term has the property ψ_p^σ. In this way we can go on as long as necessary to calculate the fixpoint. Finally we are allowed to iterate $n-q$ times more to get the property ψ_n^σ for the fixpoint.

Example 1
We can infer $\vdash \mathtt{fix}\ \lambda \mathtt{x}_{\mathtt{Int}}.\mathtt{x}_{\mathtt{Int}} : \mathbf{b}^{\mathtt{Int}}$ which is more precise than the information $\top^{\mathtt{Int}}$ obtained by [15]; in [16] it can be done. In the systems of [2, 6, 7, 16] one can infer the property $\top^{\mathtt{Int}}$ for the term $\mathtt{fix}\ \lambda \mathtt{x}^{\mathtt{Int}}.\mathtt{7}$ whereas we can infer the more precise property $\mathbf{n}^{\mathtt{Int}}$. In this system (as well as those of [12, 2, 6, 7, 16]) it is possible to infer the property

$$(\mathbf{b}^{\mathtt{Int}} \rightarrow \top^{\mathtt{Int}} \rightarrow \top^{\mathtt{Int}} \rightarrow \mathbf{b}^{\mathtt{Int}}) \wedge (\top^{\mathtt{Int}} \rightarrow \mathbf{b}^{\mathtt{Int}} \rightarrow \top^{\mathtt{Int}} \rightarrow \mathbf{b}^{\mathtt{Int}})$$

for the term

```
fix λf.(λx.λy.λz.if z = 0 then z + y else f y x (z - 1))
```

that way beyond the techniques of [8].
The term[1] `twice g` where

```
twice = λf.λx.f (f x)
    g = λy.λx.+ x (y (fix λx.x))
```

will have the strictness and totality property

$$(\top^{\mathtt{Int}} \rightarrow \top^{\mathtt{Int}}) \rightarrow \top^{\mathtt{Int}} \rightarrow \mathbf{b}^{\mathtt{Int}}$$

However, we are *not* able to prove it using the analysis in [12], because we need full power of conjunction in order to construct the proof-tree. The reason is that we need to infer that `twice` has the property

$$((\phi \rightarrow \psi) \wedge (\psi \rightarrow \chi)) \rightarrow (\phi \rightarrow \chi)$$

for any ϕ, ψ, and χ but this is not a well-formed conjunction type in [12]. □

Note that this analysis and the analysis in [12] only differ in the use of conjunction.

[1] Thanks to Nick Benton for pointing to this example

4 Soundness

In this section we will sketch the proof for soundness of the analysis (Figure 5) with respect to the denotational semantics (Figure 2). For the details see the author's forthcoming PhD Thesis.

Definition 2
A subset X of a domain D_σ is *limit closed* if whenever $d_0 \sqsubseteq d_1 \sqsubseteq \cdots$ is a chain in D_σ and $\forall i : d_i \in X$ then $\sqcup_i d_i \in X$ and it is *convex* if whenever $d_1 \sqsubseteq d_2 \sqsubseteq d_3 \in D_\sigma$ and $d_1 \in X$ and $d_3 \in X$ then $d_2 \in X$. □

First we prove that each $[\![\psi^\sigma]\!]$ is a limit-closed and convex subset of D_σ and convex:

Proposition 3 *Limit closed subsets*
$[\![\psi^\sigma]\!]$ is a limit closed and convex subset of D_σ □

Proof We assume that $d_0 \sqsubseteq d_1 \sqsubseteq \cdots$ is a chain in D_σ such that for all i we have $d_i \in [\![\psi^\sigma]\!]$, then we show by induction on the property ψ^σ that $\sqcup_i d_i \in [\![\psi^\sigma]\!]$ holds. And we assume that $d_1 \sqsubseteq d_2 \sqsubseteq d_3$ and both d_1 and d_3 is in $[\![\psi^\sigma]\!]$, then we show by induction on the property ψ^σ that d_2 is in $[\![\psi^\sigma]\!]$. We omit the details. ■

The predicate BOT is sound but it is not complete. Without properties of the form $\downarrow(\phi)$ and un-restricted conjunction or without conjunction we would have an bi-implication in Lemma 4 below. The reason is that ϕ can be an empty property, due to the conjunctions, and hence we cannot define BOT($\downarrow(\phi)$) to be true always — although it is true for all non-empty properties.

Lemma 4
$(\mathrm{BOT}(\psi^\sigma) = \mathtt{tt}) \Rightarrow (\bot_{D_\sigma} \in [\![\psi^\sigma]\!])$ □

Proof We assume that BOT(ψ^σ) is true and then we prove by induction on ψ^σ that $\bot_{D_\sigma} \in [\![\psi^\sigma]\!]$ holds. We omit the details. ■

Next we want to prove that the coercion rules are sound:

Lemma 5 *Soundness of coercions*
If $\psi^\sigma \leq_{\mathrm{ST}} \phi^\sigma$ then $[\![\psi^\sigma]\!] \subseteq [\![\phi^\sigma]\!]$ □

Proof We assume $\psi^\sigma \leq_{\mathrm{ST}} \phi^\sigma$ and then we prove by induction on the proof-tree for $\psi^\sigma \leq_{\mathrm{ST}} \phi^\sigma$ that $[\![\psi^\sigma]\!] \subseteq [\![\phi^\sigma]\!]$ holds. We omit the details. ■

The validity predicate $\models$ is defined for denotations and properties and extended to environments:

Definition 6 *Validity*
$d \models \psi^\sigma \Leftrightarrow (d \in [\![\psi^\sigma]\!])$
For environments we define
$\rho \models A \Leftrightarrow (\mathrm{dom}(A) = \mathrm{dom}(\rho) \wedge \forall\, x^\sigma \in \mathrm{dom}(\rho) : \rho\,(x^\sigma) \models A(x^\sigma))$ □

In the operational world [12] we did not introduce $\downarrow(\phi)$-types as part of the syntax. This is due to the difficulties in defining validity for $\downarrow(\phi)$-types in the operational setting.

Now soundness is:

Proposition 7 *Soundness*
$A \vdash e : \psi^\sigma \Rightarrow (\forall\, \rho : \rho \models A \Rightarrow [\![e]\!]\,\rho \models \psi^\sigma)$ □

Proof We assume $A \vdash e : \psi^\sigma$ and $\rho \models A$ then we show by induction on the proof-tree for $A \vdash e : \psi^\sigma$ that $[\![e]\!]\,\rho \models \psi^\sigma$ holds. We omit the details. ■

In the domain D_σ there are elements d to which there is no term that has d as its standard denotation. In the operational setting all the "meanings" are terms themselves so in order to express properties about the partial results of the fixpoints we find it convenient to introduce special terms approximating the fixpoint. This complicates the soundness proof in the operational setting.

5 Algorithm for Strictness and Totality Property Inference

One way to construct an algorithm for strictness and totality property inference — for both the un-restricted and restricted [12] case — is to follow the *most general type* approach by Hankin and Le Métayer [4]. For a given environment and term the algorithm will find *all* the strictness and totality properties that can be inferred for the term. Often we are only interested in knowing if a term possesses one particular property and not all of them, so this approach seems like using a sledge hammer to crack a nut. We follow the *lazy type* approach of Hankin and Le Métayer [4] where only the information necessary to answer *one* question is calculated.

The algorithm is constructed as follows:

- Make the inference system structural in the term and property. This is achieved by integrating the rule [coer] into all the appropriate rules and axioms.
- Introduce the lazy properties.
- Extract an algorithm from the lazy property inference system.

5.1 Structural Strictness and Totality Inference System

We define an inference system without a coercion rule but where the other rules have the coercion built in.

The coercion-rule maybe needed after the rules [var], [const], [if1], and [abs]. For the [abs]-rule we generate all the possible rules:

$$[\text{abs1s}]\ \frac{}{A \vdash_S \lambda x^\sigma.e : \mathbf{n}^{\sigma\to\tau}} \qquad [\text{abs2s}]\ \frac{}{A \vdash_S \lambda x^\sigma.e : \top^{\sigma\to\tau}}$$

$$[\text{abs3s}]\ \frac{A \vdash_S \lambda x^\sigma.e : \phi^\sigma \to \psi^\tau}{A \vdash_S \lambda x^\sigma.e : {\downarrow}(\phi^\sigma) \to {\downarrow}(\psi^\tau)} \qquad [\text{abs4s}]\ \frac{A, x : \phi^\sigma \vdash_S e : \psi^\tau}{A \vdash_S \lambda x^\sigma.e : \phi^\sigma \to \psi^\tau}$$

$$[\text{abs5s}]\ \frac{A \vdash_S \lambda x^\sigma.e : \phi^\sigma \to {\downarrow}(\psi^\tau)}{A \vdash_S \lambda x^\sigma.e : {\downarrow}(\phi^\sigma \to \psi^\tau)}$$

The last rule can be stated more generally as

$$[\text{downs}]\ \frac{A \vdash_S e : \text{exdown}'(\phi)}{A \vdash_S e : {\downarrow}(\phi)}$$

where we define the functions exdown and exdown′:

$$\begin{array}{llll}
\mathtt{exdown}(\mathbf{n}^\sigma) & = \top^\sigma & \mathtt{exdown}(\mathbf{b}^\sigma) & = \mathbf{b}^\sigma \\
\mathtt{exdown}(\top^\sigma) & = \top^\sigma & \mathtt{exdown}(\phi \to \psi) & = \phi \to \downarrow(\psi) \\
\mathtt{exdown}(\phi \wedge \psi) & = \downarrow(\phi) \wedge \downarrow(\psi) & \mathtt{exdown}(\downarrow(\phi)) & = \downarrow(\phi) \\
\mathtt{exdown}'(\phi \wedge \psi) & = \phi \wedge \psi & \mathtt{exdown}'(\phi) & = \mathtt{exdown}(\phi)
\end{array}$$

The functions moves the $\downarrow$ inwards one level using the coercion rules. The difference between exdown and exdown$'$ is on conjunction: we have

Fact 8

$(\downarrow(\phi) \leq_{\mathrm{L}} \mathtt{exdown}(\phi)) \wedge (\mathtt{exdown}'(\phi) \leq_{\mathrm{L}} \downarrow(\phi))$ □

Proof We will show $(\downarrow(\phi) \leq_{\mathrm{L}} \mathtt{exdown}(\phi))$ and $(\mathtt{exdown}'(\phi) \leq_{\mathrm{L}} \downarrow(\phi))$ by induction on the property ϕ. ■

The rest of the rules ([app], [if2], [if3], [fix], and [conj]) are unchanged. Since the terms are typed therefore no more terms can be typed using these rules than using Figure 5. The reason for changing the presentation is that we are interested in doing as little as possible to check that a term has a given property. In the definition of the analysis (Figure 5) we were looking at the idea of the analysis.

The new structural strictness and property inference system is sound with respect to the strictness and totality inference system:

Lemma 9

$\mathrm{A} \vdash_{\mathrm{S}} \mathtt{e} : \psi \Rightarrow \mathrm{A} \vdash \mathtt{e} : \psi$ □

Proof We show that the rules in the structural inference system can be derived in the non-structural inference system. ■

5.2 Lazy Property Inference System

Following the lazy type approach by Hankin and Le Métayer [4] we now introduce the lazy properties:

$$\mathbf{n}^\tau, \mathbf{b}^\tau, \top^\tau, (\mathrm{A}, \mathtt{e}) \in \mathcal{L}^{\mathrm{L}}_\tau \qquad \frac{\phi^\tau \in \mathcal{L}^{\mathrm{L}}_\tau \quad \psi^\tau \in \mathcal{L}^{\mathrm{L}}_\tau}{\phi^\tau \wedge \psi^\tau \in \mathcal{L}^{\mathrm{L}}_\tau}$$

$$\frac{\psi^\tau \in \mathcal{L}^{\mathrm{L}}_\tau}{\downarrow(\psi^\tau) \in \mathcal{L}^{\mathrm{L}}_\tau} \qquad \frac{\phi^\tau \in \mathcal{L}^{\mathrm{L}}_\tau \quad \psi^\sigma \in \mathcal{L}^{\mathrm{L}}_\sigma}{\phi^\tau \to \psi^\sigma \in \mathcal{L}^{\mathrm{L}}_{\tau \to \sigma}}$$

The property (A, e) is a shorthand (un-evaluated property) for the conjunction of all the properties ϕ that can be inferred for e using the environment A. The function expand maps lazy properties to strictness and totality properties due to [4]:

$$\begin{array}{ll}
\mathtt{expand} & :: \mathcal{L}^{\mathrm{L}}_\tau \to \mathcal{L}^{\mathrm{ST}}_\tau \\
\mathtt{expand}(\mathbf{n}^\tau) & = \mathbf{n}^\tau \\
\mathtt{expand}(\mathbf{b}^\tau) & = \mathbf{b}^\tau \\
\mathtt{expand}(\top^\tau) & = \top^\tau \\
\mathtt{expand}(\downarrow(\phi)) & = \downarrow(\mathtt{expand}(\phi)) \\
\mathtt{expand}(\phi \to \psi) & = \mathtt{expand}(\phi) \to \mathtt{expand}(\psi) \\
\mathtt{expand}(\phi \wedge \psi) & = \mathtt{expand}(\phi) \wedge \mathtt{expand}(\psi) \\
\mathtt{expand}((\mathrm{A}, \mathtt{e})) & = \bigwedge \{\phi \mid \mathtt{expand}(\mathrm{A}) \vdash_{\mathrm{S}} \mathtt{e} : \phi\}
\end{array}$$

and the function is extended to environments in a component-wise manner. The predicate BOT on lazy properties is defined by

$$\mathrm{BOT}((\mathrm{A}, \mathtt{e})) = \mathrm{BOT}(\mathtt{expand}((\mathrm{A}, \mathtt{e})))$$

We need two new rules for relating the lazy properties:

$$[\mathrm{env_L}]\ \frac{\mathrm{A} \vdash_{\mathrm{L}} \mathtt{e} : \phi}{(\mathrm{A}, \mathtt{e}) \leq_{\mathrm{L}} \phi}\ \text{if } \phi \neq (\mathrm{A}', \mathtt{e}') \quad [\mathrm{env_R}]\ \frac{\forall \phi : \mathrm{A} \vdash_{\mathrm{L}} \mathtt{e} : \phi \Rightarrow (\psi \leq_{\mathrm{L}} \phi)}{\psi \leq_{\mathrm{L}} (\mathrm{A}, \mathtt{e})}$$

The lazy properties are useful in the application rule:

$$[\mathrm{app_L}]\ \frac{\mathrm{A} \vdash_{\mathrm{S}} \mathtt{e}_1 : (\mathrm{A}, \mathtt{e}_2) \rightarrow \phi^{\tau}}{\mathrm{A} \vdash_{\mathrm{S}} \mathtt{e}_1\ \mathtt{e}_2 : \phi^{\tau}}$$

In the [if1]-rule we no longer construct proof-trees for $\mathtt{e}_2$ and $\mathtt{e}_3$:

$$[\mathrm{if1_L}]\ \frac{\mathrm{A} \vdash_{\mathrm{S}} \mathtt{e}_1 : \mathbf{b}^{\mathtt{Bool}} \quad \mathbf{b}^{\sigma} \leq_{\mathrm{ST}} \psi}{\mathrm{A} \vdash_{\mathrm{S}} \mathtt{if}\ \mathtt{e}_1\ \mathtt{then}\ \mathtt{e}_2\ \mathtt{else}\ \mathtt{e}_3 : \psi}$$

All the other rules remain the same.

The lazy property inference system is sound with respect to the structural strictness and totality inference system:

Lemma 10

$(((\phi \leq_{\mathrm{L}} \psi) \wedge \mathrm{A} \vdash_{\mathrm{L}} \mathtt{e} : \phi') \Rightarrow$
$((\mathtt{expand}(\phi) \leq_{\mathrm{ST}} \mathtt{expand}(\psi)) \wedge \mathtt{expand}(\mathrm{A}) \vdash_{\mathrm{S}} \mathtt{e} : \mathtt{expand}(\phi')))$ □

Proof

Suppose $\phi \leq_{\mathrm{L}} \psi$ and $\mathrm{A} \vdash_{\mathrm{L}} \mathtt{e} : \phi$, then we will show $\mathtt{expand}(\phi) \leq_{\mathrm{ST}} \mathtt{expand}(\psi)$ and $\mathtt{expand}(\mathrm{A}) \vdash_{\mathrm{S}} \mathtt{e} : \mathtt{expand}(\phi)$ by simultaneous induction on the proof-tree for $\leq_{\mathrm{L}}$ and $\vdash_{\mathrm{L}}$.

The reason that we have to do simultaneous induction on $\leq_{\mathrm{L}}$ and $\vdash_{\mathrm{L}}$ is that $\leq_{\mathrm{L}}$ depends on $\vdash_{\mathrm{L}}$ (in [$\mathrm{env_L}$] and [$\mathrm{env_R}$]) and not only $\vdash_{\mathrm{L}}$ depending on $\leq_{\mathrm{L}}$ (in [$\mathrm{var_L}$], [$\mathrm{if1_L}$], [$\mathrm{const_L}$]) as in the other inference systems. We omit the details. ■

5.3 The Lazy Property Inference Algorithm

Finally the last step is to extract the algorithm $\mathcal{T}$ (see Figure 6) for lazy property inference from the lazy property inference system. The function $\mathcal{ALL}(\sigma)$ gives a list of all the strictness and totality properties of the type σ. When the function $\mathcal{T}$ is applied to a list it gives back the list of all those strictness and totality properties for which the function is true; the function $\mathcal{ISCHAIN}(\phi)$ applied to a list of properties gives back the list of those properties that are chains with the property ϕ as the last one and BOT of the first is true. The algorithm for checking the coercions is displayed in Figure 7 and 8.

The algorithms are sound with respect to the lazy property inference system:

Lemma 11

$(\mathcal{T}(\mathrm{A}, \mathtt{e}, \phi') = \mathtt{tt} \wedge \mathcal{I}(\phi, \psi) = \mathtt{tt}) \Rightarrow (\mathrm{A} \vdash_{\mathrm{L}} \mathtt{e} : \phi' \wedge (\phi \leq_{\mathrm{L}} \psi))$ □

$$
\begin{array}{ll}
\mathcal{T}(\mathrm{A}, \mathtt{e}, \phi \wedge \psi) & = \mathcal{T}(\mathrm{A}, \mathtt{e}, \phi) \wedge \mathcal{T}(\mathrm{A}, \mathtt{e}, \psi) \\
\mathcal{T}(\mathrm{A}, \lambda \mathtt{x}^{\sigma}.\mathtt{e}, \mathbf{n}^{\sigma \to \tau}) & = \mathtt{tt} \\
\mathcal{T}(\mathrm{A}, \lambda \mathtt{x}^{\sigma}.\mathtt{e}, \top^{\sigma \to \tau}) & = \mathtt{tt} \\
\mathcal{T}(\mathrm{A}, \lambda \mathtt{x}^{\sigma}.\mathtt{e}, \mathbf{b}^{\sigma \to \tau}) & = \mathtt{ff} \\
\mathcal{T}(\mathrm{A}, \lambda \mathtt{x}^{\sigma}.\mathtt{e}, \downarrow(\phi) \to \downarrow(\psi)) & = \mathcal{T}((\mathtt{x}^{\sigma} : \phi) : \mathrm{A}, \mathtt{e}, \psi) \vee \mathcal{T}((\mathtt{x}^{\sigma} : \downarrow(\phi)) : \mathrm{A}, \mathtt{e}, \downarrow(\psi)) \\
\mathcal{T}(\mathrm{A}, \lambda \mathtt{x}^{\sigma}.\mathtt{e}, \phi \to \psi) & = \mathcal{T}((\mathtt{x}^{\sigma} : \phi) : \mathrm{A}, \mathtt{e}, \psi) \\
\mathcal{T}(\mathrm{A}, \mathtt{e}_1\, \mathtt{e}_2, \psi) & = \mathcal{T}(\mathrm{A}, \mathtt{e}_1, (\mathrm{A}, \mathtt{e}_2) \to \psi) \\
\mathcal{T}(\mathrm{A}, \mathtt{x}^{\sigma}, \psi) & = \mathcal{I}(\mathrm{A}(\mathtt{x}^{\sigma}), \psi) \\
\mathcal{T}(\mathrm{A}, \mathtt{if}\ \mathtt{e}_1\ \mathtt{then}\ \mathtt{e}_2\ \mathtt{else}\ \mathtt{e}_3, \phi) & = (\mathcal{T}(\mathrm{A}, \mathtt{e}_1, \mathbf{b}^{\mathtt{Bool}}) \wedge \mathcal{I}(\mathbf{b}^{\sigma}, \phi)) \vee \\
 & \quad (\mathcal{T}(\mathrm{A}, \mathtt{e}_2, \phi) \wedge \mathcal{T}(\mathrm{A}, \mathtt{e}_3, \phi)) \wedge \\
 & \quad (\mathcal{T}(\mathrm{A}, \mathtt{e}_1, \mathbf{n}^{\mathtt{Bool}}) \vee (\mathcal{T}(\mathrm{A}, \mathtt{e}_1, \top^{\mathtt{Bool}}) \wedge \mathrm{BOT}(\phi))) \\
\mathcal{T}(\mathrm{A}, \mathtt{c}, \phi) & = \mathcal{I}(\psi_{\mathtt{c}}, \phi) \\
\mathcal{T}(\mathrm{A}, \mathtt{fix}\ \mathtt{e}, \phi) & = \mathcal{FIX}(\mathrm{A}, \mathtt{e}, \phi) \\
\mathcal{T}(\mathrm{A}, \mathtt{e}, \downarrow(\phi)) & = \mathcal{T}(\mathrm{A}, \mathtt{e}, \mathtt{exdown}'(\phi)) \\
\mathcal{FIX}(\mathrm{A}, \mathtt{e}, \phi^{\sigma}) & = \text{let } l1 = \mathcal{T}(\mathrm{A}, \mathtt{e}, \mathcal{ALL}(\sigma)) \\
 & \qquad l2 = \mathcal{ISCHAIN}(\phi^{\sigma}, l1) \\
 & \quad \text{in } l2 \neq [\,]
\end{array}
$$

Fig. 6. Lazy Property Inference Algorithm

$$
\begin{array}{ll|ll}
\mathcal{I}(\phi^{\sigma}, \psi^{\tau}) & = \mathtt{ff} & \mathcal{I}(\phi, \phi) & = \mathtt{tt} \\
\mathcal{I}(\phi, \top^{\sigma}) & = \mathtt{tt} & \mathcal{I}(\mathbf{n}^{\sigma}, \mathbf{b}^{\sigma}) & = \mathtt{ff} \\
\mathcal{I}(\mathbf{b}^{\sigma}, \mathbf{n}^{\sigma}) & = \mathtt{ff} & \mathcal{I}(\top^{\sigma}, \mathbf{n}^{\sigma}) & = \mathtt{ff} \\
\mathcal{I}(\top^{\sigma}, \mathbf{b}^{\sigma}) & = \mathtt{ff} & \mathcal{I}(\downarrow(\mathbf{b}^{\sigma}), \mathbf{b}^{\sigma}) & = \mathtt{tt} \\
\mathcal{I}(\phi \to \psi, \mathbf{b}^{\sigma \to \tau}) & = \mathtt{ff} & \mathcal{I}(\downarrow(\phi), \mathbf{n}^{\sigma}) & = \mathtt{ff} \\
\mathcal{I}(\phi', \phi \wedge \psi) & = \mathcal{I}(\phi', \phi) \wedge \mathcal{I}(\phi', \psi) & \mathcal{I}(\phi \to \psi, \mathbf{n}^{\sigma \to \tau}) & = \mathcal{I}(\mathbf{n}^{\sigma}, \phi) \wedge \mathcal{I}(\psi, \mathbf{n}^{\tau}) \\
\mathcal{I}(\phi \wedge \psi, \mathbf{n}^{\sigma}) & = \mathcal{I}(\phi, \mathbf{n}^{\sigma}) \vee \mathcal{I}(\psi, \mathbf{n}^{\sigma}) & \mathcal{I}(\phi \wedge \psi, \mathbf{b}^{\sigma}) & = \mathcal{I}(\phi, \mathbf{b}^{\sigma}) \vee \mathcal{I}(\psi, \mathbf{b}^{\sigma}) \\
\mathcal{I}(\mathbf{n}^{\sigma \to \tau}, \phi \to \psi) & = \mathcal{I}(\top^{\tau}, \psi) & \mathcal{I}(\mathbf{b}^{\sigma \to \tau}, \phi \to \psi) & = \mathcal{I}(\top^{\tau}, \psi) \vee \mathcal{I}(\mathbf{b}^{\tau}, \psi) \\
\mathcal{I}(\top^{\sigma \to \tau}, \phi \to \psi) & = \mathcal{I}(\top^{\tau}, \psi) & \mathcal{I}(\mathbf{n}^{\sigma}, \downarrow(\phi)) & = \mathcal{I}(\mathbf{n}^{\sigma}, \phi) \\
\mathcal{I}(\mathbf{b}^{\sigma}, \downarrow(\phi)) & = \mathcal{I}(\mathbf{b}^{\sigma}, \phi) \vee (\phi = \mathbf{n}^{\sigma}) & \mathcal{I}(\top^{\sigma}, \downarrow(\phi)) & = \mathcal{I}(\top^{\sigma}, \phi) \vee (\phi = \mathbf{n}^{\sigma}) \\
\mathcal{I}(\phi \to \psi, \downarrow(\phi')) & = \mathcal{I}(\phi \to \psi, \phi') & &
\end{array}
$$

Fig. 7. Lazy Coercion Inference Algorithm (Part 1)

Proof We will assume that both $\mathcal{T}(\mathrm{A}, \mathtt{e}, \phi')$ and $\mathcal{I}(\phi, \psi)$ are true and then we will prove $\mathrm{A} \vdash_{\mathrm{L}} \mathtt{e} : \phi'$ and $\phi \leq_{\mathrm{L}} \psi$ by induction on $\mathtt{e}$, ϕ', and ψ. We omit the details. ■

Finally we have that the inference algorithm is sound with respect to the strictness and totality property inference system:

Theorem 12
$\mathcal{T}(\mathrm{A}, \mathtt{e}, \phi) \Rightarrow \mathtt{expand}(\mathrm{A}) \vdash \mathtt{e} : \mathtt{expand}(\phi)$ □

Proof This is a consequence of Lemma 9, 10, and 11. ■

The inference algorithm is not complete. Consider the term, $\mathtt{e}$, $\lambda\mathtt{x}.(\mathtt{fix}\ \lambda\mathtt{x}.\mathtt{x})$

$$
\begin{aligned}
\mathcal{I}(\phi^{\sigma} \rightarrow \psi^{\tau}, \phi'^{\sigma} \rightarrow \psi'^{\tau}) &= (\mathcal{I}(\phi', \phi) \wedge \mathcal{I}(\psi, \psi')) \vee (\mathcal{I}(\phi', \downarrow(\phi)) \wedge \mathcal{I}(\downarrow(\psi), \psi')) \vee \\
&\quad \mathcal{I}(\top^{\tau}, \psi') \\
\mathcal{I}(\downarrow(\phi), \downarrow(\psi)) &= \mathcal{I}(\phi, \psi) \vee \mathcal{I}(\mathtt{exdown}(\phi), \downarrow(\psi)) \vee \mathcal{I}(\downarrow(\phi), \mathtt{exdown}'(\psi)) \\
&\quad \vee \mathcal{I}(\mathtt{exdown}(\phi), \mathtt{exdown}'(\psi)) \\
\mathcal{I}(\phi \wedge \psi, \phi' \rightarrow \psi') &= \mathcal{I}(\phi, \phi' \rightarrow \psi') \vee \mathcal{I}(\psi, \phi' \rightarrow \psi') \vee \mathcal{I}(\top^{\tau}, \psi') \\
\mathcal{I}(\downarrow(\phi), \phi'^{\sigma} \rightarrow \psi'^{\tau}) &= \mathcal{I}(\top^{\tau}, \psi') \vee (\phi = \mathbf{b}^{\sigma \dot{\rightarrow} \tau} \wedge \mathcal{I}(\mathbf{b}^{\tau}, \psi')) \vee \\
&\quad (\phi = \phi'' \rightarrow \psi'' \wedge \mathcal{I}(\phi'' \rightarrow \downarrow(\psi''), \phi' \rightarrow \psi')) \vee \\
&\quad (\phi = (\phi'' \wedge \psi'') \wedge \mathcal{I}(\downarrow(\phi'') \wedge \downarrow(\psi''), \phi' \rightarrow \psi')) \\
\mathcal{I}(\phi \wedge \psi, \downarrow(\phi')) &= \mathcal{I}(\phi \wedge \psi, \phi') \vee \mathcal{I}(\phi, \downarrow(\phi')) \vee \mathcal{I}(\psi, \downarrow(\phi')) \\
\mathcal{I}((\mathrm{A}, \mathtt{e}), \phi) &= \mathcal{T}(\mathrm{A}, \mathtt{e}, \phi) \\
\mathcal{I}(\psi^{\sigma}, (\mathrm{A}, \mathtt{e})) &= \mathcal{C}(\mathcal{ALL}(\sigma), \mathrm{A}, \mathtt{e}, \psi) \\
\mathcal{C}([\,], \mathrm{A}, \mathtt{e}, \psi) &= \mathtt{tt} \\
\mathcal{C}(\psi{:}l, \mathrm{A}, \mathtt{e}, \phi) &= \begin{cases} \mathcal{I}(\phi, \psi) \wedge \mathcal{C}(l, \mathrm{A}, \mathtt{e}, \phi), & \text{if } \mathcal{T}(\mathrm{A}, \mathtt{e}, \psi) \\ \mathcal{C}(l, \mathrm{A}, \mathtt{e}, \phi), & \text{otherwise} \end{cases}
\end{aligned}
$$

Fig. 8. Lazy Coercion Inference Algorithm (Part 2)

and the property, ϕ, $\downarrow((\mathbf{n}^{\sigma} \rightarrow \mathbf{n}^{\tau}) \wedge (\mathbf{n}^{\sigma} \rightarrow \top^{\tau}))$. In the analysis (Figure 5) we can construct a proof-tree for $\emptyset \vdash \mathtt{e} : \phi$. However, it is not possible to infer

$$\emptyset \vdash \mathtt{e} : (\mathbf{n}^{\sigma} \rightarrow \mathbf{n}^{\tau}) \wedge (\mathbf{n}^{\sigma} \rightarrow \top^{\tau})$$

The algorithm will do as follows:

$$
\begin{aligned}
\mathcal{T}(\emptyset, \mathtt{e}, \phi) &= \mathcal{T}(\emptyset, \mathtt{e}, (\mathbf{n}^{\sigma} \rightarrow \mathbf{n}^{\tau}) \wedge (\mathbf{n}^{\sigma} \rightarrow \top^{\tau})) \\
&= \mathcal{T}(\emptyset, \mathtt{e}, (\mathbf{n}^{\sigma} \rightarrow \mathbf{n}^{\tau})) \wedge \mathcal{T}(\emptyset, \mathtt{e}, (\mathbf{n}^{\sigma} \rightarrow \top^{\tau})) \\
&= \mathtt{ff} \wedge \mathtt{tt} = \mathtt{ff}
\end{aligned}
$$

The problem is that $\downarrow(\phi \wedge \psi)$ is not equivalent to $\phi \wedge \psi$. More work is needed to find a sound *and* complete inference algorithm.

6 Conclusion

We have restated the strictness and totality analysis of [12] and removed the restriction that conjunction may only occur at the top-level. We have proven the strictness and totality analysis correct with respect to a denotational semantics. Finally we have constructed an algorithm for inferring the strictness and totality properties by following the lazy types approach of [4].

Acknowledgement The author has been funded by Odense University, Denmark. I would like to thank LOMAPS (Esprit Basic Research) and DART (Danish Science Research Council) for partial support. Part of the research was carried out while visiting Cambridge University, England; the visit was supported by the Danish Research Academy. I am grateful to Nick Benton, David Wright, Hanne Riis Nielson and Flemming Nielson for interesting discussions.

References

1. Samson Abramsky. Abstract interpretation, logical relations and Kan extensions. *Journal of Logic and Computation*, 1(1):5–39, 1990.
2. Nick Benton. *Strictness Analysis of Functional Programs*. PhD thesis, University of Cambridge, 1993. Available as Technical Report No. 309.
3. Geoffrey L. Burn, Chris Hankin, and Samson Abramsky. Strictness Analysis for Higher-order Functions. *Science of Computer Programming*, 7:249–278, 1986.
4. Chris Hankin and Daniel Le Métayer. Deriving algorithms from type inference systems: Application to strictness analysis. In *Proceedings of POPL'94*, pages 202 – 212, 1994.
5. Chris Hankin and Daniel Le Métayer. A Type-based framework for Program Analysis. In *Proceedings of SAS'94*, LNCS 864, pages 380–394, 1994.
6. Thomas P. Jensen. Strictness analysis in logical form. In *Proceedings of FPCA'91*, LNCS 523, pages 352 – 366, 1991.
7. Thomas P. Jensen. Disjunctive strictness analysis. In *Proceedings of LICS'92*, pages 174 – 185, 1992.
8. Tsung-Min Kuo and Prateek Mishra. Strictness analysis: A new perspective based on type inference. In *Proceedings of FPCA'89*, pages 260 – 272. ACM Press, 1989.
9. Alan Mycroft. The theory and practice of transforming call-by-need into call-by-value. In *Proceedings of the 4th International Symposium on Programming*, LNCS 83, pages 269–281, 1980.
10. Alan Mycroft. *Abstract Interpretation and Optimising Transformation for Applicative programs*. PhD thesis, University of Edinburgh, Scotland, 1981.
11. David Sands. Complexity analysis for a lazy higher-order language. In *Proceedings of ESOP'90*, LNCS 432, pages 361–376, 1990.
12. Kirsten Lackner Solberg, Hanne Riis Nielson, and Flemming Nielson. Strictness and totality analysis. In *Proceedings of SAS'94*, LNCS 864, pages 408 – 422, 1994.
13. D. A. Turner. Miranda: A non-strict functional language with polymorphic types. In *Proceedings of FPCA'85*, LNCS 201, pages 1 – 16, 1985.
14. Phil Wadler and John Hughes. Projections for strictness analysis. In *Proceedings of FPCA'87*, LNCS 27, 1987.
15. David A. Wright. A new technique for strictness analysis. In *Proceedings TAPSOFT'91*, LNCS 494, pages 260 – 272. Springer Verlag, 1991.
16. David A. Wright. *Reduction Types and Intensionality in the Lambda-Calculus*. PhD thesis, University of Tasmania, 1992.