Logical Predicates in Higher-Order Mathematical Operational Semantics

Sergey Goncharov[1,⋆], Alessio Santamaria[2], Lutz Schröder[1,⋆⋆], Stelios Tsampas[1(✉),⋆⋆⋆] and Henning Urbat[1,†]

[1] Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany
{sergey.goncharov,lutz.schroder,stelios.tsampas@fau.de, henning.urbat}@fau.de

[2] University of Sussex, Brighton, UK
a.santamaria@sussex.ac.uk

Abstract. We present a systematic approach to logical predicates based on universal coalgebra and higher-order abstract GSOS, thus making a first step towards a unifying theory of logical relations. We start with the observation that logical predicates are special cases of *coalgebraic invariants* on mixed-variance functors. We then introduce the notion of a *locally maximal logical refinement* of a given predicate, with a view to enabling inductive reasoning, and identify sufficient conditions on the overall setup in which locally maximal logical refinements canonically exist. Finally, we develop induction-up-to techniques that simplify inductive proofs via logical predicates on systems encoded as (certain classes of) higher-order GSOS laws by identifying and abstracting away from their boiler-plate part.

1 Introduction

Logical relations are arguably the most widely used method for reasoning on higher-order languages. Historically, early examples of logical relations [44,46,47,51,55,56,58,59] were based on denotational semantics, before the method evolved into logical relations based on operational semantics [7,17,34,50,52,53]. Today, operationally-based logical relations are ubiquitous and serve purposes ranging from strong normalization proofs [6] and safety properties [21,22] to reasoning about contextual equivalence [5,60] and formally verified compilation [8,33,45,48], in a variety of settings such as effectful [37], probabilistic [4,10,63], and differential programming [15,40,41].

Unfortunately, despite the extensive literature, there is a distinct lack of a general formal theory of (operational) logical relations. As a reasoning method, logical relations are applied in a largely empirical manner, more so because their core principles are well understood on an intuitive level. For example, there is typically no formal notion of a logical predicate or relation; instead, if a predicate or relation is defined by induction on

⋆ Supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 501369690

⋆⋆ Supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project numbers 419850228

⋆⋆⋆ Supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project numbers 419850228 and 527481841

† Supported by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 470467389

N. Kobayashi and J. Worrell (Eds.): FoSSaCS 2024, LNCS 14575, pp. 47–69, 2024.
https://doi.org/10.1007/978-3-031-57231-9_3

types and maps "related inputs to related outputs", it then meets the informal criteria to be called "logical". However, the empirical character of logical relations is problematic for two main reasons: (i) complex machinery associated to logical relations needs to be re-established anew on a per-case basis, and (ii) it is hard to abstract and simplify said machinery, even though certain parts of proofs via logical relations seem generic.

Recently, *Higher-order Mathematical Operational Semantics* [24], or *higher-order abstract GSOS*, has emerged as a unifying approach to the operational semantics of higher-order languages. In this framework, languages are represented as *higher-order GSOS laws*, a form of distributive law of a syntax functor Σ over a mixed-variance behaviour bifunctor B. In further work [62], an abstract form of *Howe's method* [16,31,32] for higher-order abstract GSOS has been identified, in which an otherwise complex and application-specific operational technique is, at the same time, lifted to an appropriate level of generality and reduced to a simple *lax bialgebra* condition.

In the present paper, we work towards establishing a theory of logical relations based on coalgebra and higher-order abstract GSOS, starting from *logical predicates*, understood as unary logical relations. In more detail, we present the following contributions:

(i) A systematization of the method of logical predicates (Section 3), achieved by
 (a) identifying logical predicates as certain coalgebraic invariants (Definition 12), parametric in a predicate lifting of the underlying mixed-variance bifunctor,
 (b) introducing the *locally maximal logical refinement* $\Box P$ of a predicate P (Definition 14), which enables inductive proofs of $\Box P$, and
 (c) identifying an abstract setting in which locally maximal logical refinements of predicates exist and are unique (Section 3.3).

(ii) The development of efficient reasoning techniques on logical predicates, which we call *induction up-to* (Theorems 34 and 36), for higher-order GSOS laws satisfying a *relative flatness* condition (Definition 30).

We illustrate (ii) by providing proofs of strong normalization for typed combinatory logic and type safety for the simply typed λ-calculus which, thanks to the use of our up-to techniques, are significantly shorter and simpler than standard arguments found in the literature. Finally, we exploit the genericity of our framework to study strong normalization on the level of higher-order GSOS laws (Theorem 42). We note that the implementation of typed languages as higher-order GSOS laws as such is also novel.

Full proofs and additional details can be found in the arXiv version [25] of our paper.

Related work While denotational logical relations have been studied in categorical generality, e.g. [27,28,29,38], general abstract foundations of operational logical relations are far less developed. In recent work [13,14], Dagnino and Gavazzo introduce a categorical notion of operational logical relations that is largely orthogonal to ours, in particular regarding the parametrization of the framework: In *op. cit.*, the authors work with a fixed *fine-grain call-by-value* language [42], parametrized by a signature of generic effects, while the notion of logical relation is kept variable and in fact is parametrized over a fibration; contrastingly, we keep to the traditional notion of logical relation but parametrize over the syntax and semantics of the language. Moreover, we work with a small-step operational semantics, whereas the semantics used in *op. cit.* is an axiomatically defined categorical evaluation semantics.

2 Preliminaries

2.1 Category Theory

Familiarity with basic category theory [43] (e.g. functors, natural transformations, (co)limits, monads) is assumed. We review some concepts and notation.

Notation. Given objects X_1, X_2 in a category C, we write $X_1 \times X_2$ for the product and $\langle f_1, f_2\rangle\colon X \to X_1 \times X_2$ for the pairing of $f_i\colon X \to X_i$, $i = 1, 2$. We let $X_1 + X_2$ denote the coproduct, $\mathsf{inl}\colon X_1 \to X_1 + X_2$ and $\mathsf{inr}\colon X_2 \to X_1 + X_2$ the injections, $[g_1, g_2]\colon X_1 + X_2 \to X$ the copairing of $g_i\colon X_i \to X$, $i = 1, 2$, and $\nabla = [\mathsf{id}_X, \mathsf{id}_X]\colon X + X \to X$ the codiagonal. The *slice category* C/X, where $X \in C$, has as objects all pairs (Y, p_Y) of an object $Y \in C$ and a morphism $p_Y\colon Y \to X$, and a morphism from (Y, p_Y) to (Z, p_Z) is a morphism $f\colon Y \to Z$ of C such that $p_Y = p_Z \cdot f$. The *coslice category* X/C is defined dually.

Extensive categories. A category C is *(finitely) extensive* [12] if it has finite coproducts and for every finite family of objects X_i ($i \in I$) the functor $E\colon \prod_{i\in I} C/X_i \to C/\coprod_{i\in I} X_i$ sending $(p_i\colon Y_i \to X_i)_{i\in I}$ to $\coprod_{i\in I} p_i\colon \coprod_i Y_i \to \coprod_i X_i$ is an equivalence of categories. A *countably extensive* category satisfies the analogous property for countable coproducts. In extensive categories, coproduct injections $\mathsf{inl}, \mathsf{inr}$ are monic, and coproducts of monomorphisms are monic; generally, coproducts behave like disjoint unions of sets.

Example 1. Examples of countably extensive categories include the category **Set** of sets and functions; the category $\mathbf{Set}^C$ of presheaves on a small category C and natural transformations; and the categories of posets and monotone maps, nominal sets and equivariant maps, and metric spaces and non-expansive maps, respectively.

Algebras. Given an endofunctor F on a category C, an *F-algebra* is a pair (A, a) consisting of an object A and a morphism $a\colon FA \to A$ (the *structure*). A *morphism* from (A, a) to an F-algebra (B, b) is a morphism $h\colon A \to B$ of C such that $h \cdot a = b \cdot Fh$. Algebras for F and their morphisms form a category $\mathbf{Alg}(F)$, and an *initial* F-algebra is simply an initial object in that category. We denote the initial F-algebra by μF if it exists, and its structure by $\iota\colon F(\mu F) \to \mu F$. Initial algebras admit the *structural induction principle*: the algebra μF has no proper subalgebras, that is, every F-algebra monomorphism $m\colon (A, a) \rightarrowtail (\mu F, \iota)$ is an isomorphism.

More generally, a *free F-algebra* on an object X of C is an F-algebra $(F^\star X, \iota_X)$ together with a morphism $\eta_X\colon X \to F^\star X$ of C such that for every algebra (A, a) and every $h\colon X \to A$ in C, there exists a unique F-algebra morphism $h^\sharp\colon (F^\star X, \iota_X) \to (A, a)$ such that $h = h^\sharp \cdot \eta_X$. If free algebras exist on every object, their formation induces a monad $F^\star\colon C \to C$, the *free monad* generated by F. Every F-algebra (A, a) yields an Eilenberg-Moore algebra $\widehat{a}\colon F^\star A \to A$ as the free extension of $\mathsf{id}_A\colon A \to A$.

The most familiar example of functor algebras are algebras for a signature. Given a set S of *sorts*, an *S-sorted algebraic signature* consists of a set Σ of operation symbols together with a map $\mathsf{ar}\colon \Sigma \to S^\star \times S$ associating to every $\mathsf{f} \in \Sigma$ its *arity*. We write $\mathsf{f}\colon s_1 \times \cdots \times s_n \to s$ if $\mathsf{ar}(\mathsf{f}) = (s_1, \ldots, s_n, s)$, and $\mathsf{f}\colon s$ if $n = 0$ (in which case f is called a *constant*). Every signature Σ induces a polynomial functor on the category $\mathbf{Set}^S$ of S-sorted sets, denoted by the same letter Σ, given by $(\Sigma X)_s = \coprod_{\mathsf{f}\colon s_1\cdots s_n \to s} \prod_{i=1}^n X_{s_i}$ for $X \in \mathbf{Set}^S$ and $s \in S$. An algebra for the functor Σ is precisely an algebra for

the signature Σ, viz. an S-sorted set $A = (A_s)_{s\in S}$ in $\mathbf{Set}^S$ equipped with an operation $\mathsf{f}^A\colon \prod_{i=1}^n A_{s_i} \to A_s$ for every $\mathsf{f}\colon s_1 \cdots s_n \to s$ in Σ. Morphisms of Σ-algebras are S-sorted maps respecting the algebraic structure. Given an S-sorted set X of variables, the free algebra $\Sigma^\star X$ is the Σ-algebra of Σ-terms with variables from X; more precisely, $(\Sigma^\star X)_s$ is inductively defined by $X_s \subseteq (\Sigma^\star X)_s$ and $\mathsf{f}(t_1,\dots,t_n) \in (\Sigma^\star X)_s$ for all $\mathsf{f}\colon s_1\cdots s_n \to s$ and $t_i \in (\Sigma^\star X)_{s_i}$. In particular, the free algebra on the empty set is the initial algebra $\mu\Sigma$; it is formed by all *closed terms* of the signature. For every Σ-algebra (A,a), the induced Eilenberg-Moore algebra $\widehat{a}\colon \Sigma^\star A \to A$ is given by the map that evaluates terms over A in the algebra A.

Coalgebras. Dual to the notion of algebra, a *coalgebra* for an endofunctor F on C is a pair (C,c) of an object C (the *state space*) and a morphism $c\colon C \to FC$ (the *structure*).

2.2 Higher-Order Abstract GSOS

We summarize the framework of higher-order abstract GSOS [24], which extends the original, first-order counterpart introduced by Turi and Plotkin [61]. In higher-order abstract GSOS, the operational semantics of a higher-order language is presented in the form of a *higher-order GSOS law*, a categorical structure parametric in

(1) a category C with finite products and coproducts;

(2) an object $V \in C$ of *variables*;

(3) an endofunctor $\Sigma\colon C \to C$, where $\Sigma = V + \Sigma'$ for some endofunctor Σ', such that free Σ-algebras exist on every object (hence Σ generates a free monad $\Sigma^\star$);

(4) a mixed-variance bifunctor $B\colon C^{\mathsf{op}} \times C \to C$.

The functors Σ and B represent the *syntax* and the *behaviour* of a higher-order language. The motivation behind B having two arguments is that transitions have labels, which behave contravariantly, and poststates, which behave covariantly; in term models the objects of labels and states will coincide. The presence of an object V of variables is a technical requirement for the modelling of languages with variable binding [19,20], such as the λ-calculus. An object of V/C, the coslice category of *V-pointed objects*, is thought of as a set X of programs with an embedding $p_X\colon V \to X$ of the variables. In point-free calculi, e.g. **xTCL** as introduced below, we put $V = 0$ (the initial object).

Definition 2. A *(V-pointed) higher-order GSOS law* of Σ over B is a family of morphisms (1) that is dinatural in $(X,p_X) \in V/C$ and natural in $Y \in C$:

$$\varrho_{(X,p_X),Y}\colon \Sigma(X \times B(X,Y)) \to B(X, \Sigma^\star(X+Y)) \tag{1}$$

Notation 3. (i) In (1), we have implicitly applied the forgetful functor $V/C \to C$ at (X,p_X). In addition, we write $\varrho_{X,Y}$ for $\varrho_{(X,p_X),Y}$ if the point p_X is clear from the context.
(ii) For $(A,a) \in \mathbf{Alg}(\Sigma)$, we view A as V-pointed by $p_A = (V \xrightarrow{\mathsf{inl}} V + \Sigma' A = \Sigma A \xrightarrow{a} A)$.

Informally, $\varrho_{X,Y}$ assigns to an operation of the language with formal arguments from X having specified next-step behaviours in $B(X,Y)$ (i.e. with labels in X and formal poststates in Y) a next-step behaviour in $B(X,\Sigma^\star(X+Y))$, i.e. with the same labels, and with poststates being program terms mentioning variables from both X and Y. Every

$$\frac{}{\mathsf{e} \overset{\checkmark}{\longrightarrow}} \qquad \frac{}{S_{\tau_1,\tau_2,\tau_3} \overset{t}{\longrightarrow} S'_{\tau_1,\tau_2,\tau_3}(t)} \qquad \frac{}{S'_{\tau_1,\tau_2,\tau_3}(p) \overset{t}{\longrightarrow} S''_{\tau_1,\tau_2,\tau_3}(p,t)}$$

$$\frac{}{S''_{\tau_1,\tau_2,\tau_3}(p,q) \overset{t}{\longrightarrow} (p\,t)\,(q\,t)} \qquad \frac{}{K_{\tau_1,\tau_2} \overset{t}{\longrightarrow} K'_{\tau_1,\tau_2}(t)} \qquad \frac{}{K'_{\tau_1,\tau_2}(p) \overset{t}{\longrightarrow} p}$$

$$\frac{}{I_\tau \overset{t}{\longrightarrow} t} \qquad \frac{p \to p'}{p\,q \to p'\,q} \qquad \frac{p \overset{q}{\longrightarrow} p'}{p\,q \to p'}$$

Fig. 1. (Call-by-name) operational semantics of **xTCL**.

higher-order GSOS law (1) induces a canonical *operational model* $\gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$, viz. a $B(\mu\Sigma, -)$-coalgebra on the initial algebra $\mu\Sigma$, defined by *primitive recursion* [36, Prop. 2.4.7] as the unique morphism γ making the following diagram commute:

$$\begin{array}{ccccc} \Sigma(\mu\Sigma) & & \xrightarrow{\qquad\qquad\iota\qquad\qquad} & & \mu\Sigma \\ {\scriptstyle \Sigma\langle \mathsf{id}, \gamma\rangle}\downarrow & & & & \downarrow{\scriptstyle \gamma} \\ \Sigma(\mu\Sigma \times B(\mu\Sigma, \mu\Sigma)) & \xrightarrow{\varrho_{\mu\Sigma,\mu\Sigma}} & B(\mu\Sigma, \Sigma^\star(\mu\Sigma + \mu\Sigma)) & \xrightarrow{B(\mu\Sigma, \hat{\iota}\cdot\Sigma^\star\nabla)} & B(\mu\Sigma, \mu\Sigma) \end{array}$$

Here, we regard the initial algebra $(\mu\Sigma, \iota)$ as V-pointed as explained in Notation 3.

Simply Typed SKI Calculus. We illustrate the ideas behind higher-order abstract GSOS with an extended version of the simply typed SKI calculus [30], a typed combinatory logic which we call **xTCL**. It is expressively equivalent to the simply typed λ-calculus but does not use variables; hence it avoids the complexities associated to variable binding and substitution in the λ-calculus, which we treat in Section 4.2. The set Ty of *types* is inductively defined as

$$\mathsf{Ty} ::= \mathsf{unit} \mid \mathsf{Ty} \twoheadrightarrow \mathsf{Ty}. \tag{2}$$

The constructor $\twoheadrightarrow$ is right-associative, i.e. $\tau_1 \twoheadrightarrow \tau_2 \twoheadrightarrow \tau_3$ is parsed as $\tau_1 \twoheadrightarrow (\tau_2 \twoheadrightarrow \tau_3)$. The terms of **xTCL** are formed over the Ty-sorted signature Σ whose operation symbols are listed below, with $\tau, \tau_1, \tau_2, \tau_3$ ranging over all types in Ty:

$\mathsf{e}\colon \mathsf{unit}$	$\mathsf{app}_{\tau_1,\tau_2}\colon (\tau_1 \twoheadrightarrow \tau_2) \times \tau_1 \to \tau_2$
$S_{\tau_1,\tau_2,\tau_3}\colon (\tau_1 \twoheadrightarrow \tau_2 \twoheadrightarrow \tau_3) \twoheadrightarrow (\tau_1 \twoheadrightarrow \tau_2) \twoheadrightarrow \tau_1 \twoheadrightarrow \tau_3$	$K_{\tau_1,\tau_2}\colon \tau_1 \twoheadrightarrow \tau_2 \twoheadrightarrow \tau_1$
$S'_{\tau_1,\tau_2,\tau_3}\colon (\tau_1 \twoheadrightarrow \tau_2 \twoheadrightarrow \tau_3) \to ((\tau_1 \twoheadrightarrow \tau_2) \twoheadrightarrow \tau_1 \twoheadrightarrow \tau_3)$	$K'_{\tau_1,\tau_2}\colon \tau_1 \to (\tau_2 \twoheadrightarrow \tau_1)$
$S''_{\tau_1,\tau_2,\tau_3}\colon (\tau_1 \twoheadrightarrow \tau_2 \twoheadrightarrow \tau_3) \times (\tau_1 \twoheadrightarrow \tau_2) \to (\tau_1 \twoheadrightarrow \tau_3)$	$I_\tau\colon \tau \twoheadrightarrow \tau$

We let $\mathsf{Tr} = \mu\Sigma$ denote the Ty-sorted set of closed Σ-terms. Informally, app represents function application (we write $s\,t$ for $\mathsf{app}(s,t)$), and the constants I_τ, K_{τ_1,τ_2}, S_{τ_1,τ_2,τ_3} represent the λ-terms $\lambda t.\, t$, $\lambda t.\, \lambda s.\, t$ and $\lambda t.\, \lambda s.\, \lambda u.\, (s\,u)\,(t\,u)$, respectively. The operational semantics of **xTCL** involves three kinds of transitions: $\overset{\checkmark}{\longrightarrow}$, $\overset{t}{\longrightarrow}$ and $\longrightarrow$. It is presented in Figure 1; here, p, p', q, t range over terms in Tr of appropriate type. Intuitively, $s \overset{\checkmark}{\longrightarrow}$ identifies s as an explicitly irreducible term; $s \overset{t}{\longrightarrow} r$ states that s acts as a function mapping t to r; and $s \to t$ indicates that s reduces to t. Our use of labelled transitions

in higher-order operational semantics is inspired by work on bisimilarity in the λ-calculus [1,26]. The use of K', S' and S'' does not impact the behaviour of programs, except for possibly adding more unlabelled transitions. For example, the standard rule $Stse \to (te)(se)$ for the S-combinator is rendered as the chain of transitions $Stse \to S'(t)\,se \to S''(t,s)\,e \to (te)(se)$. The transition system for **xTCL** is deterministic: for every term s, either $s \not\to$, or there exists a unique t such that $s \to t$, or for each appropriately typed t there exists a unique s_t such that $s \xrightarrow{t} s_t$. Therefore, given

$$B_\tau(X,Y) = Y_\tau + D_\tau(X,Y), \tag{3}$$

$$D_{\mathsf{unit}}(X,Y) = 1 = \{*\} \qquad \text{and} \qquad D_{\tau_1 \to \tau_2}(X,Y) = Y_{\tau_2}^{X_{\tau_1}}, \tag{4}$$

the operational rules in Figure 1 determine a $\mathbf{Set}^{\mathsf{Ty}}$-morphism $\gamma\colon \mathsf{Tr} \to B(\mathsf{Tr},\mathsf{Tr})$:

$$\begin{aligned} \gamma_{\mathsf{unit}}(s) &= \mathsf{inr}(*) && \text{if } s \not\to \text{ where } s\colon \mathsf{unit},\\ \gamma_\tau(s) &= \mathsf{inl}(t) && \text{if } s \to t \text{ where } s,t\colon \tau,\\ \gamma_{\tau_1\to\tau_2}(s) &= \mathsf{inr}(\lambda t.\, s_t) && \text{if } s \xrightarrow{t} s_t \text{ for } s\colon \tau_1 \to \tau_2 \text{ and } t\colon \tau_1. \end{aligned} \tag{5}$$

Proposition 4. *The object assignments* (3) *and* (4) *extend to mixed-variance bifunctors*

$$B, D\colon (\mathbf{Set}^{\mathsf{Ty}})^{\mathsf{op}} \times \mathbf{Set}^{\mathsf{Ty}} \to \mathbf{Set}^{\mathsf{Ty}}. \tag{6}$$

The semantics of **xTCL** in Figure 1 corresponds to a (0-pointed) higher-order GSOS law of the syntax functor Σ over the behaviour bifunctor B, i.e. to a family of maps (1) dinatural in $X \in \mathbf{Set}^{\mathsf{Ty}}$ and natural in $Y \in \mathbf{Set}^{\mathsf{Ty}}$. The maps $\varrho_{X,Y}$ are cotuples defined by distinguishing cases on the constructors $\mathsf{e}, S, S', S'', K, K', I, \mathsf{app}$ of **xTCL**, and each component of ϱ is determined by the rules that apply to the corresponding constructor. We provide a few illustrative cases; see [25, p. 25], for a complete definition.

$$\varrho_{X,Y}\colon \Sigma(X \times B(X,Y)) \to B(X, \Sigma^\star(X+Y)) \tag{7}$$

$$\varrho_{X,Y}\,(S''_{\tau_1,\tau_2,\tau_3}((p,f),(q,g))) = \lambda t.\,(p\,t)\,(q\,t) \tag{8}$$

$$\varrho_{X,Y}\,((p,f)\,(q,g)) = f(q) \qquad \text{if } f\colon Y_{\tau_2}^{X_{\tau_1}} \tag{9}$$

$$\varrho_{X,Y}\,((p,f)\,(q,g)) = fq \qquad \text{if } f\colon Y_{\tau_1\to\tau_2} \tag{10}$$

The operational model $\gamma\colon \mathsf{Tr} \to B(\mathsf{Tr},\mathsf{Tr})$ of ϱ coincides with the coalgebra (5).

Remark 5. The rules for application in Figure 1 implement the call-by-name evaluation strategy. Other strategies can be captured by varying the rules and consequently the corresponding higher-order GSOS law. For the call-by-value strategy, one replaces the last rule with (11) and (12) below and modifies clause (9) in the definition of ϱ accordingly. One can also model the traditional view of combinatory logic as a rewrite system [30] where any redex can be reduced, no matter how deeply. This amounts to specifying a maximally nondeterministic strategy by adding the rule (13) below to Figure 1. Notably, this makes the operational model nondeterministic, and hence the corresponding higher-order GSOS law relies on the behaviour functor $\mathcal{P}B$ instead of the original B given by (3), where $\mathcal{P}$ is the powerset functor.

$$\frac{p \xrightarrow{t} p' \quad q \to q'}{p\,q \to p\,q'} \tag{11}$$

$$\frac{p \xrightarrow{q} p' \quad q \xrightarrow{t} q'}{p\,q \to p'} \tag{12}$$

$$\frac{q \to q'}{p\,q \to p\,q'} \tag{13}$$

3 Coalgebraic Logical Predicates

3.1 Predicate Lifting

Predicates and relations on coalgebras are often most conveniently modelled through *predicate* and *relation liftings* [39] of the underlying type functors. In the following we introduce a framework of predicate liftings for mixed-variance bifunctors, adapting existing notions of relation lifting [62], which enables reasoning about "higher-order" coalgebras, such as operational models of higher-order GSOS laws. The following global assumptions ensure that predicates and relations behave in an expected manner:

Assumptions 6. From now on, we fix $\mathcal{C}$ to be a complete, well-powered and extensive category in which, additionally, strong epimorphisms are stable under pullbacks.

The categories of Example 1 satisfy these assumptions. Since $\mathcal{C}$ is complete and well-powered, every morphism f admits a (strong epi, mono)-factorization $f = m \cdot e$ [11, Prop. 4.4.3]; we call m the *image* of f. The category $\mathbf{Pred}(\mathcal{C})$ of *predicates* over $\mathcal{C}$ has as objects all monics (predicates) $P \rightarrowtail X$ from $\mathcal{C}$, and as morphisms $(p\colon P \rightarrowtail X) \to (q\colon Q \rightarrowtail Y)$ all pairs $(f\colon X \to Y, f|_P\colon P \to Q)$ such that $q \cdot f|_P = f \cdot p$ (so $f|_P$ is uniquely determined by f). (Co)products in $\mathbf{Pred}(\mathcal{C})$ are lifted from $\mathcal{C}$. The *fiber* $\mathbf{Pred}_X(\mathcal{C})$ is the subcategory of all monics $P \rightarrowtail X$ for fixed X and morphisms $(\mathsf{id}_X, -)$. It is is preordered by $p \leq q$ if p factors through q; identifying p, q if $p \leq q$ and $q \leq p$, we regard $\mathbf{Pred}_X(\mathcal{C})$ as a poset. Since $\mathcal{C}$ is complete and well-powered, $\mathbf{Pred}_X(\mathcal{C})$ is a complete lattice; we write $\bigwedge$ for meets (i.e. pullbacks) and $\bigvee$ for joins. We will also write $f^\star[P]$ for the *inverse image* of a predicate $p\colon P \rightarrowtail X$ under $f\colon Y \to X$, i.e. the pullback of p along f. The *direct image* $f_\star[Q]$ of $q\colon Q \rightarrowtail Y$ under $f\colon Y \to X$ is the image of the composite $f \cdot p\colon Q \to X$. This yields an adjunction between $\mathbf{Pred}_X(\mathcal{C})$ and $\mathbf{Pred}_Y(\mathcal{C})$, i.e. $Q \leq f^\star[P]$ iff $f_\star[Q] \leq P$.

A *predicate lifting* of an endofunctor $\Sigma\colon \mathcal{C} \to \mathcal{C}$ is an endofunctor $\overline{\Sigma}\colon \mathbf{Pred}(\mathcal{C}) \to \mathbf{Pred}(\mathcal{C})$ making the left-hand diagram below commute; similarly, a *predicate lifting* of a mixed-variance bifunctor $B\colon \mathcal{C}^{\mathsf{op}} \times \mathcal{C} \to \mathcal{C}$ is a bifunctor $\overline{B}\colon \mathbf{Pred}(\mathcal{C})^{\mathsf{op}} \times \mathbf{Pred}(\mathcal{C}) \to \mathbf{Pred}(\mathcal{C})$ making the right-hand diagram below commute. Here $|-|$ is the forgetful functor sending $p\colon P \rightarrowtail X$ to X.

$$\begin{array}{ccc} \mathbf{Pred}(\mathcal{C}) & \xrightarrow{\overline{\Sigma}} & \mathbf{Pred}(\mathcal{C}) \\ {\scriptstyle |-|}\downarrow & & \downarrow{\scriptstyle |-|} \\ \mathcal{C} & \xrightarrow{\Sigma} & \mathcal{C} \end{array} \qquad \begin{array}{ccc} \mathbf{Pred}(\mathcal{C})^{\mathsf{op}} \times \mathbf{Pred}(\mathcal{C}) & \xrightarrow{\overline{B}} & \mathbf{Pred}(\mathcal{C}) \\ {\scriptstyle |-|^{\mathsf{op}} \times |-|}\downarrow & & \downarrow{\scriptstyle |-|} \\ \mathcal{C}^{\mathsf{op}} \times \mathcal{C} & \xrightarrow{B} & \mathcal{C} \end{array} \tag{14}$$

We denote by $\overline{\Sigma}$ both the action on predicates and on the corresponding objects in $\mathcal{C}$, i.e. $\overline{\Sigma}(p\colon P \rightarrowtail X)\colon \overline{\Sigma}P \rightarrowtail \Sigma X$.

Every endofunctor Σ on $\mathcal{C}$ admits a canonical predicate lifting $\overline{\Sigma}$ mapping $p\colon P \rightarrowtail X$ to the image $\overline{\Sigma}p\colon \overline{\Sigma}P \rightarrowtail \Sigma X$ of $\Sigma p\colon \Sigma P \to \Sigma X$ [36]. Note that $\overline{\Sigma}p = \Sigma p$ if Σ preserves monos. In the remainder we will only consider canonical liftings of endofunctors.

Proposition 7. *If Σ preserves strong epis, then $\overline{\Sigma}^\star = \overline{\Sigma^\star}$.*

The canonical predicate liftings for mixed-variance bifunctors are slightly more complex. Similarly to the case of relation liftings of such functors developed in recent work [62], their construction involves suitable pullbacks.

Proposition 8. *Every bifunctor $B\colon C^{\mathsf{op}} \times C \to C$ admits a canonical predicate lifting $\overline{B}\colon \mathbf{Pred}(C)^{\mathsf{op}} \times \mathbf{Pred}(C) \to \mathbf{Pred}(C)$ sending $(p\colon P \rightarrowtail X,\ q\colon Q \rightarrowtail Y)$ to the predicate $m_{P,Q}\colon \overline{B}(P,Q) \rightarrowtail B(X,Y)$, the image of the morphism $r_{P,Q}$ given by the pullback below:*

$$\begin{array}{ccccc} & & T_{P,Q} & \xrightarrow{s_{P,Q}} & B(P,Q) \\ & \swarrow^{e_{P,Q}} & \downarrow{\scriptstyle r_{P,Q}} & \lrcorner & \downarrow{\scriptstyle B(\mathsf{id},q)} \\ \overline{B}(P,Q) & & & & \\ & \searrow_{m_{P,Q}} & B(X,Y) & \xrightarrow{B(p,\mathsf{id})} & B(P,Y) \end{array} \tag{15}$$

If B preserves monos in the covariant argument, then $B(\mathsf{id}, q)$ is monic and, since monos are pullback-stable, $\overline{B}(P,Q)$ is simply the predicate $r_{P,Q}\colon T_{P,Q} \rightarrowtail B(X,Y)$.

Example 9. The bifunctors B and D of (3) and (4) have canonical predicate liftings

$$\overline{B}_\tau(P,Q) = Q_\tau + \overline{D}_\tau(P,Q) \quad \text{where} \tag{16}$$

$$\overline{D}_{\mathsf{unit}}(P,Q) = 1, \quad \overline{D}_{\tau_1 \to \tau_2}(P,Q) = \{f\colon X_{\tau_1} \to Y_{\tau_2} \mid \forall x \in P_{\tau_1}.\, f(x) \in Q_{\tau_2}\} \subseteq Y_{\tau_2}^{X_{\tau_1}}. \tag{17}$$

Predicate liftings allow us to generalize *coalgebraic invariants* [36, §6.2], viz. predicates on the state space of a coalgebra that are closed under the coalgebra structure in a suitable sense, from endofunctors to mixed-variance bifunctors:

Notation 10. For the remainder of the paper, we fix a mixed-variance bifunctor $B\colon C^{\mathsf{op}} \times C \to C$ and a predicate lifting $\overline{B}\colon \mathbf{Pred}(C)^{\mathsf{op}} \times \mathbf{Pred}(C) \to \mathbf{Pred}(C)$.

Definition 11 (Coalgebraic invariant)**.** Let $c\colon Y \to B(X,Y)$ be a $B(X,-)$-coalgebra. Given predicates $S \rightarrowtail X$, $P \rightarrowtail Y$, we say that P is an *S-relative ($\overline{B}$-)invariant (for c)* if $P \le c^\star[\overline{B}(S,P)]$, equivalently, $c_\star[P] \le \overline{B}(S,P)$. (Mention of $\overline{B}$ is usually omitted.)

Coalgebraic invariants will feature centrally in our notion of logical predicate.

3.2 Logical Predicates via Lifted Bifunctors

As a reasoning device, the method of logical predicates (which are unary logical relations) typically applies to the following scenario: One has an operational semantics on an inductively defined set $\mu\Sigma$ of Σ-terms and a target predicate $P \rightarrowtail \mu\Sigma$ to be proved, in the sense that one wants to show $P = \mu\Sigma$. Logical predicates come into play when a direct proof of $P = \mu\Sigma$ by structural induction is not possible. The classical example of such a predicate is *strong normalization* [23,59]. The idea is to strengthen P, obtaining a predicate featuring a certain "logical" structure that does allow for a proof by induction. We now develop this scenario in our abstract bifunctorial setting.

Definition 12 (Coalgebraic logical predicate)**.** Suppose that $c\colon X \to B(X,X)$ is a $B(X,-)$ coalgebra with state space X. A predicate $P \rightarrowtail X$ is *logical (for c)* if it is a P-relative $\overline{B}$-invariant (as per Def. 11), i.e. $P \le c^\star[\overline{B}(P,P)]$, equivalently, $c_\star[P] \le \overline{B}(P,P)$.

In applications, c is the operational model $\gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$ of a higher-order language, or some coalgebra derived from it. The self-referential nature of logical predicates (as relative to themselves) is meant to cater for the property that "inputs in P are mapped to outputs in P". The following example from **xTCL** illustrates this:

Example 13. For B given by (3) and its canonical lifting $\overline{B}$, a predicate $P \rightarrowtail \mathsf{Tr}$ is logical for the operational model $\gamma\colon \mathsf{Tr} \to B(\mathsf{Tr},\mathsf{Tr})$ from (5) if $\gamma_\star[P] \leq \overline{B}(P,P)$, that is,

$$(\gamma_{\mathsf{unit}})_\star[P_{\mathsf{unit}}] \leq P_{\mathsf{unit}} + 1,$$
$$\forall \tau_1,\tau_2.\,(\gamma_{\tau_1\to\tau_2})_\star[P_{\tau_1\to\tau_2}] \leq P_{\tau_1\to\tau_2} + \{f\colon \mathsf{Tr}_{\tau_1} \to \mathsf{Tr}_{\tau_2} \mid \forall s \in P_{\tau_1}.\, f(s) \in P_{\tau_2}\},$$

using the description of $\overline{B}$ from Example 9. More explicitly, this means that

- if $s \in P_\tau$ and $s \to t$ then $t \in P_\tau$;
- if $s \in P_{\tau_1\to\tau_2}$ and $s \xrightarrow{t} u$, then $t \in P_{\tau_1}$ implies $u \in P_{\tau_2}$.

As we can see in the second clause, function terms that satisfy P produce outputs that satisfy P on all inputs that satisfy P. This is the key property of any logical predicate.

Defining a suitable logical predicate (or relation) is the centerpiece of various sophisticated arguments in higher-order settings. One standard application of logical predicates are proofs of strong normalization, which we now illustrate in the case of **xTCL**. For the operational model $\gamma\colon \mathsf{Tr} \to B(\mathsf{Tr},\mathsf{Tr})$ and terms r,s,t of compatible type, put

- $s \Rightarrow t$ if $s = s_0 \to s_1 \to \cdots \to s_n = t$ for some $n \geq 0$ and terms $s_0,\ldots,s_n$;
- $s \overset{t}{\Rightarrow} r$ if $s \Rightarrow s'$ and $s' \xrightarrow{t} r$ for some (unique) s';
- $\Downarrow(s)$ if $s \Rightarrow s'$ and $\gamma(s') \in D(\mathsf{Tr},\mathsf{Tr})$ for some (unique) s'.

Coalgebraically, this associates a *weak operational model* $\widetilde{\gamma}\colon \mathsf{Tr} \to \mathcal{P}B(\mathsf{Tr},\mathsf{Tr})$ to γ, where $\widetilde{\gamma}(t) = \{t' \mid t \Rightarrow t'\} \cup \{\gamma(t') \mid t \Rightarrow t', \gamma(t') \in D(\mathsf{Tr},\mathsf{Tr})\}$.

Strong normalization of **xTCL** asserts that $\Downarrow = \mathsf{Tr}$: every term eventually reduces to a function or explicitly terminates. We now devise three different logical predicates on Tr, each of which provides a proof of that property. The idea is to refine the target predicate $\Downarrow \rightarrowtail \mathsf{Tr}$ to a logical predicate, for which showing that it is totally true will be facilitated by its invariance w.r.t. a corresponding coalgebra structure. Our first example will be based on the following notion of refinement:

Definition 14 (Locally maximal logical refinement)**.** Let $c\colon X \to B(X,X)$ be a coalgebra and let $P \rightarrowtail X$ be a predicate. A predicate $\Box P \rightarrowtail X$ is a *locally maximal logical refinement of* P if (i) $\Box P \leq P$, (ii) $\Box P$ is logical (i.e. a $\Box P$-relative $\overline{B}$-invariant), and (iii) for every predicate $Q \leq P$ that is a $\Box P$-relative $\overline{B}$-invariant, one has $Q \leq \Box P$.

Example 15. We define the predicate $\Box\Downarrow \rightarrowtail \mathsf{Tr}$, i.e. a family of subsets $\Box\Downarrow_\tau \subseteq \mathsf{Tr}_\tau$ ($\tau \in \mathsf{Ty}$), by induction on the structure of the type τ: we put $\Box\Downarrow_{\mathsf{unit}} = \Downarrow_{\mathsf{unit}}$, and we take $\Box\Downarrow_{\tau_1\to\tau_2}$ to be the greatest subset of $\mathsf{Tr}_{\tau_1\to\tau_2}$ satisfying

$$\Box\Downarrow_{\tau_1\to\tau_2}(t) \implies \Downarrow_{\tau_1\to\tau_2}(t) \wedge \begin{cases} \Box\Downarrow_{\tau_1\to\tau_2}(t') & \text{if} \quad t \to t' \\ \Box\Downarrow_{\tau_1}(s) \implies \Box\Downarrow_{\tau_2}(t') & \text{if} \quad t \xrightarrow{s} t' \end{cases}$$

From this definition it is not difficult to verify by induction on the type that

$$\Box\Downarrow \text{ is a locally maximal logical refinement of } \Downarrow. \tag{18}$$

Our goal is to show that $\Box\Downarrow$ is a subalgebra of $\mu\Sigma$, equivalently $\overline{\Sigma}(\Box\Downarrow) \leq \iota^\star[\Box\Downarrow]$, which then implies $\Box\Downarrow = \mathsf{Tr}$ and hence $\Downarrow = \mathsf{Tr}$ by structural induction. Taking the partition $\Sigma = \Xi + \Delta$ where Ξ is the part of the signature for application and Δ is the part of the signature for the remaining term constructors, we separately prove $\overline{\Xi}(\Box\Downarrow) \leq \iota^\star[\Box\Downarrow]$ and $\overline{\Delta}(\Box\Downarrow) \leq \iota^\star[\Box\Downarrow]$. It suffices to come up with $\Box\Downarrow$-relative invariants $A, C \subseteq \Downarrow$ such that $\overline{\Xi}(\Box\Downarrow) \leq \iota^\star[A]$ and $\overline{\Delta}(\Box\Downarrow) \leq \iota^\star[C]$. Then by (18) we can conclude $A, C \subseteq \Box\Downarrow$, so

$$\overline{\Xi}(\Box\Downarrow) \leq \iota^\star[A] \leq \iota^\star[\Box\Downarrow] \qquad \text{and} \qquad \overline{\Delta}(\Box\Downarrow) \leq \iota^\star[C] \leq \iota^\star[\Box\Downarrow].$$

Let us record for further reference what it means for $Q \rightarrowtail \mathsf{Tr}$ to be a $\Box\Downarrow$-relative invariant contained in $\Downarrow$. Given $t \in Q_\tau$, the following must hold:

(1) $\Downarrow_\tau t$, (2) if $t \to t'$ then $Q_\tau(t')$, (3) if $t\colon \tau_1 \to\!\!\!\!\triangleright\ \tau_2$ and $t \xrightarrow{s} t'$ and $\Box\Downarrow_{\tau_1} s$ then $Q_{\tau_2}(t')$.

We first put $A = \Box\Downarrow \vee (\iota \cdot \mathsf{inl})_\star[\overline{\Xi}\Box\Downarrow]$, and prove (1)–(3) for $Q = A$. So let $t \in A_\tau$; we distinguish cases on the disjunction defining A. If $\Box\Downarrow_\tau t$, then (1)–(3) follow easily by definition. Otherwise, we have $t = p\,q$ such that $\Box\Downarrow_{\tau_1 \to\!\!\!\!\triangleright\ \tau_2} p$ and $\Box\Downarrow_{\tau_1} q$.

(1) By definition, $\Box\Downarrow_{\tau_1 \to\!\!\!\!\triangleright\ \tau_2} p$ and $\Box\Downarrow_{\tau_1} q$ entail that $p \overset{q}{\Rightarrow} p'$ for a (unique) term p', and that $\Box\Downarrow_{\tau_2} p'$, hence $\Downarrow_{\tau_2} p'$. Since $p\,q \Rightarrow p'$, it follows that $\Downarrow_{\tau_2} p\,q$.

(2) We distinguish cases over the semantic rules for application:

(a) $p\,q \to p'\,q$ where $p \to p'$. Then $\Box\Downarrow_{\tau_1 \to\!\!\!\!\triangleright\ \tau_2} p'$, hence $A_{\tau_2}(p'\,q)$.

(b) $p\,q \to p'$ where $p \xrightarrow{q} p'$. Since $\Box\Downarrow_{\tau_1 \to\!\!\!\!\triangleright\ \tau_2} p$ and $\Box\Downarrow_{\tau_1} q$, we have $\Box\Downarrow_{\tau_2} p'$, so $A_{\tau_2}(p')$.

(3) t does not have labelled transitions, hence this case is void.

Next, we show that $C = \Box\Downarrow \vee (\iota \cdot \mathsf{inr})_\star[\overline{\Delta}(\Box\Downarrow)]$ is a $\Box\Downarrow$-relative invariant. We consider two representative cases; the remaining cases are handled similarly.

- Case $I_\tau\colon \tau \to\!\!\!\!\triangleright\ \tau$. Since I terminates immediately, property (1) holds by definition of $\Downarrow$ and (2) holds vacuously. For (3), if $I \xrightarrow{s} t'$ and $\Box\Downarrow_\tau s$, then $t' = s \in \Box\Downarrow_\tau \subseteq C_\tau$.
- Case $S''_{\tau_1,\tau_2,\tau_3}(t,s)\colon \tau_1 \to\!\!\!\!\triangleright\ \tau_3$ with $\Box\Downarrow_{\tau_1 \to\!\!\!\!\triangleright\ \tau_2 \to\!\!\!\!\triangleright\ \tau_3} t$ and $\Box\Downarrow_{\tau_1 \to\!\!\!\!\triangleright\ \tau_2} s$. Again, (1) holds because $S''(t,s)$ terminates immediately, and (2) holds vacuously. For (3), suppose that $\Box\Downarrow_{\tau_1} r$; we have to show $(t\,r)\,(s\,r) \in C_{\tau_3}$. This follows from the inequality $\overline{\Xi}(\Box\Downarrow) \leq \iota_\star[\Box\Downarrow]$ shown above, because $\Box\Downarrow_{\tau_2 \to\!\!\!\!\triangleright\ \tau_3}(t\,r)$, $\Box\Downarrow_{\tau_2}(s\,r)$ by definition of $\Box\Downarrow$.

Note that the definition of $\Box\Downarrow$ uses both induction (over the structure of types) and coinduction (by taking at every type the greatest predicate satisfying some property).

Example 16. We give an alternative logical predicate defined purely inductively. It resembles Plotkin's original concept of logical relation [55]. We define $\Downarrow \rightarrowtail \mathsf{Tr}$ by

$$\begin{aligned} \Downarrow_{\mathsf{unit}}(t) &\iff \Downarrow_{\mathsf{unit}}(t), \\ \Downarrow_{\tau_1 \to\!\!\!\!\triangleright\ \tau_2}(t) &\iff \Downarrow_{\tau_1 \to\!\!\!\!\triangleright\ \tau_2} t \wedge (\forall s\colon \tau_1.\, t \overset{s}{\Rightarrow} t' \wedge \Downarrow_{\tau_1}(s) \implies \Downarrow_{\tau_2}(t')). \end{aligned} \tag{19}$$

It is evidently logical for the restriction $\widetilde{\widetilde{\gamma}}\colon \mathsf{Tr} \to \mathcal{P}D(\mathsf{Tr},\mathsf{Tr})$ of the weak operational model to labelled transitions, given by $\widetilde{\widetilde{\gamma}}(t) := \{\gamma(t')\}$ if $t \Rightarrow t'$ and $\gamma(t') \in D(\mathsf{Tr},\mathsf{Tr})$, and $\widetilde{\widetilde{\gamma}}(t) := \emptyset$ otherwise. A proof of strong normalization using $\Downarrow$ is given in [25, App. A].

Example 17. A more popular (cf. [57,58]) and subtly different variant of $\Downarrow$ for proving strong normalization goes back to Tait [59]. We define $\mathrm{SN} \rightarrowtail \mathsf{Tr}$ by

$$\begin{aligned} \mathrm{SN}_{\mathsf{unit}}(t) &\iff \Downarrow_{\mathsf{unit}}(t) \\ \mathrm{SN}_{\tau_1 \to \tau_2}(t) &\iff \Downarrow_{\tau_1 \to \tau_2}(t) \wedge (\forall s\colon \tau_1.\, \mathrm{SN}_{\tau_1}(s) \implies \mathrm{SN}_{\tau_2}(t\, s)) \end{aligned} \tag{20}$$

Unlike $\Downarrow$, it is not immediate that SN is logical for $\widetilde{\widetilde{\gamma}}$ (see [25, App. A]). For a proof of strong normalization based on SN in the context of the λ-calculus, see [57, Sec. 2].

While all three logical predicates $\Box\Downarrow$, $\Downarrow$, SN are eligible for proving strong normalization, with proofs of similar length and complexity, the predicate $\Box\Downarrow$ arguably has the most generic flavour, as it depends neither on a system-specific notion of weak transition (which appears in the definition of $\Downarrow$) nor on the syntax of the language (such as the application operator appearing in the definition of SN). Thus, our abstract categorical approach to logical predicates will be based on a generalization of $\Box\Downarrow$.

3.3 Constructing Logical Predicates

Our abstract coalgebraic notion of logical predicate (Definition 12) is parametric in the bifunctor B and its lifting $\overline{B}$ and decoupled from any specific syntax. Next, we develop a systematic construction that promotes a predicate P to a logical predicate, specifically to a locally maximal refinement of P, generalizing $\Box\Downarrow$ in Example 15. The construction proceeds in two stages. First, we fix the contravariant argument of the lifted bifunctor $\overline{B}$ and construct a greatest coalgebraic invariant w.r.t. the resulting endofunctor [36, §6.3]:

Definition 18 (Relative henceforth)**.** Let $c\colon Y \to B(X,Y)$ and let $S \rightarrowtail X$ be a predicate. The *(S-)relative henceforth modality* sends $P \rightarrowtail Y$ to $\Box^{\overline{B},c}(S,P) \rightarrowtail Y$, which is the supremum in $\mathbf{Pred}_Y(C)$ of all S-relative invariants contained in P:

$$\Box^{\overline{B},c}(S,P) = \bigvee \{Q \leq P \mid Q \text{ is an } S\text{-relative } \overline{B}\text{-invariant for } c\}. \tag{21}$$

We will omit the superscripts $\overline{B}, c$ when they are irrelevant or clear from the context.

Proposition 19. *The predicate $\Box(S,P)$ is the greatest S-relative $\overline{B}$-invariant contained in P. Moreover, the map $(S,P) \mapsto \Box(S,P)$ is antitone in S and monotone in P.*

Proof. The first statement follows from the Knaster-Tarski theorem since $\Box(S,P)$ is the greatest fixed point $\Box(S,P) = \nu G.\, P \wedge c^\star[\overline{B}(S,G)]$ in the complete lattice $\mathbf{Pred}_Y(C)$. The second statement holds due to the mixed variance of the predicate lifting $\overline{B}$. □

The relative henceforth modality only yields relative invariants. To obtain a logical predicate, i.e. an invariant relative to itself, we move to the second stage of our construction, which is based on ultrametric semantics, see e.g. [9]. Let us briefly recall some terminology. A metric space $(X,\, d\colon X \times X \to \mathbb{R})$ is 1-*bounded* if $d(x,y) \leq 1$ for all x, y, an *ultrametric space* if $d(x,y) \leq \max\{d(x,z), d(z,y)\}$ for all x, y, z, and *complete* if every Cauchy sequence converges. A map $f\colon (X,d) \to (X',d')$ between metric spaces is *nonexpansive* if $d'(f(x), f(y)) \leq d(x,y)$ for all x, y, and *contractive* if there exists

$c \in [0,1)$, called a *contraction factor*, such that $d'(f(x), f(y)) \leq c \cdot d(x,y)$ for all x, y. A family of maps $(f_i \colon X \to X')_{i \in I}$ is *uniformly contractive* if there exists $c \in [0,1)$ such that each f_i is contractive with factor c. By Banach's fixed point theorem, every contractive endomap $f \colon X \to X$ on a non-empty complete metric space has a unique fixed point.

Definition 20. The category C is *predicate-contractive* if

(1) every $\mathbf{Pred}_X(C)$ carries the structure of a complete 1-bounded ultrametric space;
(2) for every $f \colon X \to Y$ in C, the map $f^\star[-] \colon \mathbf{Pred}_Y(C) \to \mathbf{Pred}_X(C)$ is non-expansive;
(3) for any two co-well-ordered families $(P^i \rightarrowtail X)_{i \in I}$ and $(Q^i \rightarrowtail X)_{i \in I}$ of predicates,

$$d(\textstyle\bigwedge_{i \in I} P^i, \bigwedge_{i \in I} Q^i) \leq \sup_{i \in I} d(P^i, Q^i).$$

Here $(P^i \rightarrowtail X)_{i \in I}$ is *co-well-ordered* if each nonempty subfamily has a greatest element.

Example 21. The category $C = \mathbf{Set}^{\mathsf{Ty}}$ is predicate-contractive when equipped with the ultrametric on $\mathbf{Pred}_X(C)$ given by $d(P,Q) = 2^{-n}$ for $P, Q \rightarrowtail X$, where $n = \inf\{\sharp\tau \mid P_\tau \neq Q_\tau\}$ and $\sharp\tau$ is the size of τ, defined by $\sharp\mathsf{unit} = 1$ and $\sharp(\tau_1 \to \tau_2) = \sharp\tau_1 + \sharp\tau_2$. By convention, $\inf \emptyset = \infty$ and $2^{-\infty} = 0$. To see predicate-contractivity, first note that a function $\mathcal{F} \colon \mathbf{Pred}_Y(C) \to \mathbf{Pred}_X(C)$ is non-expansive iff

$$\inf\{\sharp\tau \mid (\mathcal{F}P)_\tau \neq (\mathcal{F}Q)_\tau\} \geq \inf\{\sharp\tau \mid P_\tau \neq Q_\tau\} \qquad \text{for all } P, Q \rightarrowtail Y,$$

and contractive (necessarily with factor at most $1/2$) iff that inequality holds strictly.

This immediately implies clause (2) of Definition 20: inverse images in $\mathbf{Set}^{\mathsf{Ty}}$ are computed pointwise, and $f_\tau^\star[P_\tau] \neq f_\tau^\star[Q_\tau]$ implies $P_\tau \neq Q_\tau$ for $f \colon X \to Y$ and $P, Q \rightarrowtail Y$. Similarly, since intersections are computed pointwise, clause (3) amounts to

$$\inf\Big\{\sharp\tau \mid \bigcap\nolimits_{i \in I} P^i_\tau \neq \bigcap\nolimits_{i \in I} Q^i_\tau\Big\} \geq \inf\{\sharp\tau \mid \exists i \in I : P^i_\tau \neq Q^i_\tau\},$$

which is clearly true, for if $\bigcap_{i \in I} P^i_\tau \neq \bigcap_{i \in I} Q^i_\tau$ then $P^i_\tau \neq Q^i_\tau$ for some $i \in I$.

Definition 22 (Contractive lifting)**.** Suppose that C is predicate-contractive. The predicate lifting $\overline{B} \colon \mathbf{Pred}(C)^{\mathsf{op}} \times \mathbf{Pred}(C) \to \mathbf{Pred}(C)$ is *contractive* if for every $S \rightarrowtail X$ the map $\overline{B}(S, -)$ is non-expansive, and the family $(\overline{B}(-, P))_{P \rightarrowtail X}$ is uniformly contractive.

Proposition 23. *Let $\overline{B}$ be contractive and $c \colon X \to B(X,X)$. For every $S \rightarrowtail X$, the map $\Box^{\overline{B},c}(S, -)$ is non-expansive, and the family $(\Box^{\overline{B},c}(-, P))_{P \rightarrowtail X}$ is uniformly contractive.*

Contractive liftings allow us to augment every predicate P to a logical predicate:

Definition 24 (Henceforth)**.** Let $\overline{B}$ be contractive and $c \colon X \to B(X,X)$. For each predicate $P \rightarrowtail X$ we define $\Box^{\overline{B},c} P \rightarrowtail X$ (where we usually omit the superscripts) to be the unique fixed point of the contractive endomap

$$S \mapsto \Box^{\overline{B},c}(S, P) \quad \text{on} \quad \mathbf{Pred}_X(C). \tag{22}$$

Theorem 25. *The predicate $\Box P$ is the unique locally maximal logical refinement of P.*

Proof. By (22), $\Box P$ is the unique predicate satisfying $\Box P = \Box(\Box P, P)$. By (21), this equality says that $\Box P$ is the greatest $\Box P$-relative invariant contained in P, as needed. □

Example 26. Let B be the behaviour bifunctor on $\mathbf{Set}^{\mathsf{Ty}}$ given by (3). Its canonical lifting $\overline{B}$ (Example 9) is contractive because $\overline{B}_{\tau_1 \to \tau_2}(P, Q)$ depends only on P_{τ_1}, Q_{τ_2}, $Q_{\tau_1 \to \tau_2}$; in other words, $\overline{B}$ decreases the size of types in the contravariant argument and does not increase it in the covariant argument. Given a coalgebra $c\colon X \to B(X, X)$ and $P \rightarrowtail X$, the fixed point $\Box^{\overline{B},c} P$ is given by the Ty-indexed family of greatest fixed points

$$\Box P_{\mathsf{unit}} = \nu G.\, P_{\mathsf{unit}} \wedge {c_{\mathsf{unit}}}^{\star}[G + 1],$$
$$\Box P_{\tau_1 \to \tau_2} = \nu G.\, P_{\tau_1 \to \tau_2} \wedge {c_{\tau_1 \to \tau_2}}^{\star}[G + \{f\colon \mathsf{Tr}_{\tau_1} \to \mathsf{Tr}_{\tau_2} \mid \forall s \in \Box P_{\tau_1}.\, f(s) \in \Box P_{\tau_2}\}].$$

This follows from Theorem 25 since the above predicate is clearly a locally maximal refinement of P. By instantiating c to the operational model $\gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$ of **xTCL** and taking $P = \Downarrow$, we recover the definition of $\Box\Downarrow$ in Example 15.

Example 27. The logical predicate $\Downarrow\!\!\Downarrow \rightarrowtail \mathsf{Tr}$ of Example 16 is precisely $\Box\Downarrow$ for $\mathcal{P}D$ w.r.t. its canonical lifting and the coalgebra $\widetilde{\widetilde{\gamma}}\colon \mathsf{Tr} \to \mathcal{P}D(\mathsf{Tr}, \mathsf{Tr})$. More generally, for a coalgebra $c\colon X \to \mathcal{P}D(X, X)$, the predicate $\Box P$ is inductively defined as follows:

$$\Box P_{\mathsf{unit}} = P_{\mathsf{unit}},$$
$$\Box P_{\tau_1 \to \tau_2} = P_{\tau_1 \to \tau_2} \wedge {c_{\tau_1 \to \tau_2}}^{\star}[\{F \subseteq X_{\tau_2}^{X_{\tau_1}} \mid \forall f \in F.\, s \in \Box P_{\tau_1} \implies f(s) \in \Box P_{\tau_2}\}].$$

Remark 28. The construction of logical predicates for typed languages is enabled by the "type-decreasing" nature of the associated behaviour bifunctors. In untyped settings, e.g. for $B(X, Y) = Y + Y^X$ on **Set** modelling untyped combinatory logic [24], the canonical lifting $\overline{B}$ is not contractive, hence the fixed point $\Box P$ in general fails to exist.

Remark 29. The forgetful functor $|-|\colon \mathbf{Pred}(C) \to C$ forms a complete lattice fibration [35], equivalently a topological functor [2], and all notions and results of the present subsection extend to that level of generality. We leave the details for future work, as our reasoning techniques found in the upcoming sections are tailored to logical predicates.

We are now in a position to state precisely what a proof via logical predicates is in our framework. Given the operational model $\gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$ of a higher-order language, a predicate lifting $\overline{B}$, and a target predicate $P \rightarrowtail \mu\Sigma$, a *proof of P via logical predicates* is a proof that $\Box P$ forms a subalgebra of the initial algebra $\mu\Sigma$, which means

$$\overline{\Sigma}(\Box P) \leq \iota^{\star}[\Box P], \quad \text{equivalently} \quad \iota_{\star}[\overline{\Sigma}(\Box P)] \leq \Box P. \tag{23}$$

Then $\Box P = \mu\Sigma$ by structural induction, whence $P = \mu\Sigma$ because $\Box P \leq P$.

Up to this point, we have streamlined and formalized coalgebraic logical predicates as a certain abstract construction on predicates (Definition 24) and presented proofs by coalgebraic logical predicates as standard structural induction on said construction. This presentation is indeed that of an abstract method: the various parts of the problem setting, namely the syntax, the behaviour and its predicate lifting, as well as the operational semantics, are all parameters. In the next section, we exploit the parametric and generic nature of this method in two main ways. First, we present up-to techniques that simplify the proof goal (23) as much as possible. Second, we look to instantiate our method to problems on *classes of higher-order languages*, as opposed to reasoning about operational models of individual languages such as **xTCL** or the λ-calculus.

4 Logical Predicates and Higher-Order Abstract GSOS

As indicated before, substantial parts of the proof of strong normalization in Example 15 look generic. Specifically, the properties (2) and (3) established for $Q = A$ and $Q = C$ are independent of the choice of predicate $P = \Downarrow$ in $\Box P$. Moreover, these steps are either obvious or follow immediately from the operational rules of **xTCL**: the predicates A and C being invariants can be attributed to the fact that except for terms of the form $S''(-,-)$, all terms evolve either to a variable or to some flat term such as $p'\,q$. The core of the proof, which is tailored to the choice of P, lies in proving property (1).

As it turns out, for a class of higher-order GSOS laws that we call *relatively flat higher-order GSOS laws*, conditions (2) and (3) are automatic. This insight leads us to a powerful up-to technique that simplifies proofs via logical predicates.

4.1 Relatively Flat Higher-Order GSOS Laws

The following definition abstracts the restricted nature of the rules of **xTCL** to the level of higher-order GSOS laws. For simplicity, we confine ourselves to 0-pointed laws, however all the results of this subsection easily extend to the V-pointed case.

Definition 30. Let $\Sigma\colon C \to C$ be a syntax functor of the form $\Sigma = \coprod_{j\in J}\Sigma_j$, where $(J,\prec)$ is a non-empty well-founded strict partial order, and put $\Sigma_{\prec k} = \coprod_{j\prec k}\Sigma_j$. A *relatively flat (0-pointed) higher-order GSOS law* of Σ over B is a J-indexed family of morphisms

$$\varrho^j_{X,Y}\colon \Sigma_j(X \times B(X,Y)) \to B(X, \Sigma^\star_{\prec j}(X+Y) + \Sigma_j\Sigma^\star_{\prec j}(X+Y)) \tag{24}$$

dinatural in $X \in C$ and natural in $Y \in C$.

We put $e_{j,X} = [\mathsf{in}^\sharp_{\prec j}, \iota\cdot \mathsf{in}_j\cdot\Sigma_j(\mathsf{in}^\sharp_{\prec j})]\colon \Sigma^\star_{\prec j}X + \Sigma_j\Sigma^\star_{\prec j}X \to \Sigma^\star X$ where $\mathsf{in}_{\prec j}\colon \Sigma_{\prec j} \to \Sigma$ and $\mathsf{in}_j\colon \Sigma_j \to \Sigma$ are the coproduct injections, with free extensions $\mathsf{in}^\sharp_{\prec j}\colon \Sigma^\star_{\prec j} \to \Sigma^\star$ and $\mathsf{in}^\sharp_j\colon \Sigma^\star_j \to \Sigma^\star$. Every relatively flat higher-order GSOS law (24) determines an ordinary higher-order GSOS law of Σ over B with components

$$\varrho_{X,Y} = \coprod\!\!\coprod_{j\in J}\Sigma_j(X\times B(X,Y)) \xrightarrow{\coprod_{j\in J}\varrho^j_{X,Y}} \coprod\!\!\coprod_{j\in J} B(X,\Sigma^\star_{\prec j}(X+Y) + \Sigma_j\Sigma^\star_{\prec j}(X+Y)) \xrightarrow{[B(X,e_{j,X+Y})]_{j\in J}} B(X,\Sigma^\star(X+Y)).$$

When we interpret a higher-order GSOS law as a set of operational rules, relative flatness means that the operations of the language can be ranked in a way that every term $\mathsf{f}(-,\cdots,-)$ with f of rank j evolves into a term that uses only operations of strictly lower rank, except possibly its head symbol which may have the same rank j.

Example 31. **xTCL** is relatively flat: put $J = \{0 \prec 1\}$, let Σ_0 contain application, and let Σ_1 contain all other operation symbols. This is immediate from the rules in Figure 1.

Definition 32. Suppose that each Σ_j preserves strong epimorphisms. A *predicate lifting* of (24) is a relatively flat 0-pointed higher-order GSOS law $(\overline{\varrho}^j)_{j\in J}$ of $\overline{\Sigma} = \coprod_j \overline{\Sigma_j}$ over $\overline{B}$ where for every $P \rightarrowtail X$ and $Q \rightarrowtail Y$ the $\mathbf{Pred}(C)$-morphism $\overline{\varrho}^j_{P,Q}$ is carried by $\varrho^j_{X,Y}$.

Remark 33. (1) The condition on Σ_j ensures $\overline{\Sigma_j}^\star = \overline{\Sigma_j^\star}$ (Proposition 7), so that the first component of $\overline{\varrho}^j_{P,Q}$ has type $\Sigma_j(X \times B(X,Y)) \to B(X, \Sigma^\star_{<j}(X+Y) + \Sigma_j \Sigma^\star_{<j}(X+Y))$.
(2) Liftings are unique if they exist: since $\overline{\varrho}^j_{P,Q}$ is a $\mathbf{Pred}(C)$-morphism, it is determined by its first component $\varrho^j_{X,Y}$. Moreover, the (di)naturality of $\overline{\varrho}^j$ follows from that of ϱ^j.
(3) For the canonical lifting $\overline{B}$, a lifting $(\overline{\varrho}^j)_{j\in J}$ of $(\varrho^j)_{j\in J}$ always exists [25, App. D].

The following theorem establishes a sound up-to technique for logical predicates. It states that for operational models of relatively flat laws, the proof goal (23) can be established by checking a substantially relaxed property.

Theorem 34 (Induction up to $\Box$). *Let $\gamma\colon \mu\Sigma \to B(\mu\Sigma, \mu\Sigma)$ be the operational model of a relatively flat 0-pointed higher-order GSOS law that admits a predicate lifting. Then for every predicate $P \rightarrowtail \mu\Sigma$ and every locally maximal logical refinement $\Box^{\gamma,\overline{B}}P$,*

$$\overline{\Sigma}(\Box^{\gamma,\overline{B}}P) \le \iota^\star[P] \qquad \textit{implies} \quad \overline{\Sigma}(\Box^{\gamma,\overline{B}}P) \le \iota^\star[\Box^{\gamma,\overline{B}}P] \quad \textit{(hence } P = \mu\Sigma\textit{)}.$$

We stress that the theorem applies to any refinement $\Box^{\gamma,\overline{B}}P$ and does not assume a specific construction (e.g. that of Section 3.3). The up-to technique facilitates proofs via logical predicates quite dramatically. For illustration, we revisit strong normalization:

Example 35. We give an alternative proof of strong normalization of **xTCL** (cf. Example 15) via induction up to $\Box$. Hence it suffices to prove

$$\overline{\Sigma}(\Box\Downarrow) \le \iota^\star[\Downarrow],$$

which states that a term is terminating if all of its subterms are in the logical predicate $\Box\Downarrow$. This is clear for terms that are not applications, since they immediately terminate (cf. Figure 1). Now consider an application $p\,q$ such that $\Box_{\tau_1\to\tau_2}\Downarrow(p)$ and $\Box_{\tau_1}\Downarrow(q)$. Since $\Box\Downarrow$ is a logical predicate contained in $\Downarrow$, this entails that $p \overset{q}{\Rightarrow} p'$ for a (unique) term p', and that $\Box\Downarrow_{\tau_2} p'$, hence $\Downarrow_{\tau_2} p'$. Since $p\,q \Rightarrow p'$, it follows that $\Downarrow_{\tau_2} p\,q$.

Analogous reasoning shows that **xTCL** is strongly normalizing under the call-by-value and the maximally nondeterministic evaluation strategy (Remark 5). In the latter case, strong normalization means that every term must eventually terminate, independently of the order of evaluation.

The reader should compare the above compact argument to the laborious original proof given in Example 15. Our up-to technique can be seen to precisely isolate the non-trivial core of the proof, while providing its generic parts for free. For a further application – type safety of the simply typed λ-calculus – see Section 4.2.

4.2 λ-Laws

We proceed to explain how our theory of logical predicates applies to languages with variables and binders. We highlight the core ideas and technical challenges in the case of the λ-calculus, and briefly sketch their categorical generalization; a full exposition can

be found in [25, App. E]. Let **STLC** be the simply typed call-by-name λ-calculus with the set Ty of types given by (2) and operational rules

$$\frac{t \longrightarrow t'}{t\,s \to t'\,s} \qquad \frac{}{(\lambda x\colon \tau_1.\, t)\,s \to t[s/x]} \tag{25}$$

where s, t, t' range over λ-terms of appropriate type, and $[-/-]$ denotes capture-avoiding substitution. To model **STLC** in higher-order abstract GSOS, we follow ideas by Fiore [18]. Our base category C is the presheaf category $(\mathbf{Set}^{\mathbb{F}/\mathsf{Ty}})^{\mathsf{Ty}}$ where $\mathbb{F}$ denotes the category of finite cardinals and functions, and the set Ty is regarded as a discrete category. An object $\Gamma\colon n \to \mathsf{Ty}$ of $\mathbb{F}/\mathsf{Ty}$ is a *typed context*, associating to each variable $x \in n$ a type; we put $|\Gamma| := n$. A presheaf $X \in (\mathbf{Set}^{\mathbb{F}/\mathsf{Ty}})^{\mathsf{Ty}}$ associates to each context Γ and each type τ a set $X_\tau(\Gamma)$ whose elements we think of as terms of type τ in context Γ.

The syntax of **STLC** is captured by the functor $\Sigma\colon (\mathbf{Set}^{\mathbb{F}/\mathsf{Ty}})^{\mathsf{Ty}} \to (\mathbf{Set}^{\mathbb{F}/\mathsf{Ty}})^{\mathsf{Ty}}$ where

$$\begin{aligned} \Sigma_{\mathsf{unit}} X &= V_{\mathsf{unit}} + K_1 + \coprod\nolimits_{\tau\in\mathsf{Ty}} X_{\tau\to\mathsf{unit}} \times X_\tau, \\ \Sigma_{\tau_1\to\tau_2} X &= V_{\tau_1\to\tau_2} + \delta^{\tau_1}_{\tau_2} X + \coprod\nolimits_{\tau\in\mathsf{Ty}} X_{\tau\to\tau_1\to\tau_2} \times X_\tau. \end{aligned} \tag{26}$$

Here $K_1 \in \mathbf{Set}^{\mathbb{F}/\mathsf{Ty}}$ is the constant presheaf on 1, V is given by $V_\tau(\Gamma) = \{x \in |\Gamma| \mid \Gamma(x) = \tau\}$, and δ by $(\delta^{\tau_1}_{\tau_2} X)(\Gamma) = X_{\tau_2}(\Gamma + \check{\tau}_1)$ with $(-) + \check{\tau}_1$ denoting context extension by a variable of type τ_1. Informally, K_1, V and δ represent the constant $\mathsf{e}\colon \mathsf{unit}$, variables, and λ-abstraction, respectively. The initial algebra for Σ is the presheaf Λ of λ-terms, i.e. $\Lambda_\tau(\Gamma)$ is the set of λ-terms (modulo α-equivalence) of type τ in context Γ [18].

The behaviour bifunctor $B^\lambda\colon ((\mathbf{Set}^{\mathbb{F}/\mathsf{Ty}})^{\mathsf{Ty}})^{\mathsf{op}} \times (\mathbf{Set}^{\mathbb{F}/\mathsf{Ty}})^{\mathsf{Ty}} \to (\mathbf{Set}^{\mathbb{F}/\mathsf{Ty}})^{\mathsf{Ty}}$ for **STLC** has two separate components: it is given by a product

$$B^\lambda(X, Y) = \langle\!\langle X, Y \rangle\!\rangle \times B(X, Y) \tag{27}$$

where

$$\begin{aligned} \langle\!\langle X, Y \rangle\!\rangle_\tau(\Gamma) &= \mathbf{Set}^{\mathbb{F}/\mathsf{Ty}}\Big(\prod\nolimits_{x\in|\Gamma|} X_{\Gamma(x)}, Y_\tau\Big), \\ B(X, Y) &= (K_1 + Y + D(X, Y)), \\ D_{\mathsf{unit}}(X, Y) &= K_1 \qquad \text{and} \qquad D_{\tau_1\to\tau_2}(X, Y) = Y^{X_{\tau_1}}_{\tau_2}, \end{aligned}$$

and $Y^{X_{\tau_1}}_{\tau_2}$ is an exponential object in $\mathbf{Set}^{\mathbb{F}/\mathsf{Ty}}$. The bifunctor $\langle\!\langle -, - \rangle\!\rangle$ models an abstract substitution structure; for instance, every λ-term $t \in \Lambda_\tau(\Gamma)$ induces a natural transformation $\prod_{x\in|\Gamma|} \Lambda_{\Gamma(x)} \to \Lambda_\tau$ in $\langle\!\langle \Lambda, \Lambda \rangle\!\rangle_\tau(\Gamma)$ mapping a tuple $(t_1, \ldots, t_{|\Gamma|})$ to the term obtained by simultaneous substitution of the terms t_i for the variables of t. The summands of the bifunctor B abstract from the possible operational behaviour of λ-terms: a term may explicitly terminate, reduce, get stuck (e.g. if it is a variable), or act as a function.

The operational rules (25) of **STLC** can be encoded into a V-pointed higher-order GSOS law of Σ over B^λ, similar to the untyped λ-calculus treated in earlier work [24]. The operational model $\langle \phi, \gamma \rangle\colon \Lambda \to \langle\!\langle \Lambda, \Lambda \rangle\!\rangle \times B(\Lambda, \Lambda)$ is the coalgebra whose components ϕ, γ describe the substitution structure and the operational behaviour of λ-terms.

At this point, a key technical issue can be observed: the canonical predicate lifting $\overline{\langle\!\langle -, - \rangle\!\rangle}$ is not contractive. Indeed, given $P \rightarrowtail X$, $Q \rightarrowtail Y$, the predicate $\overline{\langle\!\langle P, Q \rangle\!\rangle}_\tau$ consists of all natural transformations $\prod_{x\in|\Gamma|} X_{\Gamma(x)} \to Y_\tau$ that restrict to $\prod_{x\in|\Gamma|} P_{\Gamma(x)} \to Q_\tau$, and

this expression depends on $P_{\Gamma(x)}$ where the type $\Gamma(x)$ may be of higher complexity than τ. In particular, we conclude that $\overline{B^\lambda}$ is not contractive. In contrast, the canonical lifting $\overline{B}$ is contractive and hence $\square^{\gamma,\overline{B}}P$ exists for every $P \rightarrowtail \Lambda$ (Definition 24). However, it is well-known that logical predicates do not do the trick for inductive proofs in the λ-calculus, see e.g. [57, p. 9] and [49, p. 150]; rather, one needs to prove the *open extension* of the logical predicate, which is the larger predicate

$$\boxdot^{\gamma,\overline{B}}P = \phi^\star[\overline{\langle\!\langle \square^{\gamma,\overline{B}}P, \square^{\gamma,\overline{B}}P \rangle\!\rangle}].$$

The standard proof method is then to show $\boxdot^{\gamma,\overline{B}}P = \Lambda$ directly by structural induction. However, this can be greatly simplified by the following up-to-principle, which works with the original predicate $\square^{\gamma,\overline{B}}P$ and forms a counterpart of Theorem 34 for the λ-calculus:

Theorem 36 (Induction up to $\boxdot$). *Let $P \rightarrowtail \Lambda$ be a predicate. Then*

$$\overline{\Sigma}(\square^{\gamma,\overline{B}}) \leq \iota^\star[P] \qquad \textit{implies} \qquad \overline{\Sigma}(\boxdot^{\gamma,\overline{B}}P) \leq \iota^\star[\boxdot^{\gamma,\overline{B}}P] \quad \textit{(hence } P = \Lambda\textit{)}.$$

Remark 37. Concretely, the theorem states that to prove $P = \Lambda$, it suffices to prove that (1) variables satisfy P, (2) the unit expression $\mathsf{e}\colon \mathsf{unit}$ satisfies P, (3) for all application terms $p\,q$ such that $\square_{\tau_1\to\tau_2}P(\Gamma)(p)$ and $\square_{\tau_1}P(\Gamma)(q)$, we have $P_{\tau_2}(\Gamma)(p\,q)$, and (4) for all λ-abstractions $\lambda x\colon \tau_1.\,t$ such that $t \in \square_{\tau_2}P(\Gamma, x)$, we have $P_{\tau_1\to\tau_2}(\Gamma)(\lambda x\colon \tau_1.\,t)$.

Example 38. We prove type safety for **STLC** via induction up to $\boxdot$. Thus consider the predicate $\mathrm{Safe} \rightarrowtail \Lambda$ that is constantly true on open terms and given by

$$t \in \mathrm{Safe}_\tau(\varnothing) \iff (\forall e.\, t \Rightarrow e \implies (e \text{ is not an application}) \vee \exists r.\, e \to r),$$

on closed terms. We only need to check the conditions (1)–(4) of Remark 37. Conditions (1), (2), (4) are clear since variables are open terms and the term $\mathsf{e}\colon \mathsf{unit}$ and λ-abstractions do not reduce. The only interesting clause is (3) for the empty context. Thus let $p\,q$ be a closed application term with $p \in \square\mathrm{Safe}_{\tau_1\to\tau_2}(\varnothing)$ and $q \in \square\mathrm{Safe}_{\tau_1}(\varnothing)$; we need to show $p\,q \in \mathrm{Safe}_{\tau_2}(\varnothing)$. We proceed by case distinction on $p\,q \Rightarrow e$:

(a) $p \Rightarrow p'$ and $e = p'\,q$. Then $p' \in \square\mathrm{Safe}_{\tau_1\to\tau_2}(\varnothing)$ by invariance, in particular p' is safe, so p' is either not an application or reduces. In the former case, p' is necessarily a λ-abstraction since it is closed and not of type unit. Thus, in both cases, e reduces.

(b) $p \Rightarrow \lambda x.p'$ and $p'[q/x] \Rightarrow e$. Since $\square\mathrm{Safe}$ is a logical predicate, from $p \in \square\mathrm{Safe}_{\tau_1\to\tau_2}(\varnothing)$ and $q \in \square_{\tau_1}\mathrm{Safe}(\varnothing)$ we can deduce $p'[q/x] \in \square_{\tau_2}\mathrm{Safe}(\varnothing)$, whence $e \in \square_{\tau_2}\mathrm{Safe}(\varnothing)$. In particular, e is safe, which implies that e is either not an application or reduces.

As an exercise, we invite the reader to prove strong normalization of **STLC** via induction up to $\boxdot$. The reader should compare these short and simple proofs with more traditional ones, see e.g. [57].

All the above results and observations for **STLC** can be generalized and developed at the level of general higher-order abstract GSOS laws. To this end, we first abstract the behaviour functor (27) to a functor of the form $B(X, Y) = (X \multimap Y) \times B'(X, Y)$, where

$(-) \multimap (-)$ is the internal hom-functor of a suitable closed monoidal structure on the base category C. In the case of **STLC**, this structure is given by Fiore's *substitution tensor* [18]. Second, we observe that the higher-order GSOS law of **STLC** is an instance of a special kind of law that we coin *relatively flat λ-laws*. The induction-up-to-$\boxdot$ technique of Theorem 36 then can be shown to hold for operational models of relatively flat λ-laws. More details can be found in [25, App. E].

5 Strong Normalization for Deterministic Systems, Abstractly

The high level of generality in which the theory of logical predicates is developed above enables reasoning uniformly about whole families of languages and behaviours. In this section, we narrow our focus to deterministic systems and establish a general strong normalization criterion, which can be checked in concrete instances by mere inspection of the operational rules corresponding to higher-order abstract GSOS laws.

Throughout this section, we fix a 0-pointed higher-order GSOS law ϱ of a signature endofunctor $\Sigma\colon C \to C$ over a behaviour bifunctor $B\colon C^{\mathsf{op}} \times C \to C$, where

$$B(X,Y) = Y + D(X,Y) \quad \text{for some} \quad D\colon C^{\mathsf{op}} \times C \to C.$$

For instance, the type functor (3) for **xTCL** is of that form. The operational model $\gamma\colon \mu\Sigma \to \mu\Sigma + D(\mu\Sigma,\mu\Sigma)$ has an n-step extension $\gamma^{(n)}\colon \mu\Sigma \to \mu\Sigma + D(\mu\Sigma,\mu\Sigma)$, for each $n \in \mathbb{N}$, where $\gamma^{(0)}$ is the left coproduct injection and $\gamma^{(n+1)}$ is the composite

$$\mu\Sigma \xrightarrow{\gamma} \mu\Sigma + D(\mu\Sigma,\mu\Sigma) \xrightarrow{\gamma^{(n)}+\mathsf{id}} \mu\Sigma + D(\mu\Sigma,\mu\Sigma) + D(\mu\Sigma,\mu\Sigma) \xrightarrow{\mathsf{id}+\nabla} \mu\Sigma + D(\mu\Sigma,\mu\Sigma).$$

We regard $D(\mu\Sigma,\mu\Sigma)$ as a predicate on $B(\mu\Sigma,\mu\Sigma)$ via the right coproduct injection, which is monic by extensivity of C, and define the following predicates on $\mu\Sigma$:

$$\Downarrow_n = (\gamma^{(n)})^\star[D(\mu\Sigma,\mu\Sigma)] \qquad \text{and} \qquad \Downarrow = \bigvee_n \Downarrow_n .$$

In **xTCL**, these are the predicates of strong normalization or strong normalization after at most n steps, resp. Accordingly, we define strong normalization abstractly as follows:

Definition 39. The higher-order GSOS law ϱ is *strongly normalizing* if $\Downarrow = \mu\Sigma$.

We next identify two natural conditions on the law ϱ that together ensure strong normalization. The first roughly asserts that for a term $t = \mathsf{f}(x_1,\dots,x_n)$ whose variables x_i are non-progressing, the term t is either non-progressing or it progresses to a variable.

Definition 40. The higher-order GSOS law ϱ is *simple* if its components $\varrho_{X,Y}$ restrict to morphisms $\varrho^0_{X,Y}$ as in the diagram below, where η is the unit of the free monad $\Sigma^\star$:

$$\begin{array}{ccc} \Sigma(X \times D(X,Y)) & \overset{\varrho^0_{X,Y}}{\dashrightarrow} & X + Y + D(X, \Sigma^\star(X+Y)) \\ {\scriptstyle \Sigma(\mathsf{id}\times\mathsf{inr})}\downarrow & & \downarrow{\scriptstyle \eta_{X+Y}+\mathsf{id}} \\ \Sigma(X \times (Y + D(X,Y))) & \xrightarrow{\varrho_{X,Y}} & \Sigma^\star(X+Y) + D(X, \Sigma^\star(X+Y)) \end{array}$$

The second condition asserts that the rules represented by the higher-order GSOS law remain sound when strong transitions are replaced by weak ones. In the following, the *graph* of a morphism $f\colon A \to B$ is the image $\mathsf{gra}(f) \rightarrowtail A \times B$ of $\langle \mathsf{id}, f\rangle\colon A \to A \times B$.

Definition 41. The higher-order GSOS law ϱ *respects weak transitions* if for every $n \in \mathbb{N}$, the graph of the composite below is contained in $\bigvee_k \mathsf{gra}(\gamma^{(k)} \cdot \iota)$.

$$\Sigma(\mu\Sigma) \xrightarrow{\Sigma\langle \mathsf{id}, \gamma^{(n)}\rangle} \Sigma(\mu\Sigma \times B(\mu\Sigma, \mu\Sigma)) \xrightarrow{\varrho_{\mu\Sigma,\mu\Sigma}} B(\mu\Sigma, \Sigma^\star(\mu\Sigma + \mu\Sigma)) \xrightarrow{B(\mathsf{id}, \hat{\iota}\cdot\Sigma^\star\nabla)} B(\mu\Sigma, \mu\Sigma)$$

Note that the higher-order GSOS law for **xTCL** is simple and respects weak transitions. Thus, strong normalization of **xTCL** is an instance of the following strong normalization theorem for higher-order abstract GSOS. Concerning its conditions, an *ω-directed union* is a colimit of an ω-chain $X_0 \rightarrowtail X_1 \rightarrowtail X_2 \rightarrowtail \cdots$ of monics. We say that monos in $\mathcal{C}$ are *ω-smooth* if any such colimit has monic injections, and moreover for every compatible cocone of monos, the mediating morphism is monic. This property holds in every locally finitely presentable category [3, Prop. 1.62], e.g. sets, posets, or presheaves.

Theorem 42 (Strong normalization). *Suppose that the following conditions hold:*

(1) *On top of Assumptions 6, $\mathcal{C}$ is countably extensive, and monos are ω-smooth.*

(2) *Σ preserves ω-directed unions, and $\mathcal{D}$ preserves monos in the second component.*

(3) *ϱ is relatively flat, simple, and respects weak transitions.*

(4) *$\Downarrow$ has a locally maximal logical refinement w.r.t. γ and the canonical lifting $\overline{B}$.*

Then the higher-order GSOS law ϱ is strongly normalizing.

Recall that condition (4) holds if $\overline{B}$ is contractive (Theorem 25). The proof uses the induction-up-to-$\Box$ technique and a careful categorical abstraction of Example 35.

6 Conclusion and Future Work

Our work presents the initial steps towards a unifying, efficient theory of logical relations for higher-order languages based on higher-order abstract GSOS. This theory can be broadened in various directions. One obvious direction would be to extend our theory from predicates to relations. Binary logical relations are often utilized as sound (and sometimes complete) relations w.r.t. *contextual equivalence*. Additional generalizations are suggested by the large amount of existing work on logical relations. One important direction is to generalize the type system to cover, e.g., recursive types, parametric polymorphism, or dependent types. Supporting recursive types will presumably require an adaptation of the method of step-indexing [17] to our abstract setting. Another point of interest is to apply and extend our framework to effectful (e.g. probabilistic) settings [40,54], including e.g. an effectful version of the criterion of Section 5.

As indicated in Remark 29, large parts of our development in Section 3 can be reformulated in fibrational terms. This has the potential merit of enabling abstract reasoning about higher-order programs in metric and differential settings as done in previous work on fine-grain call-by-value [13,14]. In future work, we aim to develop such a generalization, and to explore the connection between our weak transition semantics and the general evaluation semantics used in *op. cit.*

References

1. Abramsky, S.: The lazy λ-calculus. In: Research topics in Functional Programming, pp. 65–117. Addison Wesley (1990)
2. Adámek, J., Herrlich, H., Strecker, G.E.: Abstract and Concrete Categories. Wiley (1990), republished in: Reprints in Theory and Applications of Categories 17 (2006), pp. 1-507, http://www.tac.mta.ca/tac/reprints/articles/17/tr17abs.html
3. Adámek, J., Rosický, J.: Locally Presentable and Accessible Categories. London Mathematical Society Lecture Note Series, Cambridge University Press (1994). https://doi.org/10.1017/CBO9780511600579
4. Aguirre, A., Birkedal, L.: Step-indexed logical relations for countable nondeterminism and probabilistic choice. In: 50th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2023). Proc. ACM Program. Lang., vol. 7. ACM (2023). https://doi.org/10.1145/3571195
5. Ahmed, A.: Step-indexed syntactic logical relations for recursive and quantified types. In: 15th European Symposium on Programming (ESOP 2006). LNCS, vol. 3924, pp. 69–83. Springer (2006). https://doi.org/10.1007/11693024_6
6. Altenkirch, T., Kaposi, A.: Normalisation by evaluation for dependent types. In: 1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016). LIPIcs, vol. 52, pp. 6:1–6:16. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik (2016). https://doi.org/10.4230/LIPIcs.FSCD.2016.6
7. Appel, A.W., McAllester, D.A.: An indexed model of recursive types for foundational proof-carrying code. ACM Trans. Program. Lang. Syst. **23**(5), 657–683 (2001). https://doi.org/10.1145/504709.504712
8. Benton, N., Hur, C.K.: Biorthogonality, step-indexing and compiler correctness. In: 14th ACM SIGPLAN International Conference on Functional Programming (ICFP 2009). p. 97–108. ACM (2009). https://doi.org/10.1145/1596550.1596567
9. Birkedal, L., Støvring, K., Thamsborg, J.: The category-theoretic solution of recursive metric-space equations. Theoretical Computer Science **411**(47), 4102–4122 (2010). https://doi.org/10.1016/j.tcs.2010.07.010
10. Bizjak, A., Birkedal, L.: Step-indexed logical relations for probability. In: Pitts, A. (ed.) 18th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2015). LNCS, vol. 9034, pp. 279–294. Springer (2015). https://doi.org/10.1007/978-3-662-46678-0_18
11. Borceux, F.: Handbook of Categorical Algebra: Volume 1: Basic Category Theory, Encyclopedia of Mathematics and Its Applications, vol. 1. Cambridge University Press (1994). https://doi.org/10.1017/CBO9780511525858
12. Carboni, A., Lack, S., Walters, R.F.C.: Introduction to extensive and distributive categories. Journal of Pure and Applied Algebra **84**(2), 145–158 (Feb 1993). https://doi.org/10.1016/0022-4049(93)90035-R
13. Dagnino, F., Gavazzo, F.: A fibrational tale of operational logical relations. In: 7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022). LIPIcs, vol. 228, pp. 3:1–3:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.FSCD.2022.3
14. Dagnino, F., Gavazzo, F.: A Fibrational Tale of Operational Logical Relations: Pure, Effectful and Differential. CoRR (2023). https://doi.org/10.48550/arXiv.2303.03271
15. Dal Lago, U., Gavazzo, F.: Differential logical relations, part ii increments and derivatives. Theor. Comput. Sci. **895**(C), 34–47 (2021). https://doi.org/10.1016/j.tcs.2021.09.027

16. Dal Lago, U., Gavazzo, F., Levy, P.B.: Effectful applicative bisimilarity: Monads, relators, and Howe's method. In: 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2017). pp. 1–12. IEEE Computer Society (2017). https://doi.org/10.1109/LICS.2017.8005117
17. Dreyer, D., Ahmed, A., Birkedal, L.: Logical step-indexed logical relations. In: 24th Annual IEEE Symposium on Logic In Computer Science (LICS 2009). pp. 71–80. IEEE Computer Society (2009). https://doi.org/10.1109/LICS.2009.34
18. Fiore, M.: Semantic analysis of normalisation by evaluation for typed lambda calculus. Math. Struct. Comput. Sci. **32**(8), 1028–1065 (2022). https://doi.org/10.1017/S0960129522000263
19. Fiore, M.P., Plotkin, G.D., Turi, D.: Abstract syntax and variable binding. In: 14th Annual IEEE Symposium on Logic in Computer Science (LICS 1999). pp. 193–202. IEEE Computer Society (1999). https://doi.org/10.1109/LICS.1999.782615
20. Fiore, M.P., Turi, D.: Semantics of name and value passing. In: 16th Annual IEEE Symposium on Logic in Computer Science (LICS 2001). pp. 93–104. IEEE Computer Society (2001). https://doi.org/10.1109/LICS.2001.932486
21. Georges, A.L., Guéneau, A., Van Strydonck, T., Timany, A., Trieu, A., Devriese, D., Birkedal, L.: Cerise: Program verification on a capability machine in the presence of untrusted code. J. ACM (2023). https://doi.org/10.1145/3623510
22. Giarrusso, P.G., Stefanesco, L., Timany, A., Birkedal, L., Krebbers, R.: Scala step-by-step: Soundness for dot with step-indexed logical relations in iris. In: 25th ACM SIGPLAN International Conference on Functional Programming (ICFP 2020). Proc. ACM Program. Lang., vol. 4. ACM (2020). https://doi.org/10.1145/3408996
23. Girard, J.Y., Taylor, P., Lafont, Y.: Proofs and types, vol. 7. Cambridge University Press (1989)
24. Goncharov, S., Milius, S., Schröder, L., Tsampas, S., Urbat, H.: Towards a higher-order mathematical operational semantics. In: 50th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2023). Proc. ACM Program. Lang., vol. 7. ACM (2023). https://doi.org/10.1145/3571215
25. Goncharov, S., Santamaria, A., Schröder, L., Tsampas, S., Urbat, H.: Logical predicates in higher-order mathematical operational semantics (2024), https://arxiv.org/abs/2401.05872
26. Gordon, A.D.: Bisimilarity as a theory of functional programming. Theor. Comput. Sci. **228**(1-2), 5–47 (1999). https://doi.org/10.1016/S0304-3975(98)00353-3
27. Goubault-Larrecq, J., Lasota, S., Nowak, D.: Logical relations for monadic types. Math. Struct. Comput. Sci. **18**(6), 1169–1217 (2008). https://doi.org/10.1017/S0960129508007172
28. Hermida, C., Reddy, U.S., Robinson, E.P.: Logical relations and parametricity - A Reynolds programme for category theory and programming languages. Electron. Notes Theor. Comput. Sci. **303**, 149–180 (2014). https://doi.org/10.1016/j.entcs.2014.02.008
29. Hermida, C.A.: Fibrations, logical predicates and indeterminates. Ph.D. thesis, University of Edinburgh (1993), https://era.ed.ac.uk/handle/1842/14057
30. Hindley, J.R., Seldin, J.P.: Lambda-Calculus and Combinators: An Introduction. Cambridge University Press, 2 edn. (2008). https://doi.org/10.1017/CBO9780511809835
31. Howe, D.J.: Equality in lazy computation systems. In: 4th Annual Symposium on Logic in Computer Science (LICS 1989). pp. 198–203. IEEE Computer Society (1989). https://doi.org/10.1109/LICS.1989.39174
32. Howe, D.J.: Proving congruence of bisimulation in functional programming languages. Inf. Comput. **124**(2), 103–112 (1996). https://doi.org/10.1006/inco.1996.0008

33. Hur, C.K., Dreyer, D.: A Kripke Logical Relation between ML and Assembly. In: 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2023). p. 133–146. ACM (2011). https://doi.org/10.1145/1926385.1926402
34. Hur, C.K., Dreyer, D., Neis, G., Vafeiadis, V.: The Marriage of Bisimulations and Kripke Logical Relations. In: 39th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2012). SIGPLAN Not., vol. 47, p. 59–72. ACM (2012). https://doi.org/10.1145/2103621.2103666
35. Jacobs, B.: Categorical Logic and Type Theory. No. 141 in Studies in Logic and the Foundations of Mathematics, North Holland (1999)
36. Jacobs, B.: Introduction to Coalgebra: Towards Mathematics of States and Observation, Cambridge Tracts in Theoretical Computer Science, vol. 59. Cambridge University Press (2016). https://doi.org/10.1017/CBO9781316823187
37. Johann, P., Simpson, A., Voigtländer, J.: A generic operational metatheory for algebraic effects. In: 25th Annual IEEE Symposium on Logic in Computer Science (LICS 2010). pp. 209–218. IEEE Computer Society (2010). https://doi.org/10.1109/LICS.2010.29
38. Katsumata, S.: A generalisation of pre-logical predicates and its applications. Ph.D. thesis, University of Edinburgh (2005), http://hdl.handle.net/1842/850
39. Kurz, A., Velebil, J.: Relation lifting, a survey. Journal of Logical and Algebraic Methods in Programming **85**(4), 475–499 (2016). https://doi.org/10.1016/j.jlamp.2015.08.002
40. Lago, U.D., Gavazzo, F.: Effectful program distancing. In: 49th Annual ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2022). Proc. ACM Program. Lang., vol. 6, pp. 1–30 (2022). https://doi.org/10.1145/3498680
41. Lago, U.D., Gavazzo, F., Yoshimizu, A.: Differential Logical Relations, Part I: The Simply-Typed Case. In: 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019). LIPIcs, vol. 132, pp. 111:1–111:14. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik (2019). https://doi.org/10.4230/LIPIcs.ICALP.2019.111
42. Levy, P., Power, J., Thielecke, H.: Modelling environments in call-by-value programming languages. Inf. Comput. **185**(2), 182–210 (2003)
43. Mac Lane, S.: Categories for the Working Mathematician, Graduate Texts in Mathematics, vol. 5. Springer, 2 edn. (1978), http://link.springer.com/10.1007/978-1-4757-4721-8
44. Milner, R.: A theory of type polymorphism in programming. Journal of Computer and System Sciences **17**(3), 348–375 (1978). https://doi.org/10.1016/0022-0000(78)90014-4
45. New, M.S., Bowman, W.J., Ahmed, A.: Fully abstract compilation via universal embedding. In: 21st ACM SIGPLAN International Conference on Functional Programming (ICFP 2016). pp. 103–116. ACM (2016). https://doi.org/10.1145/2951913.2951941
46. O'Hearn, P.W., Riecke, J.G.: Kripke logical relations and PCF. Inf. Comput. **120**(1), 107–116 (1995). https://doi.org/10.1006/inco.1995.1103
47. Ong, C.H.L.: The Lazy Lambda Calculus: An Investigation into the Foundations of Functional Programming. Ph.D. thesis, Imperial College London (1988), http://hdl.handle.net/10044/1/47211
48. Patrignani, M., Martin, E.M., Devriese, D.: On the semantic expressiveness of recursive types. In: 48th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2021). Proc. ACM Program. Lang., vol. 5. ACM (2021). https://doi.org/10.1145/3434302
49. Pierce, B.C.: Types and programming languages. MIT Press (2002)
50. Pitts, A.M.: Reasoning about local variables with operationally-based logical relations. In: 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996). pp. 152–163. IEEE Computer Society (1996). https://doi.org/10.1109/LICS.1996.561314
51. Pitts, A.M.: Relational properties of domains. Information and Computation **127**(2), 66–90 (1996). https://doi.org/10.1006/inco.1996.0052

52. Pitts, A.M.: Parametric polymorphism and operational equivalence. Mathematical Structures in Computer Science **10**(3), 321–359 (2000). https://doi.org/10.1017/S0960129500003066
53. Pitts, A.M., Stark, I.D.B.: Observable properties of higher order functions that dynamically create local names, or: What's new? In: 8th International Symposium on Mathematical Foundations of Computer Science (MFCS 1993). LNCS, vol. 711, pp. 122–141. Springer (1993). https://doi.org/10.1007/3-540-57182-5_8
54. Pitts, A.M., Stark, I.D.B.: Operational reasoning for functions with local state. In: Gordon, A.D., Pitts, A.M. (eds.) Higher Order Operational Techniques in Semantics, pp. 227–274. Cambridge University Press, New York, NY, USA (1998)
55. Plotkin, G.D.: Lambda-definability and logical relations. Tech. rep., University of Edinburgh (1973)
56. Sieber, K.: Reasoning about sequential functions via logical relations. In: Fourman, M.P., Johnstone, P.T., Pitts, A.M. (eds.) Applications of Categories in Computer Science: Proceedings of the London Mathematical Society Symposium, Durham 1991. p. 258–269. London Mathematical Society Lecture Note Series, Cambridge University Press (1992). https://doi.org/10.1017/CBO9780511525902.015
57. Skorstengaard, L.: An Introduction to Logical Relations (2019). https://doi.org/10.48550/arXiv.1907.11133
58. Statman, R.: Logical relations and the typed lambda-calculus. Information and Control **65**(2), 85–97 (1985). https://doi.org/10.1016/S0019-9958(85)80001-2
59. Tait, W.W.: Intensional interpretations of functionals of finite type I. J. Symb. Log. **32**(2), 198–212 (1967). https://doi.org/10.2307/2271658
60. Timany, A., Stefanesco, L., Krogh-Jespersen, M., Birkedal, L.: A logical relation for monadic encapsulation of state: Proving contextual equivalences in the presence of runst. In: 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2017). Proc. ACM Program. Lang., vol. 2. ACM (2017). https://doi.org/10.1145/3158152
61. Turi, D., Plotkin, G.D.: Towards a mathematical operational semantics. In: 12th Annual IEEE Symposium on Logic in Computer Science (LICS 1997). pp. 280–291 (1997). https://doi.org/10.1109/LICS.1997.614955
62. Urbat, H., Tsampas, S., Goncharov, S., Milius, S., Schröder, L.: Weak similarity in higher-order mathematical operational semantics. In: 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2023). IEEE Computer Society Press (2023). https://doi.org/10.1109/LICS56636.2023.10175706
63. Wand, M., Culpepper, R., Giannakopoulos, T., Cobb, A.: Contextual equivalence for a probabilistic language with continuous random variables and recursion. In: 23rd ACM SIGPLAN International Conference on Functional Programming (ICFP 2018). Proc. ACM Program. Lang., vol. 2. ACM (2018). https://doi.org/10.1145/3236782