Step-indexed models of call-by-name: a tutorial example

Aleš Bizjak[1] and Lars Birkedal[1]

[1]Aarhus University {abizjak,birkedal}@cs.au.dk

June 19, 2014

Abstract

In this tutorial paper we show how to construct a step-indexed logical relation for a call-by-name programming language with recursive types and show that it is complete with respect to contextual equivalence. We then show how the same constructions can be used to define a step-indexed model, in the standard categorical sense, of the language.

We hope that this will make step-indexed techniques more readily available for researchers interested in call-by-name or call-by-need based languages (such as Haskell) and make it clear that step-indexed models can indeed be seen as (operationally-based) models in the technical sense (and not only as a reasoning technique for reasoning about contextual equivalence).

1 Introduction

In recent years, step-indexed logical relations have proved to be useful for reasoning about contextual equivalence for programming languages with "advanced" features, such as recursive types, impredicative polymorphism, and general references. The techniques have been almost exclusively developed for call-by-value programming languages, e.g. [3, 1, 2, 4, 8]; we are only aware of one simple application to a call-by-name programming language [10] where the model is unary and used to prove type-soundness.

In this tutorial paper we show how to construct a step-indexed model of a call-by-name programming language with recursive types. Moreover, we also show how to define a step-indexed model, in the standard categorical sense, of the language. We hope that this (1) will make step-indexed techniques more readily available for researchers interested in call-by-name or call-by-need based languages (such as Haskell); and (2) will make it clear that step-indexed models can indeed be seen as (operationally-based) models in the technical sense (and not only as a reasoning technique, for reasoning about contextual equivalence). We hope that this tutorial may complement the recent tutorial paper by

Pitts [12], who explains how to define a step-indexed model for a call-by-value language (but does not consider categorical models).

We now explain the contents of the paper in a bit more detail. We call the programming language PCF_μ since it is a variant of PCF extended with recursive types. We use step-indexing both to model fixed points of terms, but also to interpret recursive types. Building on recent ideas of Hoshino [9] for call-by-value, we do not only develop an interpretation of the types of PCF_μ using a logical relation, but we actually show how to use step-indexing to define a category with the requisite properties to model PCF_μ. Hoshino shows that his model for a call-by-value language is adequate; here we prove not only adequacy, but also that the model is fully abstract. (We discuss why that is in Section 5.)

Earlier work [7] has combined step-indexed logical relations with biorthogonality in order to (1) get compleness wrt. contextual equivalence, and (2) to model control features, such as call-cc. (See also the tutorial by Pitts [12].) Here we also use biorthogonality, for completeness (full abstraction), and also because it makes it *simpler* to define the model, in particular to ensure that we only observe computation to a value of ground type.

In Section 2 we define the PCF_μ language and state some simple properties of the operational semantics. In Section 3 we define a step-indexed logical relations interpretation of the types of PCF_μ and show that it provides a sound and complete method for reasoning about contextual equivalence of programs in PCF_μ.

In Section 4 we define the category $\mathcal{C}$ and prove that it yields (a fully abstract) model. The objects of $\mathcal{C}$ are indexed collections of relations on closed terms of PCF_μ. Morphisms from X to Y are certain equivalence classes of open terms that, roughly speaking, for any step-index, take related *computations* in X to related *computations* in Y. The notion of *computation* is defined using biorthogonality. The equivalence relation is inspired by the one in [9] and is defined by taking the transitive closure of the quasi-reflexive symmetric interior of the step-indexed relation defining when a term contextually approximates another one.

We prove that the category $\mathcal{C}$ is cartesian closed and has fixed point of all endomorphisms. Hence it soundly interprets PCF [6]. We prove that it also interprets recursive types. The adequacy of the model is proved straightforwardly, in particular there is no need use another logical relation to relate the model to the operational semantics (as for standard denotational models [13]), since the model is defined using the operational semantics.

Finally, we prove full abstraction. The proof follows essentially along the same lines of the proof of full abstraction in [8], and relies on the use of biorthogonality. It is also crucial that the objects of $\mathcal{C}$ consists of relations on *well-typed* terms and that the evaluation contexts used in the definition of biorthogonality are also well-typed. (See the discussion in Section 5.)

We conclude the paper with a brief discussion in Section 5.

The prerequisites for reading this paper are mild. For the first part on the logical relation, readers are expected to be familiar with call-by-name operational semantics, simple type systems and proving properties of such using

$$t \overset{\text{def}}{=} x \mid \langle\rangle \mid \mathbf{tt} \mid \mathbf{ff} \mid \mathbf{if}\,(t,t,t) \mid \mathbf{fst}\,t \mid \mathbf{snd}\,t \mid \langle t,t\rangle \mid t\,t \mid \lambda\,x\,.\,t \mid \mathbf{fold}\,(t) \mid \mathbf{unfold}\,(t)$$

$$e \overset{\text{def}}{=} [-] \mid \mathbf{fst}\,e \mid \mathbf{snd}\,e \mid e\,t \mid \mathbf{if}\,(e,t,t) \mid \mathbf{unfold}\,(e)$$

$$\begin{aligned} C \overset{\text{def}}{=}\ & [-] \mid \mathbf{fst}\,C \mid \mathbf{snd}\,C \mid \langle C,t\rangle \mid \langle t,C\rangle \mid C\,t \mid t\,C \mid \lambda\,x\,.\,C \mid \\ & \mathbf{if}\,(C,t,t) \mid \mathbf{if}\,(t,C,t) \mid \mathbf{if}\,(t,t,C) \mid \mathbf{unfold}\,(C) \mid \mathbf{fold}\,(C) \end{aligned}$$

Figure 1: Terms and evaluation contexts of PCF$_\mu$.

standard logical relations arguments. For the second part on the categorical model, readers are expected to understand what a cartesian closed category is and that PCF can be interpreted in such a category with fixed points of all endomorphisms. (The second part can be skipped by readers only interested in understanding how the step-indexed logical relation can be used as a proof method for reasoning about contextual equivalence.)

2 The language

We consider a call-by-name variant of PCF with recursive types and with booleans and the unit type as the only ground types.

Assuming a countably infinite supply of term variables, represented by the meta-variable x, we define terms t, evaluation contexts e and contexts C using the rules in Figure 1. Observe that there is no type information in the terms and that we don't include an explicit fixed point combinator as we will be able to define it as a typeable term, since we have recursive types.

We denote by $\mathcal{T}$ the set of *closed* terms of PCF$_\mu$, by $\mathcal{E}$ the set of evaluation contexts and by $\mathbb{C}$ the set of contexts. Let $s\left[t/x\right]$ denote the capture-avoiding substitution of term t for variable x in term s (if x is the only free variable in s, we will also write this as $s(t)$). We define the evaluation relation, $\rightarrow\, \subseteq \mathcal{T} \times \mathcal{T}$, and the reduction relation, $\rightsquigarrow\, \subseteq \mathcal{T} \times \mathcal{T}$, in Figure 2 ($\rightsquigarrow$ is the reflexive and transitive closure of $\rightarrow$). Let $\mathcal{U} = \left\{(e[\mathbf{unfold}\,(\mathbf{fold}\,(t))], e[t]) \mid t \in \mathcal{T}, e \in \mathcal{E}\right\}$. We call the reductions in $\mathcal{U}$ *unfold-fold* reductions and define

$$\rightarrow_1 \overset{\text{def}}{=} (\rightsquigarrow \setminus \mathcal{U}) \circ \mathcal{U} \circ (\rightsquigarrow \setminus \mathcal{U})$$

Thus $t \rightarrow_1 t'$ if $t \rightsquigarrow t'$ and exactly one of the reductions is an unfold-fold reduction.

Let $\mathbb{N}$ be the set of natural numbers $\{1, 2, 3, \ldots\}$. For $n \in \mathbb{N}$ we define the

$$\begin{aligned}\textbf{if}\,(\textbf{tt},t,t') &\to t \\ \textbf{if}\,(\textbf{ff},t,t') &\to t' \\ \textbf{fst}\,\langle t,t'\rangle &\to t \\ \textbf{snd}\,\langle t,t'\rangle &\to t'\end{aligned} \qquad \begin{aligned}(\lambda\,x\,.\,s)\;t &\to s\,[t/x] \\ \textbf{unfold}\,(\textbf{fold}\,(t)) &\to t \\ t\to t' &\Rightarrow e[t]\to e[t']\end{aligned}$$

$$\rightsquigarrow \overset{\text{def}}{=} \bigcup_{n=0}^{\infty} \to^{n}$$

Figure 2: Evaluation relations. We assume that $t, t' \in \mathcal{T}$, x is a variable, s is a term with at most x free and $e \in \mathcal{E}$. The relation $\to$ is deterministic.

n-step reduction relation $\overset{\text{n}}{\rightsquigarrow} \subseteq \mathcal{T} \times \mathcal{T}$ as

$$\overset{\text{n}}{\rightsquigarrow} = (\rightsquigarrow \backslash \mathcal{U}) \cup \bigcup_{i=0}^{n-1} \to_1{}^{i}$$

i.e., $t \overset{\text{n}}{\rightsquigarrow} t'$ when $t \rightsquigarrow t'$ and at most $n-1$ reductions are unfold-fold reductions.[1] The reason for counting only unfold-fold reductions in $\overset{\text{n}}{\rightsquigarrow}$ will become apparent later, but the essence is that this provides some additional freedom which is convenient when working with the logical relation in Section 3 and crucial for establishing some of the properties of the model in Section 4.

Assuming a countably infinite supply of type variables, distinct from term variables and denoted by the meta-variable α, we define the types τ in Figure 3, the typing judgment for terms in Figure 4, for evaluation contexts in Figure 5 and for contexts in Figure 6. The typing judgment for evaluation contexts, $e \vdash \tau \Rrightarrow \sigma$, means that for any closed term t of type τ, $e[t]$ is a closed term of type σ and the judgment $C : \left(\Gamma \;\middle|\; \sigma\right) \rightsquigarrow \left(\Delta \;\middle|\; \tau\right)$ means that if $\Gamma \vdash t : \sigma$ then $\Delta \vdash C[t] : \tau$. We denote by $\mathbb{T}$ be the set of closed types.

Using these we define in Figure 7 for each closed type τ and context Γ the set of terms of type τ in contexts Γ, $\mathbb{T}\,(\Gamma \rhd \tau)$, and the set of closed terms of type τ, $\mathcal{T}_\tau$. For a pair of closed types τ, σ we define the set of typeable evaluation contexts, $\mathbb{E}\,(\tau \rhd \sigma)$, and when $\sigma = \mathbf{2}$ we write $\mathbb{G}\,(\tau)$ for $\mathbb{E}\,(\tau \rhd \sigma)$ (ground evaluation contexts). For a context Γ and a pair of types τ, σ we define a set of *closing* contexts $\mathbb{C}\,(\Gamma, \tau \rhd \sigma)$ and for a context Γ we define the set of *closing substitutions* of $\mathbb{S}\,(\Gamma)$.

Let $\mathfrak{TIR}^2$ be the set of quadruples

$$\left\{(\Gamma, t, t', \sigma) \;\middle|\; \Gamma \vdash t : \sigma \land \Gamma \vdash t' : \sigma\right\}.$$

[1]The reason for allowing only strictly less than n unfold-fold reductions in $t \overset{\text{n}}{\rightsquigarrow} t'$ is that it slightly simplifies later definitions. We could also allow n unfold-fold reductions, but then the definition of the interpretation of recursive types in Figure 9 would have to be altered so that the Lemma 14 would still hold.

[2]$\mathfrak{T}$ype $\mathfrak{I}$indexed $\mathfrak{R}$relations

$$\tau \overset{\mathrm{def}}{=} \alpha \;\big|\; \mathbf{1} \;\big|\; \mathbf{2} \;\big|\; \tau \otimes \tau \;\big|\; \tau \Rightarrow \tau \;\big|\; \mu\,\alpha.\tau$$

Figure 3: Types of the language.

$$\frac{x : \tau \in \operatorname{dom}\Gamma}{\Gamma \vdash x : \tau} \qquad \frac{}{\Gamma \vdash \mathbf{tt} : \mathbf{2}} \qquad \frac{}{\Gamma \vdash \mathbf{ff} : \mathbf{2}}$$

$$\frac{\Gamma \vdash t : \mathbf{2} \qquad \Gamma \vdash s : \tau \qquad \Gamma \vdash r : \tau}{\Gamma \vdash \mathbf{if}\,(t, s, r) : \tau} \qquad \frac{}{\Gamma \vdash \langle\rangle : \mathbf{1}} \qquad \frac{\Gamma, x : \tau \vdash t : \sigma}{\Gamma \vdash \lambda x\,.\,t : \tau \Rightarrow \sigma}$$

$$\frac{\Gamma \vdash t : \tau \Rightarrow \sigma \qquad \Gamma \vdash s : \tau}{\Gamma \vdash t\,s : \sigma} \qquad \frac{\Gamma \vdash t : \sigma \qquad \Gamma \vdash s : \tau}{\Gamma \vdash \langle t, s\rangle : \sigma \otimes \tau} \qquad \frac{\Gamma \vdash t : \sigma \otimes \tau}{\Gamma \vdash \mathbf{fst}\,t : \sigma}$$

$$\frac{\Gamma \vdash t : \sigma \otimes \tau}{\Gamma \vdash \mathbf{snd}\,t : \tau} \qquad \frac{\Gamma \vdash t : \tau\,[\mu\,\alpha.\tau/\alpha]}{\Gamma \vdash \mathbf{fold}\,(t) : \mu\,\alpha.\tau} \qquad \frac{\Gamma \vdash t : \mu\,\alpha.\tau}{\Gamma \vdash \mathbf{unfold}\,(t) : \tau\,[\mu\,\alpha.\tau/\alpha]}$$

Figure 4: Standard typing judgment. We assume that all types declared in Γ are closed.

$$\frac{}{- \vdash \tau \Rrightarrow \tau} \qquad \frac{e \vdash \tau \Rrightarrow \sigma \otimes \rho}{\mathbf{fst}\,e \vdash \tau \Rrightarrow \sigma} \qquad \frac{e \vdash \tau \Rrightarrow \sigma \otimes \rho}{\mathbf{snd}\,e \vdash \tau \Rrightarrow \rho}$$

$$\frac{e \vdash \tau \Rrightarrow \sigma \Rightarrow \rho \qquad \cdot \vdash t : \sigma}{e\,t \vdash \tau \Rrightarrow \rho}$$

$$\frac{e \vdash \tau \Rrightarrow \mathbf{2} \qquad \cdot \vdash t : \sigma \qquad \cdot \vdash t' : \sigma}{\mathbf{if}\,(e, t, t') \vdash \tau \Rrightarrow \sigma}$$

$$\frac{e \vdash \tau \Rrightarrow \mu\,\alpha.\sigma}{\mathbf{unfold}\,(e) \vdash \tau \Rrightarrow \sigma\,[\mu\,\alpha.\sigma/\alpha]}$$

Figure 5: Typing judgment for evaluation contexts.

$$\frac{}{- \,:\, (\Gamma \mid \tau) \rightsquigarrow (\Gamma \mid \tau)} \qquad \frac{C \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \sigma \otimes \delta)}{\mathbf{fst}\, C \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \sigma)} \qquad \frac{C \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \sigma \otimes \delta)}{\mathbf{snd}\, C \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \delta)}$$

$$\frac{C \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \sigma) \qquad \Gamma \vdash t : \delta}{\langle C, t\rangle \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \sigma \otimes \delta)} \qquad \frac{C \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \delta) \qquad \Gamma \vdash t : \sigma}{\langle t, C\rangle \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \sigma \otimes \delta)}$$

$$\frac{C \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \delta \Rightarrow \sigma) \qquad \Gamma \vdash t : \delta}{C\, t \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \sigma)}$$

$$\frac{C \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \delta) \qquad \Gamma \vdash t : \delta \Rightarrow \sigma}{t\, C \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \sigma)}$$

$$\frac{C \,:\, (\Delta, x : \sigma \mid \tau) \rightsquigarrow (\Gamma, x : \sigma \mid \delta)}{\lambda\, x \,.\, C \,:\, (\Delta, x : \sigma \mid \tau) \rightsquigarrow (\Gamma \mid \sigma \Rightarrow \delta)} \qquad \frac{C \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \mu\,\alpha.\sigma)}{\mathbf{unfold}\,(C) \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \sigma\,[\mu\,\alpha.\sigma/\alpha])}$$

$$\frac{C \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \sigma\,[\mu\,\alpha.\sigma/\alpha])}{\mathbf{fold}\,(C) \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \mu\,\alpha.\sigma)} \qquad \frac{C \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \mathbf{2}) \qquad \Gamma \vdash t : \delta \qquad \Gamma \vdash t' : \delta}{\mathbf{if}\,(C, t, t') \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \delta)}$$

$$\frac{\Gamma \vdash t : \mathbf{2} \qquad C \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \delta) \qquad \Gamma \vdash t' : \delta}{\mathbf{if}\,(t, C, t') \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \delta)}$$

$$\frac{\Gamma \vdash t : \mathbf{2} \qquad \Gamma \vdash t' : \delta \qquad C \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \delta)}{\mathbf{if}\,(t, t', C) \,:\, (\Delta \mid \tau) \rightsquigarrow (\Gamma \mid \delta)}$$

Figure 6: Typing judgment for contexts. The interesting rule is the rule for function abstraction where a variable is bound by the λ. All other rules are straightforward and follow the structure of contexts.

$$\mathbb{T}(\Gamma \rhd \tau) \stackrel{\text{def}}{=} \{t \mid \Gamma \vdash t:\tau\} \qquad \mathbb{E}(\tau \rhd \sigma) \stackrel{\text{def}}{=} \{e \in \mathcal{E} \mid e \vdash \tau \Rrightarrow \sigma\}$$
$$\mathcal{T}_\tau \stackrel{\text{def}}{=} \mathbb{T}(\cdot \rhd \tau) \qquad \mathbb{G}(\tau) \stackrel{\text{def}}{=} \{e \in \mathcal{E} \mid e \vdash \tau \Rrightarrow \mathbf{2}\}$$

$$\mathbb{S}(\cdot) = \{\emptyset\}$$
$$\mathbb{S}(\Gamma, x:\tau) = \{\gamma[x \mapsto t] \mid \gamma \in \mathbb{S}(\Gamma), t \in \mathcal{T}_\tau\}$$

$$\mathbb{C}(\Gamma, \tau \rhd \sigma) \stackrel{\text{def}}{=} \{C \in \mathbb{C} \mid C : (\Gamma \mid \tau) \rightsquigarrow (\cdot \mid \sigma)\}.$$

Figure 7: The set of terms of type τ in context Γ, $\mathbb{T}(\Gamma \rhd \tau)$, the set of closed terms of type τ, $\mathcal{T}_\tau$, the set of evaluation contexts with hole of type τ and result of type σ, $\mathbb{E}(\tau \rhd \sigma)$, the set of *ground* evaluation contexts of type τ, $\mathbb{G}(\tau)$ and the set of closing contexts with a hole of type τ binding variables declared in Γ resulting in term of type σ, $\mathbb{C}(\Gamma, \tau \rhd \sigma)$. $\mathbb{S}(\Gamma)$ is the pointwise extension of $\mathcal{T}$ to contexts and is defined inductively on Γ.

We say that $\mathcal{R}$ is a *type-indexed relation* if $\mathcal{R} \subseteq \mathfrak{TIR}$ and in this case write $\Gamma \vdash t \mathrel{\mathcal{R}} t' : \sigma$ for $(\Gamma, t, t', \sigma) \in \mathcal{R}$.

A type-indexed relation $\mathcal{R}$ is

adequate if for all $t, t' \in \mathcal{T}_\mathbf{2}$ $\cdot \vdash t \mathrel{\mathcal{R}} t' : \mathbf{2}$ implies $t \rightsquigarrow \mathbf{tt} \Rightarrow t' \rightsquigarrow \mathbf{tt}$,

reflexive if $\Gamma \vdash t:\sigma$ implies $\Gamma \vdash t \mathrel{\mathcal{R}} t : \sigma$,

transitive if $\Gamma \vdash t \mathrel{\mathcal{R}} t' : \sigma$ and $\Gamma \vdash t' \mathrel{\mathcal{R}} t'' : \sigma$ imply $\Gamma \vdash t \mathrel{\mathcal{R}} t'' : \sigma$

precongruence if it is reflexive, transitive and closed under the rules in Figure 8.

We call the rules in Figure 8 *compatibility* rules and a type-indexed relation satisfying them *compatible.* The compatibility rules follow the typing rules and are precisely the rules we need and expect for modular reasoning.

Following [11] we define two notions of approximation and equivalence, contextual approximation and equivalence and CIU approximation and equivalence.

Definition 1 (contextual approximation and equivalence)**.**

$$\Gamma \vdash s \leq_{ctx} t : \tau \stackrel{def}{=} \forall C \in \mathbb{C}(\Gamma, \tau \rhd \mathbf{2}), C[s] \rightsquigarrow \mathbf{tt} \Rightarrow C[t] \rightsquigarrow \mathbf{tt}$$
$$\Gamma \vdash s \equiv_{ctx} t : \tau \stackrel{def}{=} \Gamma \vdash s \leq_{ctx} t : \tau \wedge \Gamma \vdash t \leq_{ctx} s : \tau$$

Definition 2 (CIU approximation and equivalence)**.**

$$\Gamma \vdash s \leq_{ciu} t : \tau \stackrel{def}{=} \forall e \in \mathbb{G}(\tau), \forall \gamma \in \mathbb{S}(\Gamma), e[\gamma(s)] \rightsquigarrow \mathbf{tt} \Rightarrow e[\gamma(t)] \rightsquigarrow \mathbf{tt}$$
$$\Gamma \vdash s \equiv_{ciu} t : \tau \stackrel{def}{=} \Gamma \vdash s \leq_{ciu} t : \tau \wedge \Gamma \vdash t \leq_{ciu} s : \tau$$

$$\frac{}{\Gamma \vdash \mathbf{tt} \;\mathcal{R}\; \mathbf{tt} : \mathbf{2}} \qquad \frac{}{\Gamma \vdash \mathbf{ff} \;\mathcal{R}\; \mathbf{ff} : \mathbf{2}} \qquad \frac{}{\Gamma \vdash \langle\rangle \;\mathcal{R}\; \langle\rangle : \mathbf{1}}$$

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x \;\mathcal{R}\; x : \tau} \qquad \frac{\Gamma \vdash t \;\mathcal{R}\; t' : \sigma \otimes \delta}{\Gamma \vdash \mathbf{fst}\, t \;\mathcal{R}\; \mathbf{fst}\, t' : \sigma} \quad \frac{\Gamma \vdash t \;\mathcal{R}\; t' : \sigma \otimes \delta}{\Gamma \vdash \mathbf{snd}\, t \;\mathcal{R}\; \mathbf{snd}\, t' : \delta}$$

$$\frac{\Gamma \vdash t \;\mathcal{R}\; t' : \sigma \qquad \Gamma \vdash s \;\mathcal{R}\; s' : \delta}{\Gamma \vdash \langle t, s\rangle \;\mathcal{R}\; \langle t', s'\rangle : \sigma \otimes \delta} \quad \frac{\Gamma \vdash t \;\mathcal{R}\; t' : \sigma \Rightarrow \delta \qquad \Gamma \vdash s \;\mathcal{R}\; s' : \sigma}{\Gamma \vdash t\, s \;\mathcal{R}\; t'\, s' : \delta}$$

$$\frac{\Gamma, x : \sigma \vdash t \;\mathcal{R}\; t' : \delta}{\Gamma \vdash \lambda x\,.\,t \;\mathcal{R}\; \lambda x\,.\,t' : \sigma \Rightarrow \delta} \quad \frac{\Gamma \vdash t \;\mathcal{R}\; t' : \mu\,\alpha.\sigma}{\Gamma \vdash \mathbf{unfold}\,(t) \;\mathcal{R}\; \mathbf{unfold}\,(t') : \sigma\left[\mu\,\alpha.\sigma/\alpha\right]}$$

$$\frac{\Gamma \vdash t \;\mathcal{R}\; t' : \sigma\left[\mu\,\alpha.\sigma/\alpha\right]}{\Gamma \vdash \mathbf{fold}\,(t) \;\mathcal{R}\; \mathbf{fold}\,(t') : \mu\,\alpha.\sigma} \quad \frac{\Gamma \vdash p \;\mathcal{R}\; p' : \mathbf{2} \qquad \Gamma \vdash t \;\mathcal{R}\; t' : \tau \qquad \Gamma \vdash s \;\mathcal{R}\; s' : \tau}{\Gamma \vdash \mathbf{if}\,(p, t, s) \;\mathcal{R}\; \mathbf{if}\,(p', t', s') : \tau}$$

Figure 8: Compatibility rules for a type-indexed relation $\mathcal{R}$.

Contextual equivalence says that we consider two terms equivalent if we may safely interchange them when part of any complete program. This is the notion of equivalence we need for proving, for instance, correctness of compiler optimizations. **C**losed **I**nstantiation of **U**se equivalence says that we consider two terms equivalent if they behave in the same way when used (put into an evaluation position in a term). Informally, if a term in a context is never used, that is, evaluated, then it can be safely replaced by any other term, whereas if it is used, then it will be used under some evaluation context. Nevertheless, it is not obvious that these two notions of equivalence (or approximation) coincide. We prove this as Theorem 18 by constructing another notion of approximation.

The two notions of equivalence are useful for proving different properties. For example, it is easy to see, Lemma 19, that CIU-equivalence is closed under reductions, but seeing that CIU-equivalence is compatible is non-trivial, whereas, for example, it is easy to see that contextual equivalence is a precongruence, but it is very difficult to show directly from the definition that contextual equivalence is closed under reductions.

It is not difficult to prove that contextual approximation is an adequate precongruence and in fact it is the largest such.

Proposition 3. *If $\mathcal{R}$ is an adequate precongruence then for all terms $t, t' \in \mathbb{T}\,(\Gamma \rhd \sigma)$, $\Gamma \vdash t \;\mathcal{R}\; t' : \sigma$ implies $\Gamma \vdash t \leq_{ctx} t' : \sigma$.*

Proof sketch. Let $C \in \mathbb{C}\,(\Gamma, \sigma \rhd \mathbf{2})$. As $\mathcal{R}$ is a precongruence, $\cdot \vdash C[t] \;\mathcal{R}\; C[t'] : \mathbf{2}$. The fact that $\mathcal{R}$ is also adequate concludes the proof. □

2.1 Some properties of the syntax and the reduction relations

We list here some syntactic properties used later. They are all proved by a simple induction on the typing derivation.

Lemma 4. *If* $e \in \mathbb{E}\left(\tau \otimes \sigma \rhd \rho\right)$ *then*

$$(e = - \wedge \rho = \tau \otimes \sigma) \vee (\exists f \in \mathbb{E}\left(\tau \rhd \rho\right), e = f[\mathbf{fst}\,-]) \vee (\exists f \in \mathbb{E}\left(\sigma \rhd \rho\right), e = f[\mathbf{snd}\,-])$$

Lemma 5. *If* $e \in \mathbb{E}\left(\tau \Rightarrow \sigma \rhd \rho\right)$ *then*

$$(e = - \wedge \rho = \tau \Rightarrow \sigma) \vee (\exists t \in \mathcal{T}_\tau, \exists f \in \mathbb{E}\left(\sigma \rhd \rho\right), e = f[-t])$$

Lemma 6. *If* $e \in \mathbb{E}\left(\mu\,\alpha.\tau \rhd \rho\right)$ *then*

$$(e = - \wedge \rho = \mu\,\alpha.\tau) \vee \left(\exists f \in \mathbb{E}\left(\tau\left[\mu\,\alpha.\tau/\alpha\right] \rhd \rho\right), e = f[\mathbf{unfold}\,(-)]\right)$$

3 Logical relation

Let $\mathcal{C}_0$ be the set of decreasing sequences of binary relations on closed *typeable* terms,[3]

$$\mathcal{C}_0 \stackrel{\text{def}}{=} \left\{X_\tau : \mathbb{N} \to \mathcal{P}\left(\mathcal{T}_\tau \times \mathcal{T}_\tau\right) \;\middle|\; \tau \in \mathbb{T} \wedge \forall n \in \mathbb{N}, X_\tau(n) \supseteq X_\tau(n+1)\right\},$$

and let $\mathcal{G}$ be the set of decreasing sequences of binary relations on *typeable* ground evaluation contexts, explicitly

$$\mathcal{G} \stackrel{\text{def}}{=} \left\{X_\tau : \mathbb{N} \to \mathcal{P}\left(\mathbb{G}\left(\tau\right) \times \mathbb{G}\left(\tau\right)\right) \;\middle|\; \tau \in \mathbb{T} \wedge \forall n \in \mathbb{N}, X_\tau(n) \supseteq X_\tau(n+1)\right\}.$$

Each element of $\mathcal{C}_0$ and $\mathcal{G}$ thus has a type associated with it and it relates only typeable terms or evaluation contexts of that type. We will often omit type indices of elements of $\mathcal{C}_0$ and $\mathcal{G}$ when we have no explicit need for them.

If $X \in \mathcal{C}_0$ and $(c, c') \in X_\tau(n)$ we think of c and c' as indistinguishable if we only have n computation steps available. The monotonicity condition, $X_\tau(n) \supseteq X_\tau(n+1)$, then states that if we have more computation steps available, we can distinguish more.

We define two functions, $(\cdot)\downarrow : \mathcal{C}_0 \to \mathcal{G}$ and $(\cdot)\uparrow : \mathcal{G} \to \mathcal{C}_0$, as follows

$$X_\tau \downarrow (n) \stackrel{\text{def}}{=} \left\{(e, e') \in \mathbb{G}\left(\tau\right) \times \mathbb{G}\left(\tau\right) \;\middle|\; \forall i \leq n, \forall (v, v') \in X_\tau(i), e[v] \overset{\text{i}}{\rightsquigarrow} \mathbf{tt} \Rightarrow e'[v'] \rightsquigarrow \mathbf{tt}\right\}$$

$$X_\tau \uparrow (n) \stackrel{\text{def}}{=} \left\{(t, t') \in \mathcal{T}_\tau \times \mathcal{T}_\tau \;\middle|\; \forall i \leq n, \forall (e, e') \in X(i), e[t] \overset{\text{i}}{\rightsquigarrow} \mathbf{tt} \Rightarrow e'[t'] \rightsquigarrow \mathbf{tt}\right\}.$$

[3]The reason for subscript $_0$ in $\mathcal{C}_0$ is that we intend to use this set as the set of object of the category $\mathcal{C}$ in Section 4.

It is easy to see that the maps are well defined and we denote their composite from $\mathcal{C}_0$ to $\mathcal{C}_0$ by $(\cdot)^{\top\top}$. We refer to related elements in $X^{\top\top}(n)$, i.e. $(c, c') \in X^{\top\top}(n)$, as *computations*.

The quantification over lower indices in the definitions of maps might appear mysterious, but it is there simply to ensure that the constructions stay within $\mathcal{C}_0$ and $\mathcal{G}$, i.e. that the constructed sequence of relations is monotonically decreasing. Intuitively, this is again because we think of terms related at n to be that are indistinguishable if we only use n unfold-fold reductions available to test with (the unfold-fold reductions are the only reduction steps we count).

We will often implicitly use the following lemma, expressing the fact that $(\cdot)^{\top\top}$ is a closure operator.

Lemma 7. *Let $X, Y \in \mathcal{C}_0$.*

- $\forall n \in \mathbb{N}$, $X(n) \subseteq X^{\top\top}(n)$.
- *If* $\forall n \in \mathbb{N}, X(n) \subseteq Y(n)$ *then* $\forall n \in \mathbb{N}, X^{\top\top}(n) \subseteq Y^{\top\top}(n)$.
- $\left(X^{\top\top}\right)^{\top\top} = X^{\top\top}$.

We define constructions on $\mathcal{C}_0$ which will be used for modeling the types of the language.

$$\begin{aligned}
\mathbb{B}_{\mathbf{2}}(n) &\overset{\text{def}}{=} \{(\mathbf{tt}, \mathbf{tt}), (\mathbf{ff}, \mathbf{ff})\} \\
(X_\sigma \times Y_\tau)(n) &\overset{\text{def}}{=} \left\{(\langle c, d\rangle, \langle c', d'\rangle) \;\middle|\; (c, c') \in X_\sigma^{\top\top}(n) \wedge (d, d') \in Y_\tau^{\top\top}(n)\right\} \\
\left((Y_\sigma)^{(X_\tau)}\right)(n) &\overset{\text{def}}{=} \bigcap_{i=1}^{n} \left\{ (\lambda\, x\,.\, c, \lambda\, x\,.\, c') \;\middle|\; \begin{array}{l} \cdot \vdash \lambda\, x\,.\, c : \tau \Rightarrow \sigma \wedge \cdot \vdash \lambda\, x\,.\, c' : \tau \Rightarrow \sigma \wedge \\ \forall (d, d') \in X_\tau^{\top\top}(i), \forall (e, e') \in Y_\sigma \downarrow (i), \\ e[(\lambda\, x\,.\, c)\ d] \overset{\text{i}}{\rightsquigarrow} \mathbf{tt} \Rightarrow e'[(\lambda\, x\,.\, c')\ d'] \rightsquigarrow \mathbf{tt} \end{array} \right\}
\end{aligned}$$

The difference from the constructions used for modeling call-by-value languages is that the canonical elements of the product type in a call-by-name language are pairs of *computations*, rather than pairs of values. There is also a natural change in that the canonical elements of the function type have to map related *computations* to related computations, rather than related *values* to related computations.

Having defined these, we define the interpretation of types in Figure 9. The interpretation is defined by induction on the index and type — formally $[\![\tau]\!]\,(n)$ is defined inductively on the lexicographic ordering of $\mathbb{N} \times \mathbb{T}$.

We extend the interpretation of types to contexts as follows

$$\begin{aligned}
[\![\cdot]\!] &= n \mapsto \{(\emptyset, \emptyset)\} \\
[\![\Gamma, x : \tau]\!] &= n \mapsto \{(\gamma[x \mapsto c], \gamma'[x \mapsto c']) \;\mid\; (\gamma, \gamma') \in [\![\Gamma]\!]\,(n) \wedge (c, c') \in [\![\tau]\!]^{\top\top}(n)\}.
\end{aligned}$$

Note that related substitutions map variables to *computations*. This choice is forced by the definition of the interpretation of the arrow type (specifically

$$
\begin{aligned}
[\![\mathbf{2}]\!] &= n \mapsto \{(\mathbf{tt},\mathbf{tt}),(\mathbf{ff},\mathbf{ff})\}\\
[\![\mathbf{1}]\!] &= n \mapsto \{(\langle\rangle,\langle\rangle)\}\\
[\![\tau \otimes \sigma]\!] &= [\![\tau]\!] \times [\![\sigma]\!]\\
[\![\tau \Rightarrow \sigma]\!] &= [\![\sigma]\!]^{[\![\tau]\!]}\\
[\![\mu\,\alpha.\tau]\!]\,(1) &= \mathcal{T}_{\mu\,\alpha.\tau} \times \mathcal{T}_{\mu\,\alpha.\tau}\\
[\![\mu\,\alpha.\tau]\!]\,(n+1) &= \bigcap_{i=1}^{n} \Big\{ (\mathbf{fold}\,(c)\,,\mathbf{fold}\,(c')) \;\Big|\; (c,c') \in [\![\tau\,[\mu\,\alpha.\tau/\alpha]]\!]^{\top\top}(i) \Big\}
\end{aligned}
$$

Figure 9: Interpretation of types.

the proof of compatibility for the case of lambda terms in Proposition 15). This is different from the situation in logical relations for call-by-value languages, where closing substitutions map variables to related values.

Finally, we define the *logical approximation relation* $\prec$ as a subset of $\mathfrak{TIR}$ as follows

$$\Gamma \vdash t \prec t' : \sigma \overset{\text{def}}{=} \forall n \in \mathbb{N}, \forall(\gamma,\gamma') \in [\![\Gamma]\!]\,(n), (\gamma(t),\gamma'(t')) \in [\![\sigma]\!]^{\top\top}(n).$$

3.1 Soundness with respect to contextual approximation

We wish to show that $\Gamma \vdash t \prec t' : \sigma$ implies $\Gamma \vdash t \leq_{ctx} t' : \sigma$ but we will show more. We will show that the logical, CIU and contextual approximations all coincide.

We first state two lemmas which, although proved rather trivially, are nevertheless important and often used in later proofs. They show that we can do some reductions on related terms so that the terms are still related.

Lemma 8. *Let $n \in \mathbb{N}$ and $(c,c') \in X^{\top\top}(n)$. If $d \rightsquigarrow c$ and $d_1 \rightsquigarrow c' \rightsquigarrow d_2$, then $(d,d_1) \in X^{\top\top}(n)$ and $(d,d_2) \in X^{\top\top}(n)$.*

Lemma 9. *Let $n \in \mathbb{N}$ and $(c,c') \in X^{\top\top}(n)$. If $d \rightarrow_1 c$ and $d_1 \rightsquigarrow c' \rightsquigarrow d_2$, then $(d,d_1) \in X^{\top\top}(n+1)$ and $(d,d_2) \in X^{\top\top}(n+1)$.*

Another simple property of computations is that a diverging computation approximates any other computation of the same type.

Lemma 10. *Let $X_\sigma \in \mathcal{C}_0$ and $t \in \mathcal{T}_\sigma$. If t diverges then $\forall t' \in \mathcal{T}_\sigma, \forall n \in \mathbb{N}, (t,t') \in X_\sigma^{\top\top}(n)$*

The next four lemmas show how we can extend related evaluation contexts with elimination forms to form a new pair of related evaluation contexts. They are crucial for showing compatibility of elimination forms in Proposition 15.

Lemma 11. *For all $X \in \mathcal{C}_0$, $n \in \mathbb{N}$, $(e, e') \in X \downarrow (n)$, $(c, c') \in X(n)$, $(d, d') \in X(n)$,*

$$(e[\mathbf{if}\,(-, c, d)], e'[\mathbf{if}\,(-, c', d')]) \in \mathbb{B} \downarrow (n).$$

Proof. Let $i \leq n$ and $(t, t') \in \mathbb{B}(i)$ and assume $e[\mathbf{if}\,(t, c, d)] \overset{\text{i}}{\rightsquigarrow} \mathbf{tt}$. We naturally consider two cases

$t = t' = \mathbf{tt}$ Then $e[c] \overset{\text{i}}{\rightsquigarrow} \mathbf{tt}$ and so, $e'[c'] \rightsquigarrow \mathbf{tt}$ and so $e'[\mathbf{if}\,(t', c', d') \rightsquigarrow \mathbf{tt}$

$t = t' = \mathbf{ff}$ Then $e[d] \overset{\text{i}}{\rightsquigarrow} \mathbf{tt}$ and so, $e'[d'] \rightsquigarrow \mathbf{tt}$ and so $e'[\mathbf{if}\,(t', c', d')] \rightsquigarrow \mathbf{tt}$. □

Lemma 12. *Let $X, Y \in \mathcal{C}_0$ and $n \in \mathbb{N}$. If $(e, e') \in X \downarrow (n)$ then $(e[\mathbf{fst}\,-], e'[\mathbf{fst}\,-]) \in (X \times Y) \downarrow (n)$ and $(e[\mathbf{snd}\,-], e'[\mathbf{snd}\,-]) \in (Y \times X) \downarrow (n)$.*

Proof. Notice that because we are in a call-by-name language, the two conclusions we have to prove are in fact completely symmetric, so we content ourselves in only proving the case for **fst**.

Take $i \leq n$ and $(\langle c, d\rangle, \langle c', d'\rangle) \in (X \times Y)(i)$ and assume $e[\mathbf{fst}\,\langle c, d\rangle] \overset{\text{i}}{\rightsquigarrow} \mathbf{tt}$. This means $e[c] \overset{\text{i}}{\rightsquigarrow} \mathbf{tt}$. Using the assumptions $(e, e') \in X \downarrow (n) \subseteq X \downarrow (i)$ and $(c, c') \in X^{\top\top}(n) \subseteq X^{\top\top}(i)$, we have $e'[c'] \rightsquigarrow \mathbf{tt}$, but this also means $e'[\mathbf{fst}\,\langle c', d'\rangle] \rightsquigarrow \mathbf{tt}$, and we are done. □

Lemma 13. *Let $X, Y \in \mathcal{C}_0$ and $n \in \mathbb{N}$. If $(e, e') \in Y \downarrow (n)$ and $(c, c') \in X^{\top\top}(n)$ then $(e[-c], e'[-c']) \in \left(Y^X\right) \downarrow (n)$.*

Proof. Take $i \leq n$ and $(f, f') \in \left(Y^X\right)(i)$ and assume $e[f\,c] \overset{\text{i}}{\rightsquigarrow} \mathbf{tt}$. By construction $\forall i \leq n, \forall (b, b') \in X^{\top\top}(i)$ and $\forall (g, g') \in Y \downarrow (i), g[f\,b] \overset{\text{i}}{\rightsquigarrow} \mathbf{tt} \Rightarrow g'[f'\,b'] \rightsquigarrow \mathbf{tt}$. If we instantiate this with (e, e') and (c, c') we get $e'[f'\,c'] \rightsquigarrow \mathbf{tt}$, which is exactly what we need. □

Lemma 14. *If $(e, e') \in [\![\tau\,[\mu\,\alpha.\tau/\alpha]]\!] \downarrow (n)$ then $(e[\mathbf{unfold}\,(-)], e'[\mathbf{unfold}\,(-)]) \in [\![\mu\,\alpha.\tau]\!] \downarrow (n)$.*

Proof. Let $1 < i \leq n$ and $(\mathbf{fold}\,(c), \mathbf{fold}\,(c')) \in [\![\mu\,\alpha.\sigma]\!](i)$. By definition of the interpretation of recursive types this means $(c, c') \in [\![\tau\,[\mu\,\alpha.\tau/\alpha]]\!](i-1)$. Assume $e\,[\mathbf{unfold}\,(\mathbf{fold}\,(c))] \overset{\text{i}}{\rightsquigarrow} \mathbf{tt}$. This implies $e[c] \overset{\text{i-1}}{\rightsquigarrow} \mathbf{tt}$ and so $e'[c'] \rightsquigarrow \mathbf{tt}$ and so $e'\,[\mathbf{unfold}\,(\mathbf{fold}\,(c'))] \rightsquigarrow \mathbf{tt}$ and we are done.

For $i = 1$ the conclusion holds vacuously as for $(c, c') \in X(1)$, $e[\mathbf{unfold}\,(c)] \overset{1}{\rightsquigarrow} \mathbf{tt}$ does not hold.[4] □

Proposition 15. *The logical approximation relation is closed under all the rules in Figure 8. In particular, it is reflexive.*

[4]Here it is crucial that $t \overset{\text{n}}{\rightsquigarrow} t'$ means that t evaluates to t' in strictly less than n unfold-fold reductions.

Proof. Note first that any relation that is closed under the rules in Figure 8 is necessarily reflexive. Most of the cases in the proof of compatibility for the logical approximation are straightforward. We only show a few representative ones here, to explain how the above lemmas are used for proving compatibility of elimination forms and how the compatibility of introduction forms is shown directly.

- The case for lambda abstraction. Suppose $\Gamma, x : \sigma \vdash t \prec t' : \delta$. We must show
$$\Gamma \vdash \lambda\, x\,.\, t \prec \lambda\, x\,.\, t' : \sigma \Rightarrow \delta.$$
Let $n \in \mathbb{N}$, $(\gamma, \gamma') \in [\![\Gamma]\!]\,(n)$. We must show $(\gamma(\lambda\, x\,.\, t), \gamma'(\lambda\, x\,.\, t')) \in [\![\sigma \Rightarrow \delta]\!]^{\top\top}(n)$ but we will do even better and show $(\gamma(\lambda\, x\,.\, t), \gamma'(\lambda\, x\,.\, t')) \in [\![\sigma \Rightarrow \delta]\!]\,(n)$. First, $\gamma(\lambda\, x\,.\, t) = \lambda\, x\,.\, \gamma(t)$ and $\gamma'(\lambda\, x\,.\, t') = \lambda\, x\,.\, \gamma'(t')$ so the terms are of the right form. Next, let $i \leq n$, $(c, c') \in \sigma^{\top\top}(i)$ and $(e, e') \in \delta \downarrow (i)$ and assume $e[(\lambda\, x\,.\, \gamma(t))\, c] \overset{i}{\rightsquigarrow} \mathbf{tt}$. This implies that $e\,[(\gamma[x \mapsto c])\,(t)] \overset{i}{\rightsquigarrow} \mathbf{tt}$ and as it is easily seen that $(\gamma[x \mapsto c], \gamma'[x \mapsto c']) \in [\![\Gamma, x : \sigma]\!]$ we use the assumption $\Gamma, x : \sigma \vdash t \prec t' : \delta$ and instantiate it with i and $(\gamma[x \mapsto c], \gamma'[x \mapsto c'])$ to get $((\gamma[x \mapsto c])\,(t), (\gamma'[x \mapsto c'])\,(t')) \in [\![\delta]\!]^{\top\top}(i)$. This then implies $e'\,[(\gamma'[x \mapsto c'])\,(t')] \rightsquigarrow \mathbf{tt}$, which implies $e'\,[(\lambda\, x\,.\, \gamma'(t'))\, c'] \rightsquigarrow \mathbf{tt}$

- The case for application. Suppose $\Gamma \vdash t \prec t' : \sigma \Rightarrow \delta$ and $\Gamma \vdash s \prec s' : \sigma$. We must show $\Gamma \vdash t\, s \prec t'\, s' : \delta$ so take $n \in \mathbb{N}$ and $(\gamma, \gamma') \in [\![\Gamma]\!]\,(n)$. We must then show
$$(\gamma\,(t\, s)\,, \gamma'\,(t'\, s')) \in [\![\delta]\!]^{\top\top}(n).$$
Unfolding the definitions, let $i \leq n$, $(e, e') \in [\![\delta]\!] \downarrow (i)$. Using Lemma 13 and the induction hypothesis we get $(e[-\, \gamma(s)], e'[-\, \gamma'(s')]) \in [\![\sigma \Rightarrow \delta]\!] \downarrow (i)$ and also by induction hypothesis we have $(\gamma(t), \gamma'(t')) \in [\![\sigma \Rightarrow \delta]\!]^{\top\top}(n) \subseteq [\![\sigma \Rightarrow \delta]\!]^{\top\top}(i)$. In the end, we have to show $e[\gamma\,(t\, s)] \overset{i}{\rightsquigarrow} \mathbf{tt} \Rightarrow e'[\gamma'\,(t'\, s')] \rightsquigarrow \mathbf{tt}$ and this now follows directly from what we showed, as substitution distributes over application.

- The case for **fold**. Suppose $\Gamma \vdash t \prec t' : \tau\,\left[\mu\, \alpha.\tau/\alpha\right]$. We must show
$$\Gamma \vdash \mathbf{fold}\,(t) \prec \mathbf{fold}\,(t') : \mu\, \alpha.\tau$$
so let $n \in \mathbb{N}, (\gamma, \gamma') \in [\![\Gamma]\!]\,(n)$. By assumption $(\gamma(t), \gamma'(t')) \in \left[\!\left[\tau\,\left[\mu\, \alpha.\tau/\alpha\right]\right]\!\right]^{\top\top}(n)$. Then by construction of $[\![\mu\, \alpha.\tau]\!]$ we have $(\mathbf{fold}\,(\gamma(t))\,, \mathbf{fold}\,(\gamma'(t'))) \in [\![\mu\, \alpha.\tau]\!]\,(n+1) \subseteq [\![\mu\, \alpha.\tau]\!]^{\top\top}(n)$ which is exactly what was needed.

- The case for **unfold**. Suppose $\Gamma \vdash t \prec t' : \mu\, \alpha.\tau$. We must show
$$\Gamma \vdash \mathbf{unfold}\,(t) \prec \mathbf{unfold}\,(t') : \tau\,\left[\mu\, \alpha.\tau/\alpha\right]$$
so let $n \in \mathbb{N}$, $(\gamma, \gamma') \in [\![\Gamma]\!]\,(n)$, $i \leq n$, $(e, e') \in \left[\!\left[\tau\,\left[\mu\, \alpha.\tau/\alpha\right]\right]\!\right] \downarrow (i)$. Lemma 14 shows $(e[\mathbf{unfold}\,(-)], e'[\mathbf{unfold}\,(-)]) \in [\![\mu\, \alpha.\tau]\!] \downarrow (i)$. Combining this with the induction hypothesis we proceed exactly as in the case for application above. □

Corollary 16. *Let $\Gamma = (x_i : \tau_i)_{i=1}^n$ and assume $\Gamma \vdash t : \sigma$. Let $k \in \mathbb{N}$ and for $i = 1, 2, \ldots, n$, $(c_i, c_i') \in [\![\tau_i]\!]^{\top\top}(k)$. Then $\left(t\left[c_i/x_i\right]_{i=1}^n, t\left[c_i'/x_i\right]_{i=1}^n\right) \in [\![\sigma]\!]^{\top\top}(k)$.*

Proof. Let $\gamma = [x_i \mapsto c_i]_{i=1}^n$ and $\gamma' = [x_i \mapsto c_i']_{i=1}^n$. Then $(\gamma, \gamma') \in [\![\Gamma]\!](k)$. Proposition 15 shows $\Gamma \vdash t \prec t : \sigma$ and if we instantiate this with k and (γ, γ') we get the desired conclusion. □

Proposition 17. *For all terms t, t' of type σ in context Γ, we have $\Gamma \vdash t \prec t' : \sigma$ if and only if $\Gamma \vdash t \leq_{ciu} t' : \sigma$.*

Proof. $\Rightarrow$ Assume $\Gamma \vdash t \prec t' : \sigma$. Let $\gamma \in \mathbb{S}(\Gamma)$ be a closing substitution and $e \in \mathbb{G}(\sigma)$. Assuming $e[\gamma(t)] \rightsquigarrow \mathbf{tt}$ we have to show $e[\gamma(t')] \rightsquigarrow \mathbf{tt}$. Proposition 15 shows that $\forall n \in \mathbb{N}$, $(\gamma, \gamma) \in [\![\Gamma]\!](n)$ and thus also that $\forall n \in \mathbb{N}$, $(\gamma(t), \gamma(t')) \in [\![\sigma]\!]^{\top\top}(n)$. The same proposition then also shows that $\forall n \in \mathbb{N}, (e[\gamma(t)], e[\gamma(t')]) \in [\![\mathbf{2}]\!]^{\top\top}(n)$. The assumption that $e[\gamma(t)] \rightsquigarrow \mathbf{tt}$ implies there exists a natural number k, such that $e[\gamma(t)] \overset{k}{\rightsquigarrow} \mathbf{tt}$. It is easy to see directly from the definition that $\forall n \in \mathbb{N}$, $(-, -) \in [\![\mathbf{2}]\!]\downarrow(n)$. Picking an $n \geq k$ we have $e[\gamma(t')] \rightsquigarrow \mathbf{tt}$, as required.

$\Leftarrow$ Let $n \in \mathbb{N}$, $(\gamma, \gamma') \in [\![\Gamma]\!](n)$. We have to show $(\gamma(t), \gamma'(t')) \in [\![\sigma]\!]^{\top\top}(n)$ so let $i \leq n$, $(e, e') \in [\![\sigma]\!]\downarrow(i)$ and assume $e[\gamma(t)] \overset{i}{\rightsquigarrow} \mathbf{tt}$. Corollary 16 implies $(\gamma(t), \gamma'(t)) \in [\![\sigma]\!]^{\top\top}(n)$ (reflexivity). Monotonicity implies $(\gamma(t), \gamma'(t)) \in [\![\sigma]\!]^{\top\top}(i)$ and thus $e'[\gamma'(t)] \rightsquigarrow \mathbf{tt}$. Instantiating the assumption that t CIU-approximates t' with γ' and e' we get $e'[\gamma'(t')] \rightsquigarrow \mathbf{tt}$.

□

The importance of this proposition is that it establishes that both the CIU-approximation and logical approximation are precongruences. Indeed, CIU-approximation is obviously adequate, reflexive and transitive but it is not obviously compatible. On the other hand, Proposition 15 shows that logical approximation is compatible, which implies that is reflexive, but transitivity is not immediate from the definition and in fact the direct proof quickly fails. Establishing that the two approximation relations coincide shows that both are adequate precongruences.

Theorem 18. *CIU, contextual and logical approximation relations coincide.*

Proof. Proposition 3 and Proposition 17 imply that CIU-approximation and logical approximation imply contextual approximation, so we only need to establish that contextual approximation implies CIU-approximation. To that end let $\Gamma = (x_i : \tau_i)_{i=1}^n$ and assume $\Gamma \vdash t \leq_{ctx} t' : \sigma$, let $\gamma = [x_i \mapsto s_i]_{i=1}^n$ be a closing substitution for t and t' and suppose $e[\gamma(t)] \rightsquigarrow \mathbf{tt}$. This implies that

$$e\left[((((\lambda x_1 \,.\, \lambda x_2 \,.\, \ldots \lambda x_n \,.\, t)\ s_1)\ s_2)\ \cdots)\ s_n\right] \rightsquigarrow \mathbf{tt} \tag{1}$$

and it is easy to see that $e\left[((((\lambda x_1 \,.\, \lambda x_2 \,.\, \ldots \lambda x_n \,.\, -)\ s_1)\ s_2)\ \cdots)\ s_n\right] \in \mathbb{C}(\Gamma, \sigma \rhd \mathbf{2})$.

We instantiate $\Gamma \vdash t \leq_{ctx} t' : \sigma$ with the above context and using (1) to get

$$e\,[((((\lambda\, x_1\,.\,\lambda\, x_2\,.\,\ldots\lambda\, x_n\,.\,t')\;s_1)\;s_2)\;\cdots)\;s_n] \rightsquigarrow \mathbf{tt}$$

which implies $e\,[\gamma(t')] \rightsquigarrow \mathbf{tt}$. □

The coincidence of these relations enables us to easily prove some simple properties which we will use for establishing properties of the categorical model.

Lemma 19. *Let $b, c \in \mathcal{T}_\sigma$. If $b \rightsquigarrow c$ then $\cdot \vdash b \equiv_{ciu} c : \sigma$.*

Lemma 20. *Let $e \in \mathbb{G}\,(\sigma)$, $a \in \mathcal{T}_{\tau\Rightarrow\sigma}$ and $b, c \in \mathcal{T}_\tau$. If $b \rightsquigarrow c$ then $e[a\,b] \rightsquigarrow \mathbf{tt}$ if and only if $e[a\,c] \rightsquigarrow \mathbf{tt}$.*

Proof. Both directions follow from the same argument. From Lemma 19 we have $\cdot \vdash b \leq_{ciu} c : \tau$ and $\cdot \vdash a \leq_{ciu} a : \tau \Rightarrow \sigma$. Proposition 17 implies that CIU-approximation satisfies compatibility properties so $\cdot \vdash a\,b \leq_{ciu} a\,c : \sigma$. Instantiating this assumption with e and empty substitutions concludes the proof. □

Proposition 21. *If $\Gamma, x : \sigma \vdash t \leq_{ctx} t' : \tau$ and $s \in \mathbb{T}\,(\Gamma \rhd \sigma)$ then $\Gamma \vdash t\,[s/x] \leq_{ctx} t'\,[s/x] : \tau$*

Proof. CIU-approximation is trivially seen to have this property. We therefore use Theorem 18 to conclude the proof. □

Proposition 22. *If $\Gamma \vdash s \leq_{ciu} s' : \sigma$ and $\Gamma, x : \sigma \vdash t : \tau$ then $\Gamma \vdash t\,[s/x] \leq_{ciu} t\,[s'/x] : \tau$.*

Proof. Take a closing substitution γ for Γ, $e \in \mathbb{G}\,(\tau)$ and assume $e\left[\gamma\left(t\,[s/x]\right)\right] \rightsquigarrow \mathbf{tt}$. As γ is a closing substitution we have $e\left[\gamma\left(t\,[s/x]\right)\right] = e\left[\gamma(t)\,[\gamma(s)/x]\right]$ so we have $e\,[(\lambda\, x\,.\,\gamma(t))(\gamma(s))] \rightsquigarrow \mathbf{tt}$. By Proposition 21 $\cdot \vdash \gamma(s) \leq_{ctx} \gamma(s') : \sigma$ and since $e\,[(\lambda\, x\,.\,\gamma(t))\;-] \in \mathbb{C}\,(\cdot, \sigma \rhd \mathbf{2})$ the proof is done. □

To finish this section we give a simple concrete application of the development to establish the extensionality for terms of the function type.

Proposition 23. *If $f, f' \in \mathcal{T}_{\sigma\Rightarrow\tau}$ and $\forall c \in \mathcal{T}_\sigma, \cdot \vdash f\,c \leq_{ctx} f'\,c : \tau$ then $\cdot \vdash f \leq_{ctx} f' : \sigma \Rightarrow \tau$.*

Proof. We will show that f CIU-approximates f'. Let $e \in \mathbb{G}\,(\sigma \Rightarrow \tau)$ and assume $e[f] \rightsquigarrow \mathbf{tt}$. Lemma 5 shows that $e = e'\,[-\,d]$ for some $d \in \mathcal{T}_\sigma$ and $e \in \mathbb{G}\,(\tau)$. Instantiating the assumption $\forall c \in \mathcal{T}_\sigma, \cdot \vdash f\,c \leq_{ctx} f'\,c : \tau$ with d shows $e'[f'\,d] \rightsquigarrow \mathbf{tt}$ and thus $e[f'] \rightsquigarrow \mathbf{tt}$, as required. □

4 The model

We define a categorical model of the language and show that it is sound, adequate and complete with respect to the operational notion of observation at base type. Soundness in this case means that the model validates the basic equations arising from the reduction relation $\rightarrow$, adequacy means, roughly, that if two terms have equal denotations then they are contextually equivalent and completeness (or full abstraction) means that two contextually equivalent terms have equal denotations.

Showing that the category is cartesian closed essentially amounts to showing β and η laws for functions and products, which are relatively easy to establish using the logical relation. Technically, some care is required due to the handling of contexts. It is usual in simple categorical models to interpret terms as morphisms from the interpretation of the context to the interpretation of the type, where the interpretation of the context is the product, in the category, of the interpretations of the types. What this amounts to is substituting projections from a single variable for individual variables and to account for this tupleing properly requires some care.

Let $\mathcal{C}$ be a category with the set of objects $\mathcal{C}_0$. Morphisms from X_τ to Y_σ are equivalence classes of a certain partial equivalence relation on the set $\mathbb{T}(x : \tau \rhd \sigma)$ which we now define. To do this we first need a few auxiliary definitions.

Given $X_\tau, Y_\sigma \in \mathcal{C}_0$ we define the binary relation $\prec_{X_\tau, Y_\sigma}$ on $\mathbb{T}(x : \tau \rhd \sigma)$ exactly as we did the logical approximation relation, the difference being that we only relate terms that have at most one free variable and that the objects are now more general, not just the objects arising as the interpretations of types.

Explicitly, the relation $\prec_{X,Y}$ is defined as follows

$$t \prec_{X,Y} s \overset{\text{def}}{=} \forall n \in \mathbb{N}, \forall (c, c') \in X^{\top\top}(n), (t(c), s(c')) \in Y^{\top\top}(n).$$

For each pair of objects $X, Y \in \mathcal{C}_0$ we then define the relation $\sim_{X,Y}$ as the quasi-reflexive symmetric interior of $\prec_{X,Y}$ and then $\approx_{X,Y}$ as the transitive closure of $\sim_{X,Y}$. Explicitly

$$\begin{aligned} \sim_{X,Y} &\overset{\text{def}}{=} \left\{(t, t') \;\middle|\; t \prec_{X,Y} t \wedge t' \prec_{X,Y} t' \wedge t \prec_{X,Y} t' \wedge t' \prec_{X,Y} t\right\} \\ \approx_{X,Y} &\overset{\text{def}}{=} \bigcup_{n=1}^{\infty} \sim_{X,Y}^{n}. \end{aligned}$$

Denoting the equivalence class of t under $\approx_{X,Y}$ as $[t]_{X,Y}$ we define the morphisms

$$\mathrm{Hom}_{\mathcal{C}}(X, Y) = \left\{[t]_{X,Y} \;\middle|\; t \approx_{X,Y} t\right\}.$$

If $t \in [s] \in \mathrm{Hom}_{\mathcal{C}}(X, Y)$ we say that t *realizes* a morphism from X to Y or that t is a *realizer* for $[s]$.

Composition is by substitution, $[t]; [s] \overset{\text{def}}{=} [s(t)]$, and the identity morphism is the equivalence class of the term consisting just of the variable x.

Rationale for the definition of morphisms We now explain the rationale behind the definition of morphisms, specifically the use of quasi-reflexive symmetric interior and the transitive closure. We explain this by showing how the proof that composition is well-defined proceeds.

It is not immediately clear that composition is well-defined. For it to be well-defined means that if $t \approx_{X,Y} t'$ and $s \approx_{Y,Z} s'$ then $s(t) \approx_{X,Z} s'(t')$. Using the definition of transitive closure we have to prove that if

$$t \sim_{X,Y} t_1 \sim_{X,Y} t_2 \sim_{X,Y} \cdots \sim_{X,Y} t_{\mathbf{n}} \sim_{X,Y} t' \tag{2}$$

and

$$s \sim_{Y,Z} s_1 \sim_{Y,Z} s_2 \sim_{Y,Z} \cdots \sim_{Y,Z} s_{\mathbf{m}} \sim_{Y,Z} s' \tag{3}$$

then there exists a k and $r_1, r_2, \ldots, r_k$ such that

$$s(t) \sim_{X,Z} r_1 \sim_{X,Z} r_2 \sim_{X,Z} \cdots \sim_{X,Z} s'(t') \tag{4}$$

and the important thing to notice here is that n and m are in general not equal. It is easy to see that if $u \prec_{X,Y} u'$ and $v \prec_{Y,Z} v'$ then $v(u) \prec_{X,Z} v'(u')$ and it is then immediate that the same holds for the $\sim$ relation.

If n and m in (2) and (3) were equal, we could thus easily take $k = n$ and $r_i = s_i(t_i)$, but in general we cannot expect this to be the case, if the underlying relation is not well-behaved.

Taking the transitive closure of only the quasi-reflexive symmetric interior of the original approximation relation $\prec$ is what ensures that we can always extend the chains to be of the same length. In other words, it ensures that $\sim^n \subseteq \sim^{n+1}$ (see Lemma 24 for why this is the case).

In the same way we can transfer other compatibility results that hold for the $\prec$ relation to the $\approx$ relation and thus to constructions on morphisms. For example, if we knew that $\prec$ was compatible for pairing, meaning that if $t \prec_{X,Y} t'$ and $s \prec_{X,Z} s'$ then $\langle t, s\rangle \prec_{X,Y\times Z} \langle t', s'\rangle$ we can, in the same way as we did above for composition, prove that $t \approx_{X,Y} t'$ and $s \approx_{X,Z} s'$ then $\langle t, s\rangle \approx_{X,Y\times Z} \langle t', s'\rangle$.

We will use this property of the relations to only show properties of the $\prec$ relation, relying on the fact that we can transport results to the $\approx$, as we outlined in the discussion above. See [5] for more details on this construction.

The fact that composition is associative is a consequence of associativity of substitution.

In the rest of this tutorial we will, when it will not cause confusion, omit explicit typing requirements and just implicitly assume that when we talk about realizers for morphisms then they have the appropriate type.

The following two lemmas expose simple, yet constantly used properties. The first lemma crucially depends on the fact that we have taken the transitive closure of only the *quasi-reflexive symmetric* interior of the relation.

Lemma 24. *Let $X, Y \in \mathcal{C}_0$ and t realize a morphism from X to Y. We have $[t] \in \mathrm{Hom}_{\mathcal{C}}(X, Y) \iff t \prec_{X,Y} t$.*

Proof. It is obvious that if $t \prec_{X,Y} t$ then $[t] \in \mathrm{Hom}_{\mathcal{C}}(X, Y)$. For the other direction the premise implies there exists a $n \in \mathbb{N}$ and $t_1, t_2, \ldots, t_n \in \mathcal{T}$, such that

$$t \sim_{X,Y} t_1 \sim_{X,Y} t_2 \sim_{X,Y} \cdots \sim_{X,Y} t_n \sim_{X,Y} t.$$

It follows from the definition of $\sim_{X,Y}$ that if $t \sim_{X,Y} t_1$ then $t \prec_{X,Y} t$ which concludes the proof. $\square$

We now show that the category $\mathcal{C}$ is cartesian closed with fixed points of all endomorphisms.

Let $\top_{\mu\,\alpha.\alpha} \in \mathcal{C}_0$ be the everywhere empty relation; $\top_{\mu\,\alpha.\alpha}(n) \overset{\text{def}}{=} \emptyset$

Proposition 25. *$\top$ is the terminal object in $\mathcal{C}$. The unique morphism from any object X to $\top$ is given by the equivalence class of* $\Omega = (\lambda\, x\,.\; \mathbf{unfold}\,(x)\;x)\,\big(\mathbf{fold}\,(\lambda\, x\,.\;\mathbf{unfold}\,(x)\;x)\big)$.

Proof. It is clear that $\top \in \mathcal{C}_0$ and we immediately observe that for all $n \in \mathbb{N}$, $\top\downarrow(n) = \mathbb{G}\,(\mu\,\alpha.\alpha) \times \mathbb{G}\,(\mu\,\alpha.\alpha)$.

Pick an object X. We first have to show $\Omega \prec_{X,\top} \Omega$ and this is simple; take $m \in \mathbb{N}$ and $(c,c') \in X^{\top\top}(n)$. Then $\Omega(c) = \Omega(c') = \Omega$ and as $\Omega \rightarrow_1 \Omega$, the term Ω diverges. Using Lemma 10 we have our result. Now suppose $\varphi \prec_{X,\top} \varphi$ is another such morphism. We claim that $\forall n \in \mathbb{N}, \forall (c,c') \in X^{\top\top}(n), \forall e \in \mathbb{G}\,(\mu\,\alpha.\alpha)$, $e[\varphi(c)]$ must not converge to $\mathbf{tt}$. Suppose it does for some e. Then, as $\top\downarrow(n) = \mathbb{G}\,(\mu\,\alpha.\alpha) \times \mathbb{G}\,(\mu\,\alpha.\alpha)$, for any evaluation context $e' \in \mathbb{G}\,(\mu\,\alpha.\alpha)$, $e'[\varphi(c')]$ must converge to $\mathbf{tt}$, but this is clearly nonsense. We thus use Lemma 10 to get $\varphi \prec_{X,\top} \Omega$ and the other direction also follows using the same reasoning. We thus have $[\varphi] = [\Omega]$ and therefore uniqueness. □

Interpretation of 2 The following two properties will be used when showing that the interpretation of the ground type **2** is sound.

Lemma 26. *If $\varphi \prec_{X,Y} \varphi'$, $\psi \prec_{X,Y} \psi'$ and $\delta \prec_{X,\mathbb{B}} \delta'$ then* $\mathbf{if}\,(\delta,\varphi,\psi) \prec_{X,Y} \mathbf{if}\,(\delta',\varphi',\psi')$.

Proof. A simple corollary of Lemma 11. □

Proposition 27. *If $[\varphi], [\psi] \in \mathrm{Hom}_{\mathcal{C}}\,(X,Y)$ then $[\mathbf{if}\,(\mathbf{tt},\varphi,\psi)] \in \mathrm{Hom}_{\mathcal{C}}\,(X,Y)$ and $[\mathbf{if}\,(\mathbf{tt},\varphi,\psi)] = [\varphi]$ and $[\mathbf{if}\,(\mathbf{ff},\varphi,\psi)] \in \mathrm{Hom}_{\mathcal{C}}\,(X,Y)$ and $[\mathbf{if}\,(\mathbf{ff},\varphi,\psi)] = [\psi]$.*

Proof. We will only prove the cases for $\mathbf{tt}$, the case for $\mathbf{ff}$ being completely symmetric.

Take $n \in \mathbb{N}$, $(c,c') \in X^{\top\top}(n)$, $i \le n$, $(e,e') \in Y\downarrow(i)$ and assume $e[\mathbf{if}\,(\mathbf{tt},\varphi(c),\psi(c))] \overset{i}{\rightsquigarrow} \mathbf{tt}$. Then $e[\varphi(c)] \rightsquigarrow \mathbf{tt}$ and so $e'\,[\varphi(c')] \rightsquigarrow \mathbf{tt}$.

For the other direction take $n \in \mathbb{N}$, $(c,c') \in X^{\top\top}(n)$, $i \le n$, $(e,e') \in Y\downarrow(i)$ and assume $e[\varphi(c)] \overset{i}{\rightsquigarrow} \mathbf{tt}$. Then $e'\,[\varphi(c')] \rightsquigarrow \mathbf{tt}$ and so $e'\,[\mathbf{if}\,(\mathbf{tt},\varphi(c'),\psi(c'))] \rightsquigarrow \mathbf{tt}$. □

The last proposition expresses that **2** is a weak sum of $\top$ and $\top$.

Products We now prove that the category has binary products.

Proposition 28. *For $X, Y \in \mathcal{C}_0$, $X \times Y$ is the product object. Projections are morphisms realized by* **fst** *and* **snd** *and tupleing of morphisms realized by φ and ψ is a morphism realized by $\langle\varphi,\psi\rangle$.*

We will need the following lemma in the proof of the proposition.

Lemma 29. *Let $(e, e') \in (X \times Y) \downarrow (n)$ for some n. Then either there exists an $f \in \mathcal{E}$, such that $e = f[\mathbf{fst} -]$ or there exists an f, such that $e = f[\mathbf{snd} -]$ and similarly for e'.*

Proof. Immediate consequence of Lemma 4. □

Note that this lemma (among others) would not be true, were we to make observations at the product type, for example if we just observed termination at any type. This is an essential difference with the call-by-value language, where observing termination or observing termination at a particular type or observing termination to a particular value of the base type makes no difference, because in a call-by-value language there are many more evaluation contexts, i.e. more test functions.

Proof (Proposition 28). Lemma 12 tells us that $\mathbf{fst}\, x \prec_{X \times Y, X} \mathbf{fst}\, x$ and $\mathbf{snd}\, x \prec_{X \times Y, Y} \mathbf{snd}\, x$ hold and so $\mathbf{fst}\, x$ and $\mathbf{snd}\, x$ are well defined morphisms.

We now show that for any object Z and $\varphi \prec_{Z, X \times Y} \varphi$

$$\langle \mathbf{fst}\, \varphi, \mathbf{snd}\, \varphi \rangle \prec_{Z, X \times Y} \varphi$$

and

$$\varphi \prec_{Z, X \times Y} \langle \mathbf{fst}\, \varphi, \mathbf{snd}\, \varphi \rangle .$$

So assume such φ and take $n \in \mathbb{N}, (c, c') \in Z^{\top\top}(n), i \leq n, (e, e') \in X \times Y \downarrow (i)$ and assume $e[\langle \mathbf{fst}\, \varphi(c), \mathbf{snd}\, \varphi(c) \rangle] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$. Lemma 29 implies e must be of the form $e = f[\mathbf{fst} -]$ or $e = f[\mathbf{snd} -]$. In either case, we have $e[\varphi(c)] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$ and so $e'[\varphi(c')] \rightsquigarrow \mathbf{tt}$.

For the second one, take the same n, c, c' and i, e, e' and assume $e[\varphi(c)] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$. As $\varphi \prec_{Z, X \times Y} \varphi$, we immediately have $e'[\varphi(c')] \rightsquigarrow \mathbf{tt}$. Lemma 29 tells us that $e' = f'[\mathbf{fst} -]$ or $e' = f'[\mathbf{snd} -]$ for some f' and as $f'\, [\mathbf{fst}\, \langle \mathbf{fst}\, \varphi(c'), \mathbf{snd}\, \varphi(c') \rangle] \rightarrow f'[\mathbf{fst}\, \varphi(c')] = e'[\varphi(c')] \rightsquigarrow \mathbf{tt}$ and similarly for $\mathbf{snd}$, we are done.

We only have to show that tupleing respects the equivalence relation. So take $\varphi \prec_{Z,X} \varphi'$ and $\psi \prec_{Z,Y} \psi'$ so we need to show $\langle \varphi, \psi \rangle \prec_{Z, X \times Y} \langle \varphi', \psi' \rangle$. Take $n \in \mathbb{N}, (c, c') \in Z^{\top\top}(n), i \leq n, (e, e') \in (X \times Y) \downarrow (n)$ and assume $e\, [\langle \varphi(c), \psi(c) \rangle] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$. Using the assumptions $\varphi \prec_{Z,X} \varphi'$ and $\psi \prec_{Z,Y} \psi'$ we get $(\varphi(c), \varphi'(c')) \in X^{\top\top}(n)$ and $(\psi(c), \psi'(c'))\, Y^{\top\top}(n)$ and so

$$(\langle \varphi(c), \psi(c) \rangle, \langle \varphi'(c'), \psi'(c') \rangle) \in (X \times Y)(n) \subseteq (X \times Y)^{\top\top}(n),$$

so we have obtained the necessary property almost trivially.

We deal with transitivity as we have outlined in Section 4 to show

- if $[\varphi] = [\varphi']$ and $[\psi] = [\psi']$ then $[\langle \varphi, \psi \rangle] = [\langle \varphi', \psi' \rangle]$
- if $[\xi]; [\mathbf{fst}\, x] = [\varphi]$ and $[\xi]; [\mathbf{snd}\, x] = [\psi]$ then $[\xi] = [\langle \varphi, \psi \rangle]$.

which establishes the universal property of products. □

Exponentials We are now ready to prove that the category $\mathcal{C}$ has exponentials and is cartesian closed. Most of the proofs are straightforward, only the uniqueness of transposes requires some work.

Proposition 30. *For $X, Y \in \mathcal{C}_0$, Y^X is the exponential object. Evaluation is given by the equivalence class of $\varepsilon = (\mathbf{fst}\, x)\,(\mathbf{snd}\, x)$ and the transpose of a morphism from $Z \times X$ to Y realized by φ is given by the morphism realized by $\Lambda\,(\varphi) = \lambda\, y\,.\,\varphi(\langle x, y\rangle)$.*

We first prove some auxiliary lemmas.

Lemma 31. *Let $n \in \mathbb{N}$ and $(c, c') \in (X \times Y)^{\top\top}(n)$. For all $d \in \mathcal{T}$,*

$$(c, \langle \mathbf{fst}\, c', \mathbf{snd}\,\langle d, \mathbf{snd}\, c'\rangle\rangle) \in (X \times Y)^{\top\top}(n).$$

Proof. Take $i \leq n$, $(e, e') \in (X \times Y) \downarrow (i)$ and assume $e[c] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$. Lemma 29 tells us that e' must be of the form $e' = f'[\mathbf{fst} -]$ or $e' = f'[\mathbf{snd} -]$. In both cases we get what is required by a simple inspection. $\square$

Lemma 32. *Let $n \in \mathbb{N}$, $(e, e') \in \left(Y^X\right) \downarrow (n)$. Then there exist evaluation contexts f, f' and closed terms d, d', such that $e = f[-d]$ and $e' = f'[-d']$.*

Proof. This is an immediate consequence of Lemma 5. $\square$

Proof (Proposition 30). First we show that ε is well defined. Let $n \in \mathbb{N}, (c, c') \in \left(Y^X \times X\right)^{\top\top}(n)$. Lemma 12 gives us that $(\mathbf{fst}\, c, \mathbf{fst}\, c') \in \left(Y^X\right)^{\top\top}(n)$ and $(\mathbf{snd}\, c, \mathbf{snd}\, c') \in X^{\top\top}(n)$ and then we use Lemma 13 to get what we need.

Next we show that transposes are well defined, so assume $\varphi \prec_{Z\times X, Y} \psi$ and we need to show $\Lambda\,(\varphi) \prec_{Z, Y^X} \Lambda\,(\psi)$. So take $n \in \mathbb{N}, (c, c') \in Z^{\top\top}(n)$. We will show directly, that $(\Lambda\,(\varphi)\,(c), \Lambda\,(\psi)\, c') \in \left(Y^X\right)(n)$, which is more than we need to prove. By construction and the fact that c and c' are closed terms, we have $\Lambda\,(\varphi)\,(c) = \lambda\, y\,.\,\varphi\,(\langle c, y\rangle)$ and $\Lambda\,(\psi)\,(c') = \lambda\, y\,.\,\psi\,(\langle c', y\rangle)$, so the terms are of the right form. To show that they are related at step n take $i \leq n$, $(d, d') \in X^{\top\top}(i)$ and $(e, e') \in Y \downarrow (i)$ and assume $e\,[(\lambda\, y\,.\,\varphi\,(\langle c, y\rangle))\; d] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$. This also means that $e\,[\varphi\,(\langle c, d\rangle)] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$ and using the fact that $(\langle c, d\rangle, \langle c', d'\rangle) \in (Z \times X)(i) \subseteq (Z \times X)^{\top\top}(i)$ and $\varphi \prec_{Z\times X, Y} \psi$ we also have $(\varphi\,(\langle c, d\rangle), \psi\,(\langle c', d'\rangle)) \in Y^{\top\top}(i)$ and so $e'\,[\psi\,(\langle c', d'\rangle)] \rightsquigarrow \mathbf{tt}$ and as $e'\,[(\lambda\, y\,.\,\psi\,(\langle c', y\rangle))\; d'] \rightarrow e'\,[\psi\,(\langle c', d'\rangle)] \rightsquigarrow \mathbf{tt}$ we are done.

We are thus only left with showing the universal property and we do this by showing that **eval** induces an inverse to $\Lambda\,(\cdot)$.

Let φ realize a morphism from $Z \times X$ to Y. We first show that

$$(\mathbf{fst}\,(\langle \Lambda\,(\varphi)\,(\mathbf{fst}\, x)\,, \mathbf{snd}\, x\rangle))\;(\mathbf{snd}(\langle \Lambda\,(\varphi\,(\mathbf{fst}\, x))\,, \mathbf{snd}\, x\rangle)) \prec_{Z\times X, Y} \varphi$$

So take $n \in \mathbb{N}, (c, c') \in (Z \times X)^{\top\top}(n), i \leq n, (e, e') \in Y \downarrow (i)$ and assume

$$e\,[(\mathbf{fst}\,(\langle \Lambda\,(\varphi)\,(\mathbf{fst}\, c)\,, \mathbf{snd}\, c\rangle))\;(\mathbf{snd}(\langle \Lambda\,(\varphi\,(\mathbf{fst}\, c))\,, \mathbf{snd}\, c\rangle))] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$$

Simplifying a bit, this implies that

$$e[\varphi\left(\langle \mathbf{fst}\, c, \mathbf{snd}(\langle \Lambda\left(\varphi\left(\mathbf{fst}\, c\right)\right), \mathbf{snd}\, c\rangle)\rangle\right)] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$$

Using the fact that $\left(\langle \mathbf{fst}\, c, \mathbf{snd}(\langle \Lambda\left(\varphi\left(\mathbf{fst}\, c\right)\right), \mathbf{snd}\, c\rangle)\rangle, \langle \mathbf{fst}\, c', \mathbf{snd}\, c'\rangle\right) \in (Z \times X)(i)$, which follows from Lemma 8 and Lemma 12, and the fact that φ is related to itself, we get $e'\left[\varphi\left(\langle \mathbf{fst}\, c', \mathbf{snd}\, c'\rangle\right)\right] \rightsquigarrow \mathbf{tt}$. Using Lemma 4 it follows that $\cdot \vdash \langle \mathbf{fst}\, c', \mathbf{snd}\, c'\rangle \equiv_{ciu} c' : \tau_Z \otimes \tau_X$, where τ_Z and τ_X are types associated with the objects Z and X, respectively. Proposition 22 thus shows that $\cdot \vdash \varphi\left(\langle \mathbf{fst}\, c', \mathbf{snd}\, c'\rangle\right) \equiv_{ciu} \varphi(c') : \tau_Y$ which implies $f'\left[\varphi(c')\right] \rightsquigarrow \mathbf{tt}$.

For the other direction we use Lemma 31. Take the same n, c, c' and i, e, e' and assume $e[\varphi(c)] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$. Then the lemma tells us that $e'\left[\varphi\left(\langle \mathbf{fst}\, c', \mathbf{snd}\,\langle \Lambda\left(\varphi\left(\mathbf{fst}\, c'\right)\right), c'\rangle\rangle\right)\right] \rightsquigarrow \mathbf{tt}$ which is, as we have seen before enough. This implies that transposing has a post-inverse and that it maps morphisms appropriately.

Suppose now that φ realizes a morphism from Z to Y^X. We will show

$$\varphi \prec_{Z,Y^X} \Lambda\left(\varepsilon\left(\langle \varphi(\mathbf{fst}\, x), \mathbf{snd}\, x\rangle\right)\right)$$

and

$$\Lambda\left(\varepsilon\left(\langle \varphi(\mathbf{fst}\, x), \mathbf{snd}\, x\rangle\right)\right) \prec_{Z,Y^X} \varphi$$

one at a time.

So take $n \in \mathbb{N}, (c, c') \in Z^{\top\top}(n), i \leq n$ and $(e, e') \in \left(Y^X\right) \downarrow (i)$ and assume $e[\varphi(c)] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$. Using Lemma 32 we have $e = f[-d]$ and $e' = f'[-d']$ for some $f, f' \in \mathcal{E}$ and $d, d' \in \mathcal{T}$. We then have

$$\begin{aligned} e'\left[\Lambda\left(\varepsilon\left(\langle \varphi(\mathbf{fst}\, x), \mathbf{snd}\, x\rangle\right)\right)(c')\right] &\rightsquigarrow f'\left[\varepsilon\left(\langle \varphi(\mathbf{fst}\, x), \mathbf{snd}\, x\rangle\right)\left(\langle c', d'\rangle\right)\right] \\ &= f'\left[\varepsilon\left(\langle \varphi(\mathbf{fst}\,\langle c', d'\rangle), \mathbf{snd}\,\langle c', d'\rangle\rangle\right)\right] \\ &\rightsquigarrow f'\left[\left(\varphi(\mathbf{fst}\,\langle c', d'\rangle)\right)\left(\mathbf{snd}\,\langle \varphi(\mathbf{fst}\,\langle c', d'\rangle), \mathbf{snd}\,\langle c', d'\rangle\rangle\right)\right]. \end{aligned}$$

Lemma 8 tells us that $\left(\varphi(c), \varphi(\mathbf{fst}\,\langle c', d'\rangle)\right) \in \left(Y^X\right)^{\top\top}(n)$ and obviously

$$\mathbf{snd}\,\langle \varphi(\mathbf{fst}\,\langle c', d'\rangle), \mathbf{snd}\,\langle c', d'\rangle\rangle \rightsquigarrow d'$$

due to the call by name nature of the evaluation relation. Using Lemma 20 we have that if $f'\left[\left(\varphi(\mathbf{fst}\,\langle c', d'\rangle)\right) d'\right] \rightsquigarrow \mathbf{tt}$ then also $f'\left[\left(\varphi(\mathbf{fst}\,\langle c', d'\rangle)\right)\left(\mathbf{snd}\,\langle \varphi(\mathbf{fst}\,\langle c', d'\rangle), \mathbf{snd}\,\langle c', d'\rangle\rangle\right)\right] \rightsquigarrow \mathbf{tt}$ and the first one reduces to $\mathbf{tt}$ because, as mentioned above, $\left(\varphi(c), \varphi(\mathbf{fst}\,\langle c', d'\rangle)\right) \in \left(Y^X\right)^{\top\top}(n)$.

For the other approximation again take $n \in \mathbb{N}, (c, c') \in Z^{\top\top}(n), i \leq n$ and $(e, e') \in \left(Y^X\right) \downarrow (i)$ and now assume that $e[\Lambda\left(\varepsilon\left(\langle \varphi(\mathbf{fst}\, x), \mathbf{snd}\, x\rangle\right)\right)] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$. Again, there exist f, f', d and d' such that $e = f[-d]$ and $e' = f'[-\,d']$. The last assumption thus implies

$$f\left[\left(\varphi(\mathbf{fst}\,\langle c, d\rangle)\right)\left(\mathbf{snd}\,\langle \varphi(\mathbf{fst}\,\langle c, d\rangle), \mathbf{snd}\,\langle c, d\rangle\rangle\right)\right] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$$

which further implies

$$f\,[(\lambda\, x\,.\ (\varphi(\mathbf{fst}\,\langle c,x\rangle))\,(\mathbf{snd}\,\langle\varphi(\mathbf{fst}\,\langle c,x\rangle),\mathbf{snd}\,\langle c,x\rangle\rangle))\ d] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$$

If we can show that

$$\begin{aligned} \big(\,\lambda\, x\,.\ (\varphi(\mathbf{fst}\,\langle c,x\rangle))\,(\mathbf{snd}\,\langle\varphi(\mathbf{fst}\,\langle c,x\rangle),\mathbf{snd}\,\langle c,x\rangle\rangle)\,, \qquad (5)\\ \lambda\, x\,.\ (\varphi(\mathbf{fst}\,\langle c',x\rangle))\,(\mathbf{snd}\,\langle\varphi(\mathbf{fst}\,\langle c',x\rangle),\mathbf{snd}\,\langle c',x\rangle\rangle)\,\big) \in \big(Y^X\big)\,(n) \qquad (6)\end{aligned}$$

we can conclude

$$\begin{aligned} f'\,[(\lambda\, x\,.\ (\varphi(\mathbf{fst}\,\langle c',x\rangle))\,(\mathbf{snd}\,\langle\varphi(\mathbf{fst}\,\langle c',x\rangle),\mathbf{snd}\,\langle c',x\rangle\rangle))\ d'] \rightsquigarrow \mathbf{tt}\\ f'\ [(\varphi(\mathbf{fst}\,\langle c',d'\rangle))\,(\mathbf{snd}\,\langle\varphi(\mathbf{fst}\,\langle c',d'\rangle),\mathbf{snd}\,\langle c',d'\rangle\rangle)] \rightsquigarrow \mathbf{tt}\end{aligned}$$

which by Lemma 20 implies

$$e'\,[\varphi(\mathbf{fst}\,\langle c',d'\rangle)] \rightsquigarrow \mathbf{tt}$$

but what we need to conclude is

$$e'\,[\varphi(c')] \rightsquigarrow \mathbf{tt}\,.$$

It is easy to see that $\cdot \vdash \mathbf{fst}\,\langle c',d'\rangle \leq_{ciu} c' : \tau_Z$ and using Proposition 22 we get exactly what is required.

We thus only have to show that (5) holds. To this end let $i \leq n, (d,d') \in X^{\top\top}(i), (e,e') \in Y\downarrow(i)$ and assume

$$e\,[(\lambda\, x\,.\ (\varphi(\mathbf{fst}\,\langle c,x\rangle))\,(\mathbf{snd}\,\langle\varphi(\mathbf{fst}\,\langle c,x\rangle),\mathbf{snd}\,\langle c,x\rangle\rangle))\ d] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}\,.$$

This again implies

$$e\,[(\varphi(\mathbf{fst}\,\langle c,d\rangle))\,(\mathbf{snd}\,\langle\varphi(\mathbf{fst}\,\langle c,d\rangle),\mathbf{snd}\,\langle c,d\rangle\rangle)] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}\,.$$

Lemma 8 implies

$$(\mathbf{snd}\,\langle\varphi(\mathbf{fst}\,\langle c,d\rangle),\mathbf{snd}\,\langle c,d\rangle\rangle\,,\mathbf{snd}\,\langle\varphi(\mathbf{fst}\,\langle c',d'\rangle),\mathbf{snd}\,\langle c',d'\rangle\rangle) \in X^{\top\top}(i)$$

and also $(\mathbf{fst}\,cd, \mathbf{fst}\,c'd') \in Z^{\top\top}(n)$. As $\varphi \prec_{Z,Y^X} \varphi$ we get $(\varphi\,(\mathbf{fst}\,\langle c,d\rangle)\,,\varphi\,(\mathbf{fst}\,\langle c',d'\rangle)) \in \big(Y^X\big)^{\top\top}(n)$. Lemma 13 then implies

$$e'\,[(\varphi(\mathbf{fst}\,\langle c',d'\rangle))\,(\mathbf{snd}\,\langle\varphi(\mathbf{fst}\,\langle c',d'\rangle),\mathbf{snd}\,\langle c',d'\rangle\rangle)] \rightsquigarrow \mathbf{tt}\,.$$

which implies the required goal.

This shows that transposes are unique and that for each X the object mapping $Y \mapsto X^Y$ can be extended to a functor with a left adjoint, so the category is cartesian closed. □

Fixed points To model general recursion we define a fixed point combinator as

$$Y(f) = (\lambda\, z\, .\, f\ (\mathbf{unfold}\,(z)\ z))\ \big(\mathbf{fold}\,((\lambda\, z\, .\ (f\ (\mathbf{unfold}\,(z)\ z))))\big)$$

for any term f. In the special case where x is the only possible free variable in f, we write $\mathcal{F}(f) = Y\,(\lambda\, x\, .\, f)$

Lemma 33. *If $X \in \mathcal{C}_0$ and $\varphi \prec_{X,X} \psi$ then $\forall n \in \mathbb{N}, (\mathcal{F}(\varphi), \mathcal{F}(\psi)) \in X^{\top\top}(n)$.*

Proof. We use well-founded induction to prove this use.

So take $n \in \mathbb{N}$ and assume $\forall k < n, (\mathcal{F}(\varphi), \mathcal{F}(\psi)) \in X^{\top\top}(k)$. Take $i \leq n$, $(e, e') \in X \downarrow (i)$ and assume $e[\mathcal{F}(\varphi)] \overset{\mathrm{i}}{\rightsquigarrow} \mathbf{tt}$. By construction we have

$$\mathcal{F}(\varphi) \rightsquigarrow \varphi \left(\mathbf{unfold}\,(\mathbf{fold}\,((\lambda\, z\, .\, (\lambda\, x\, .\, \varphi)\ (\mathbf{unfold}\,(z)\ z))))\ \big(\mathbf{fold}\,((\lambda\, z\, .\ ((\lambda\, x\, .\, \varphi)\ (\mathbf{unfold}\,(z)\ z))))\big)\right)$$

and

$$\mathbf{unfold}\,(\mathbf{fold}\,((\lambda\, z\, .\, (\lambda\, x\, .\, \varphi)\ (\mathbf{unfold}\,(z)\ z))))\ \big(\mathbf{fold}\,((\lambda\, z\, .\ ((\lambda\, x\, .\, \varphi)\ (\mathbf{unfold}\,(z)\ z))))\big) \to_1 \mathcal{F}(\varphi).$$

By the induction hypothesis $(\mathcal{F}(\varphi), \mathcal{F}(\psi)) \in X^{\top\top}(i-1)$ and so using Lemma 9 and the fact that

$$\mathbf{unfold}\,(\mathbf{fold}\,((\lambda\, z\, .\, (\lambda\, x\, .\, \psi)\ (\mathbf{unfold}\,(z)\ z))))\ \big(\mathbf{fold}\,((\lambda\, z\, .\ ((\lambda\, x\, .\, \psi)\ (\mathbf{unfold}\,(z)\ z))))\big) \rightsquigarrow \mathcal{F}(\psi)$$

we have that

$$e'\left[\psi \left(\mathbf{unfold}\,(\mathbf{fold}\,((\lambda\, z\, .\, (\lambda\, x\, .\, \psi)\ (\mathbf{unfold}\,(z)\ z))))\ \big(\mathbf{fold}\,((\lambda\, z\, .\ ((\lambda\, x\, .\, \psi)\ (\mathbf{unfold}\,(z)\ z))))\big)\right)\right] \rightsquigarrow \mathbf{tt}$$

and so $e'[\mathcal{F}(\psi)] \rightsquigarrow \mathbf{tt}$. □

Using this lemma we can prove the following.

Corollary 34. *If $X \in \mathcal{C}_0$ and $\varphi \prec_{X,X} \psi$ then $\mathcal{F}(\varphi) \prec_{1,X} \mathcal{F}(\psi)$.*

And finally we can prove that we do indeed have fixed points.

Proposition 35. *If $X \in \mathcal{C}_0$ and $\varphi \prec_{X,X} \varphi$ then $\mathcal{F}(\varphi) \sim_{1,X} \varphi\,(\mathcal{F}(\varphi))$.*

To sum up, we have established the validity of the following theorem.

Theorem 36. *The category $\mathcal{C}$ is a cartesian closed category with fixed points of all endomorphisms.*

4.1 Interpretation and soundness

The interpretation of types is defined in Figure 9. It is exactly the same as the relational interpretation of types used in defining the logical approximation relation in Section 3. In order to define the interpretation of terms we need some additional properties of the interpretation of recursive types.

Recursive types The following proposition establishes that the interpretation of a recursive type and an unfolded recursive type are the same and the isomorphism is given by morphisms realized by **fold** and **unfold**.

Proposition 37. *Let $\mu\,\alpha.\tau$ be a well-formed closed type. Denote $[\![\mu\,\alpha.\tau]\!]$ by X and $[\![\tau\,[\mu\,\alpha.\tau/\alpha]]\!]$ by Y.*

The following hold

$$\mathbf{fold}\,(x) \prec_{Y,X} \mathbf{fold}\,(x) \tag{7}$$
$$\mathbf{unfold}\,(x) \prec_{X,Y} \mathbf{unfold}\,(x) \tag{8}$$
$$\mathbf{fold}\,(\mathbf{unfold}\,(x)) \sim_{X,X} x \tag{9}$$
$$\mathbf{unfold}\,(\mathbf{fold}\,(x)) \sim_{Y,Y} x \tag{10}$$

and so $X \approx Y$.

Proof. The proof follows by simply inspecting the definition of the interpretation of recursive types.

Ad (7) Take $n \in \mathbb{N}$ and $(c, c') \in Y^{\top\top}(n)$. This means $(\mathbf{fold}\,(c)\,, \mathbf{fold}\,(c')) \in X(n+1) \subseteq X(n) \subseteq X^{\top\top}(n)$ so we are done.

Ad (8) Take $n \in \mathbb{N}$, $(c, c') \in X^{\top\top}(n)$, $i \leq n$, $(e, e') \in Y \downarrow (i)$ and assume $e[\mathbf{unfold}\,(c)] \overset{i}{\rightsquigarrow} \mathbf{tt}$. Lemma 14 then shows that also $e'\,[\mathbf{unfold}\,(c')] \rightsquigarrow \mathbf{tt}$, by the now familiar argument.

Ad (9) We only show the approximation $\mathbf{fold}\,(\mathbf{unfold}\,(x)) \preceq_{X,X} x$. The other direction is similar. Let $n \in \mathbb{N}, (c, c') \in X^{\top\top}, i \leq n, (e, e') \in X \downarrow (i)$ and assume $e[\mathbf{fold}\,(\mathbf{unfold}\,(c))] \overset{i}{\rightsquigarrow} \mathbf{tt}$. Lemma 6 implies that there exists an f, such that $e = f[\mathbf{unfold}\,(-)]$. The assumption on termination then implies $f[\mathbf{unfold}\,(c)] \overset{i}{\rightsquigarrow} \mathbf{tt}$ which is the same as saying $e[c] \overset{i}{\rightsquigarrow} \mathbf{tt}$. The conclusion now follows trivially.

Ad (10) This relation follows immediately from Lemma 8. $\square$

Interpretation of terms In order to define the interpretation of terms we first explicitly define projections from the n-fold product $(\cdots(X_1 \times X_2) \times \cdots X_{n-1}) \times X_n$ by induction on n. If $i \leq n$ we define a realizer π_i^n for a morphism $(\cdots(X_1 \times X_2) \times \cdots X_{n-1}) \times X_n \to X_i$ as follows

$$\pi_1^1(x) = x$$
$$\pi_i^{n+1}(x) = \begin{cases} \mathbf{snd}\,x & i = n+1 \\ \pi_i^n\,(\mathbf{fst}\,x) & i \leq n \end{cases}$$

The interpretation in the model and its soundness are straightforward and mostly (except for recursive types) follows from standard facts about interpretation of PCF in a CCC with fixed points. The interpretation of $\Gamma \vdash t : \sigma$ is a

$$\begin{aligned}
\llbracket \Gamma \vdash x_i : \tau_i \rrbracket &= [\pi_i^n(x)] \\
&\quad \text{for } \Gamma = (x_i : \tau_i)_{i=1}^n \\
\llbracket \Gamma \vdash \mathbf{tt} : \mathbf{2} \rrbracket &= [\mathbf{tt}] \\
\llbracket \Gamma \vdash \mathbf{ff} : \mathbf{2} \rrbracket &= [\mathbf{ff}] \\
\llbracket \Gamma \vdash \mathbf{if}\,(t, s, r) : \tau \rrbracket &= [\mathbf{if}\,(\vartheta, \varphi, \psi)] \\
&\quad \text{for } \vartheta \in \llbracket \Gamma \vdash t : \mathbf{2} \rrbracket, \varphi \in \llbracket \Gamma \vdash s : \tau \rrbracket, \psi \in \llbracket \Gamma \vdash r : \tau \rrbracket \\
\llbracket \Gamma \vdash \langle\rangle : \mathbf{1} \rrbracket &= \langle\rangle \\
\llbracket \Gamma \vdash \lambda\, z\,.\,t : \tau \Rightarrow \sigma \rrbracket &= \Lambda\left(\llbracket \Gamma, z : \tau \vdash t : \sigma \rrbracket\right) \\
\llbracket \Gamma \vdash t\, s : \sigma \rrbracket &= \langle \llbracket \Gamma \vdash t : \tau \Rightarrow \sigma \rrbracket, \llbracket \Gamma \vdash s : \tau \rrbracket \rangle\, ; [\varepsilon] \\
\llbracket \Gamma \vdash \langle t, s \rangle : \sigma \otimes \tau \rrbracket &= \langle \llbracket \Gamma \vdash t : \sigma \rrbracket, \llbracket \Gamma \vdash s : \tau \rrbracket \rangle \\
\llbracket \Gamma \vdash \mathbf{fst}\, t : \sigma \rrbracket &= \llbracket \Gamma \vdash t : \sigma \otimes \tau \rrbracket\, ; [\mathbf{fst}\, x] \\
\llbracket \Gamma \vdash \mathbf{snd}\, t : \tau \rrbracket &= \llbracket \Gamma \vdash t : \sigma \otimes \tau \rrbracket\, ; [\mathbf{snd}\, x] \\
\llbracket \Gamma \vdash \mathbf{fold}\,(t) : \mu\, \alpha.\tau \rrbracket &= \llbracket \Gamma \vdash t : \tau\, [\mu\, \alpha.\tau / \alpha] \rrbracket\, ; [\mathbf{fold}\,(x)] \\
\llbracket \Gamma \vdash \mathbf{unfold}\,(t) : \tau\, [\mu\, \alpha.\tau / \alpha] \rrbracket &= \llbracket \Gamma \vdash t : \mu\, \alpha.\tau \rrbracket\, ; [\mathbf{unfold}\,(x)]
\end{aligned}$$

Figure 10: Interpretation of terms

morphism from $\llbracket \Gamma \rrbracket$ to $\llbracket \sigma \rrbracket$, where $\llbracket \Gamma \rrbracket = (\cdots(\llbracket \tau_1 \rrbracket \times \llbracket \tau_2 \rrbracket) \times \cdots \llbracket \tau_{n-1} \rrbracket) \times \llbracket \tau_n \rrbracket$, if $\Gamma = (x_i : \tau_i)_{i=1}^n$. It is spelled out in Figure 10. Soundness of the interpretation of recursive types follows from Proposition 37.

4.2 Adequacy

Lemma 38. *If* $[\varphi] \in \mathrm{Hom}_{\mathcal{C}}(\top, \mathbf{2})$ *and* $[\varphi] = [\mathbf{tt}]$ *then* $\varphi(\Omega) \rightsquigarrow \mathbf{tt}$ *and if* $[\varphi] = [\mathbf{ff}]$ *then* $\varphi(\Omega) \rightsquigarrow \mathbf{ff}$.

Proof. We first observe that $\forall n \in \mathbb{N}, (\Omega, \Omega) \in \top^{\top\top}(n)$ and that $\forall n \in \mathbb{N}, (-,-) \in \mathbf{2}\downarrow(n)$. The case of $\mathbf{tt}$ is now very simple; we will prove the claim by induction on the length of the transitivity chain.

The base case is when $\mathbf{tt} \prec \varphi$. For any $n \geq 1$ we have $\mathbf{tt} \overset{n}{\rightsquigarrow} \mathbf{tt}$ and so $\varphi(\Omega) \rightsquigarrow \mathbf{tt}$.

Otherwise $\mathbf{tt} \prec \varphi_1 \prec \varphi_2 \prec \cdots \prec \varphi_k \prec \varphi_{k+1}$. The induction hypothesis shows that $\varphi_k(\Omega) \rightsquigarrow \mathbf{tt}$ in some number of steps. We then instantiate $\varphi_k \prec \varphi_{k+1}$ with (Ω, Ω) and $(-,-)$ and crucially use the fact that these are related indefinitely, to get $\varphi_{k+1}(\Omega) \rightsquigarrow \mathbf{tt}$.

To show the case for $\mathbf{ff}$, we observe that $\mathbf{if}\,(-, \mathbf{ff}, \mathbf{tt})$ is related to itself as an evaluation context for any n so we proceed in the same way as in the case for $\mathbf{tt}$. □

Lemma 39. *If* $\Gamma \vdash t : \tau$ *and* $\Gamma = (x_i : \tau_i)_{i=1}^n$ *then* $t\,[\pi_i^n x / x_i] \in \llbracket \Gamma \vdash t : \tau \rrbracket$

Proof. We use induction on the typing derivation. The only interesting case is for function abstraction, all the other cases follow straightforwardly from the induction hypothesis

Case $\Gamma \vdash \lambda\, x_{n+1} \,.\, t : \tau \Rightarrow \sigma$ We inductively assume that $t\left[\pi_i^{n+1} x / x_i\right] \in [\![\Gamma, x_{n+1} : \tau \vdash t : \sigma]\!]$. By definition of the interpretation function, $[\![\Gamma \vdash \lambda\, x_{n+1} \,.\, t : \tau \Rightarrow \sigma]\!]$ is then the equivalence class of

$$\begin{aligned}\lambda\, z \,.\, t\left[\pi_i^{n+1} x / x_i\right]\left[\langle x, z\rangle / x\right] &= \lambda\, z \,.\, t\left[\pi_i^{n+1}\langle x, z\rangle / x_i\right] \\ &= \lambda\, z \,.\, t\left[\pi_i^{n}\left(\mathbf{fst}\,\langle x, z\rangle\right) / x_i\right]\left[\mathbf{snd}\,\langle x, z\rangle / x_{n+1}\right]\end{aligned}$$

We claim that the last term is equivalent to

$$\lambda\, z \,.\, t\left[\pi_i^{n} x / x_i\right]\left[z / x_{n+1}\right]$$

To show this let $k \in \mathbb{N}$, $(c, c') \in [\![\Gamma]\!]^{\top\top}(k)$. We will show directly that

$$\left(\lambda\, z \,.\, t\left[\pi_i^{n}\left(\mathbf{fst}\,\langle c, z\rangle\right) / x_i\right]\left[\mathbf{snd}\,\langle x, z\rangle / x_{n+1}\right], \lambda\, z \,.\, t\left[\pi_i^{n} c' / x_i\right]\left[z / x_{n+1}\right]\right) \in [\![\tau \Rightarrow \sigma]\!](n).$$

So take $i \leq k, (d, d') \in [\![\tau]\!]^{\top\top}(i), (e, e') \in [\![\sigma]\!]\downarrow(i)$. It is easy to see, using Lemma 12, Lemma 8 and induction on n, that for $i = 1, 2, \ldots, n$,

$$(\pi_i^n(\mathbf{fst}\,\langle c, d\rangle), \pi_i^n c') \in [\![\tau_i]\!]^{\top\top}(k)$$

and

$$(\pi_i^n c, \pi_i^n(\mathbf{fst}\,\langle c', d'\rangle)) \in [\![\tau_i]\!]^{\top\top}(k)$$

and that also

$$(\mathbf{snd}\,\langle c, d\rangle, d') \in [\![\tau]\!]^{\top\top}(i)$$

and

$$(d, \mathbf{snd}\,\langle c', d'\rangle) \in [\![\tau]\!]^{\top\top}(i).$$

Using Corollary 16 we get what is required. □

Theorem 40 (Adequacy)**.** *If* $[\![\cdot \vdash t : \mathbf{2}]\!] = [\mathbf{tt}]$ *then* $t \rightsquigarrow \mathbf{tt}$ *and if* $[\![\cdot \vdash t : \mathbf{2}]\!] = [\mathbf{ff}]$ *then* $t \rightsquigarrow \mathbf{ff}$*.*

Proof. Lemma 39 shows that $t \in [\![\cdot \vdash t : \mathbf{2}]\!]$, Lemma 38 then concludes the proof. □

Corollary 41. *If* $[\![\Gamma \vdash t : \sigma]\!] = [\![\Gamma \vdash s : \sigma]\!]$ *then* $\Gamma \vdash t \equiv_{ctx} s : \sigma$*.*

Proof. Let $C \in \mathbb{C}\left(\Gamma, \sigma \rhd \mathbf{2}\right)$. As the interpretation function is compositional we have $[\![\cdot \vdash C[t] : \mathbf{2}]\!] = [\![\cdot \vdash C[s] : \mathbf{2}]\!]$ (formally, this can be proved by a straightforward induction on C). Now soundness ensures that if $C[t] \rightsquigarrow \mathbf{tt}$ then $[\![\cdot \vdash C[t] : \mathbf{2}]\!] = [\![\cdot \vdash \mathbf{tt} : \mathbf{2}]\!] = [\mathbf{tt}]$. Theorem 40 now concludes the proof. □

4.3 Completeness

Theorem 42 (Full abstraction)**.**

$$(x_i : \tau_i)_{i=1}^n \vdash t \equiv_{ctx} s : \tau \Rightarrow [\![(x_i : \tau_i)_{i=1}^n \vdash t : \tau]\!] = [\![(x_i : \tau_i)_{i=1}^n \vdash s : \tau]\!]$$

Proof. Theorem 18 implies that contextual equivalence implies CIU-equivalence. Further, Lemma 39 shows it is enough to show that terms $t\left[\pi_i^n x/x_i\right]_{i=1}^n$ and $s\left[\pi_i^n x/x_i\right]_{i=1}^n$ are related.

So let $k \in \mathbb{N}$, $(c, c') \in [\![(x_i : \tau_i)_{i=1}^n]\!]\,(k)$, $j \leq k$, $(e, e') \in [\![\tau]\!] \downarrow (j)$ such that

$$e\left[t\left[\pi_i^n c/x_i\right]_{i=1}^n\right] \overset{\text{j}}{\rightsquigarrow} \mathbf{tt}\,.$$

Let $\xi = \tau_1 \otimes \tau_2 \otimes \cdot \otimes \tau_n$. As $x : \xi\tau_i \vdash t\left[\pi_i^n x/x_i\right]_{i=1}^n : \tau$ we use Corollary 16 to get

$$e'\left[t\left[\pi_i^n c'/x_i\right]_{i=1}^n\right] \rightsquigarrow \mathbf{tt}\,.$$

Now we use the assumption that t and s are CIU-equivalent and we are done. □

5 Discussion

We have showed how to construct a step-indexed logical relation for reasoning about contextual equivalence of programs in PCF$_\mu$. In fact, the method is not only sound, but also complete (Theorem 18). The completeness proof relies on (1) the fact that the logical relations are over *well-typed* terms (since contextual equivalence and CIU equivalence is defined on well-typed terms), and (2) the use of biorthogonality. These ingredients also ensure that the categorical model is fully abstract.

Because of completeness it is perhaps not surprising that we can define a category which models PCF$_\mu$ (since, after all, that can also be done by taking objects to be types and morphims to be terms with a free variable modulo contextual equivalence). But we emphasize that the definition of a categorical model of PCF$_\mu$ based on step-indexing can also be done in cases where we do not use PCF$_\mu$-well-typed terms as realizers (as long as the language of realizers is sufficiently expressive). Indeed, one can give a step-indexed model of a typed programming language where the realizers come from some other untyped programming language (this is what Hoshino did for a call-by-value language in [9]).

References

[1] Amal Ahmed. *Semantics of Types for Mutable State.* PhD thesis, Princeton University, 2004.

[2] Amal Ahmed, Derek Dreyer, and Andreas Rossberg. State-dependent representation independence. In *POPL*, 2009.

[3] Andrew Appel and David McAllester. An indexed model of recursive types for foundational proof-carrying code. *TOPLAS*, 23(5):657–683, 2001.

[4] L. Birkedal, B. Reus, J. Schwinghammer, K. Støvring, J. Thamsborg, and H. Yang. Step-indexed Kripke models over recursive worlds. In *Proceedings of POPL*, 2011.

[5] Lars Birkedal and Aleš Bizjak. A note on the transitivity of step-indexed logical relations. http://www.cs.au.dk/ birke/papers/step-transitivity.pdf, November 2012.

[6] Torben Braüner. A simple adequate categorical model for pcf. In *In Proceedings of Third International Conference on Typed Lambda Calculi and Applications*, pages 82–98. Springer-Verlag, 1997.

[7] Derek Dreyer, Georg Neis, and Lars Birkedal. The impact of higher-order state and control effects on local relational reasoning. In *ICFP*, pages 143–156, 2010.

[8] Derek Dreyer, Georg Neis, and Lars Birkedal. The impact of higher-order state and control effects on local relational reasoning. *J. Funct. Program.*, 22(4-5):477–528, 2012.

[9] N. Hoshino. Step indexed realizability semantics for a call-by-value language based on basic combinatorial objects. In *Logic in Computer Science (LICS), 2012 27th Annual IEEE Symposium on*, pages 385–394, 2012.

[10] Benedikt Meurer. A step-indexed semantic model of types for the call-by-name lambda calculus. *CoRR*, abs/1105.1985, 2011.

[11] A. M. Pitts. Typed operational reasoning. In B. C. Pierce, editor, *Advanced Topics in Types and Programming Languages*, chapter 7, pages 245–289. The MIT Press, 2005.

[12] A. M. Pitts. Step-indexed biorthogonality: a tutorial example. In A. Ahmed, N. Benton, L. Birkedal, and M. Hofmann, editors, *Modelling, Controlling and Reasoning About State*, number 10351 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2010. Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, Germany.

[13] Glynn Winskel. *The formal semantics of programming languages - an introduction.* Foundation of computing series. MIT Press, 1993.