STEP-INDEXED RELATIONAL REASONING FOR COUNTABLE NONDETERMINISM

LARS BIRKEDAL, ALEŠ BIZJAK, AND JAN SCHWINGHAMMER

Aarhus University
e-mail address: birkedal@cs.au.dk

Aarhus University
e-mail address: abizjak@cs.au.dk

Saarland University
e-mail address: jan@ps.uni-saarland.de

Abstract. Programming languages with countable nondeterministic choice are computationally interesting since countable nondeterminism arises when modeling fairness for concurrent systems. Because countable choice introduces non-continuous behaviour, it is well-known that developing semantic models for programming languages with countable nondeterminism is challenging. We present a step-indexed logical relations model of a higher-order functional programming language with countable nondeterminism and demonstrate how it can be used to reason about contextually defined may- and must-equivalence. In earlier step-indexed models, the indices have been drawn from ω. Here the step-indexed relations for must-equivalence are indexed over an ordinal greater than ω.

1. Introduction

Programming languages with countable nondeterministic choice are computationally interesting since countable nondeterminism arises when modeling fairness for concurrent systems. In this paper we show how to construct simple semantic models for reasoning about may- and must-equivalence in a call-by-value higher-order functional programming language with countable nondeterminism, recursive types and impredicative polymorphism.

Models for languages with nondeterminism have originally been studied using denotational techniques. In the case of countably branching nondeterminism it is not enough to consider standard ω-continuous complete partial orders and the denotational models become quite involved [3, 6]. This has sparked research in operationally-based theories of equivalence for nondeterministic higher-order languages [1, 12, 13, 14, 15, 20]. In particular, Lassen investigated operationally-based relational methods for countable nondeterminism

1998 ACM Subject Classification: F.3.2 Semantics of Programming Languages.

Key words and phrases: Countable choice, lambda calculus, program equivalence.

A preliminary version of this work has been presented at the 20th EACSL Annual Conference on Computer Science Logic (CSL'11), 12-15 September 2011, Bergen, Norway, see Section 7 for how the present paper relates to the conference paper.

LOGICAL METHODS IN COMPUTER SCIENCE DOI:10.2168/LMCS-???

and suggested that it would be interesting to consider also methods based on logical relations, i.e., where the *types* of the programming languages are given a relational interpretation [12, page 47]. Such an interpretation would allow one to relate terms of different types, as needed for reasoning about parametricity properties of polymorphic types.

For languages with recursive types, however, logical relations cannot be defined by induction on types. In the case of deterministic languages, this problem has been addressed by the technique of syntactic minimal invariance [4] (inspired by domain theory [17]). The idea here is that one proves that a syntactically definable fixed point on a recursive type is contextually equivalent to the identity function, and then uses a so-called unwinding theorem for syntactically definable fixed points when showing the existence of the logical relations. However, in the presence of countable nondeterminism it is not clear how to define the unwindings of the syntactic fixed point in the programming language. Indeed, Lassen proved an unwinding theorem for his language with countable nondeterminism, but he did so by extending the language with new terms needed for representing the unwindings and left open the question of whether this is a conservative extension of the language.

Here we give a logical relations model of our language where we do not rely on syntactic minimal invariance for constructing the logical relations. Instead, we use the idea of step-indexed logical relations [2]. In particular, we show how to use step-indexing over ordinals larger than ω to reason about must-equivalence in the presence of countable nondeterminism.

This approach turns out to be both simple and also useful for reasoning about concrete may- and must-equivalences. We show that our logical relations are sound and complete with respect to the contextually defined notions of may- and must-equivalence. Moreover, we show how to use our logical relations to establish some concrete equivalences. In particular, we prove the recursion-induction rule from Lassen [12] and establish the syntactic minimal invariance property (without extending the language with new unwinding terms). We also include an example to show that the model can be used to prove parametricity properties (free theorems) of polymorphic types.

Overview of the technical development. One way to understand the failure of ω-continuity in an operational setting is to consider the must-convergence predicate $e \Downarrow$, which by Tarski's fixed point theorem can be defined as the least fixed point of the monotone functional $\Phi(R) = \{e \mid \forall e'.\ e \longmapsto e' \ \Rightarrow\ e' \in R\}$ on sets of terms. Here $e \longmapsto e'$ means that e reduces to e' in one step. However, due to the countable branching the fixed point is not reached by ω-many iterations $\bigcup_{n\in\omega} \Phi^n(\emptyset)$. The reason is that even when a program has no infinite reduction sequences, we cannot in general bound the length of reduction sequences by any $n < \omega$.

The idea of step-indexed semantics is a stratified construction of relations which facilitates the interpretation of recursive types, and in previous applications this stratification has typically been realized by indexing over ω. However, as we pointed out, the closure ordinal of the inductively defined must-convergence predicate is strictly larger than ω: the least fixed point $\Downarrow$ is reached after ω_1-many iterations, for ω_1 the least uncountable ordinal. (In fact, the least non-recursive ordinal would suffice [3].) Thus, one of the key steps in our development is the definition of α-indexed uniform relations, for arbitrary ordinals α, in Section 3.

In Section 4 we define a logical ω-indexed uniform relation, and use this relation to prove a CIU theorem for may-contextual equivalence. The logical relation combines step-indexing and biorthogonality, and we can prove that it coincides with may-contextual equivalence;

$$
\begin{aligned}
\tau &::= \alpha \mid \mathbf{1} \mid \tau_1 \times \tau_2 \mid \tau_1 \to \tau_2 \mid \mu\alpha.\tau_1 + \ldots + \tau_n \mid \forall\alpha.\tau \\
v &::= x \mid \langle\rangle \mid \langle v_1, v_2\rangle \mid \lambda x.e \mid \mathtt{in}_i\, v \mid \Lambda\alpha.e \\
e &::= v \mid \mathtt{?} \mid \mathtt{proj}_i\, v \mid v\, e \mid \mathtt{case}\ v\ \mathtt{of}\ \mathtt{in}_1\, x_1.\, e_1 \mid \ldots \mid \mathtt{in}_n\, x_n.\, e_n \mid v\,\tau \\
E &::= [\,] \mid v\, E
\end{aligned}
$$

Figure 1: Types, terms and evaluation contexts

$$
\begin{aligned}
\mathtt{proj}_i\, \langle v_1, v_2\rangle &\longmapsto v_i & \mathtt{case}\ (\mathtt{in}_j\, v)\ \mathtt{of}\ (\ldots \mid \mathtt{in}_j\, x_j.\, e_j \mid \ldots) &\longmapsto e_j[v/x_j] \\
(\lambda x.e)\, v &\longmapsto e[v/x] & \mathtt{?} &\longmapsto \underline{n} \qquad (n \in \mathbb{N}) \\
(\Lambda\alpha.e)\,\tau &\longmapsto e[\tau/\alpha] & v\, e &\longmapsto v\, e' \quad \text{if } e \longmapsto e'
\end{aligned}
$$

Figure 2: Operational semantics

the proofs are similar to those in [19]. Section 5 considers the case of must-contextual equivalence. The only modifications that this requires, compared to Section 4, are the use of ω_1-indexed uniform relations and of a suitably adapted notion of biorthogonality.

Summary of contributions. In summary, the contribution of this paper is a simple, operationally-based logical relations model of countable nondeterminism in a higher-order language, and the use of this model for proving several non-trivial applications in Section 6. In particular, we derive a least prefixed point property for recursive functions in our language and characterize the elements of the type $\forall\alpha.\alpha \times \alpha \to \alpha$, using relational parametricity.

Laird [11] has developed a fully abstract denotational model based on bidomains for a calculus similar to the one studied here but without recursive and polymorphic types; our model appears to be the first model of countable nondeterminism for a language with impredicative polymorphism. Finite nondeterminism and polymorphism has been studied for a call-by-name language by Johann et. al. [10], who developed an operational theory for algebraic effects.

2. A lambda calculus with countable choice

Syntax and operational semantics. Figure 1 gives the syntax of a higher-order functional language with recursive and polymorphic types, and a (countably branching) choice construct. We assume disjoint, countably infinite sets of *type variables*, ranged over by α, and *term variables*, ranged over by x. The free type variables of types and terms, $ftv(\tau)$ and $ftv(e)$, and free term variables $fv(e)$, are defined in the usual way. The notation $(\cdot)[\vec{\tau}/\vec{\alpha}]$ denotes the simultaneous capture-avoiding substitution of types $\vec{\tau}$ for the free type variables $\vec{\alpha}$ in types and terms; similarly, $e[\vec{v}/\vec{x}]$ denotes simultaneous capture-avoiding substitution of values $\vec{v}$ for the free term variables $\vec{x}$ in e.

To reduce the number of proof cases in the formal development, we keep the syntax minimal. For instance, we only include $\mathtt{proj}_1 v$, for v a value and not an expression. In examples we may use additional syntactic sugar. We write $\mathtt{let}\ x = e\ \mathtt{in}\ e'$ for $(\lambda x.e')\, e$ and $e\,\tau$ for $\mathtt{let}\ f = e\ \mathtt{in}\ f\,\tau$ for some fresh f.

$$\frac{x{:}\tau \in \Gamma \qquad \Delta \vdash \Gamma}{\Delta; \Gamma \vdash x : \tau} \qquad \frac{\Delta \vdash \Gamma}{\Delta; \Gamma \vdash \langle\rangle : \mathbf{1}} \qquad \frac{\Delta; \Gamma \vdash v_1 : \tau_1 \qquad \Delta; \Gamma \vdash v_2 : \tau_2}{\Delta; \Gamma \vdash \langle v_1, v_2 \rangle : \tau_1 \times \tau_2}$$

$$\frac{\Delta; \Gamma, x{:}\tau_1 \vdash e : \tau_2}{\Delta; \Gamma \vdash \lambda x.e : \tau_1 \to \tau_2} \qquad \frac{\Delta; \Gamma \vdash v : \tau_j[\mu\alpha.\tau_1 + \ldots + \tau_n/\alpha]}{\Delta; \Gamma \vdash \mathtt{in}_j\, v : \mu\alpha.\tau_1 + \ldots + \tau_n}\ 1 \le j \le n$$

$$\frac{\Delta, \alpha; \Gamma \vdash e : \tau}{\Delta; \Gamma \vdash \Lambda\alpha.e : \forall\alpha.\tau} \qquad \frac{\Delta; \Gamma \vdash v : \tau_1 \times \tau_2}{\Delta; \Gamma \vdash \mathtt{proj}_i\, v : \tau_i} \qquad \frac{\Delta; \Gamma \vdash v : \tau' \to \tau \qquad \Delta; \Gamma \vdash e : \tau'}{\Delta; \Gamma \vdash v\, e : \tau}$$

$$\frac{\Delta; \Gamma \vdash v : \mu\alpha.\tau_1 + \ldots + \tau_n \qquad \ldots \qquad \Delta; \Gamma, x_j{:}\tau_j[\mu\alpha.\tau_1 + \ldots + \tau_n/\alpha] \vdash e_j : \tau \qquad \ldots}{\Delta; \Gamma \vdash \mathtt{case}\ v\ \mathtt{of}\ (\ldots \mid \mathtt{in}_j\, x_j.\, e_j \mid \ldots) : \tau}$$

$$\frac{\Delta; \Gamma \vdash v : \forall\alpha.\tau \qquad \Delta \vdash \tau'}{\Delta; \Gamma \vdash v\,\tau' : \tau[\tau'/\alpha]} \qquad \frac{\Delta \vdash \Gamma}{\Delta; \Gamma \vdash \mathtt{?} : \mathtt{nat}}$$

$$\frac{\varnothing \vdash \tau}{\vdash [] : \tau \multimap \tau} \qquad \frac{\varnothing; \varnothing \vdash v : \tau \to \tau_2 \qquad \vdash E : \tau_1 \multimap \tau}{\vdash v\, E : \tau_1 \multimap \tau_2}$$

Figure 3: Typing of terms and evaluation contexts, where $\Gamma ::= \emptyset \mid \Gamma, x{:}\tau$ and $\Delta ::= \emptyset \mid \Delta, \alpha$. The notation $\Delta \vdash \tau$ means that $\mathit{ftv}(\tau) \subseteq \Delta$, and $\Delta \vdash \Gamma$ means that $\Delta \vdash \tau$ holds for all $x{:}\tau \in \Gamma$.

We define the unary natural numbers datatype as $\mathtt{nat} = \mu\alpha.\mathbf{1} + \alpha$ and write $\underline{0} = \mathtt{in}_1\,\langle\rangle$ and $\underline{n{+}1} = \mathtt{in}_2(\underline{n})$. The 'erratic' (finitely branching) choice construct $e_1\ \mathtt{or}\ e_2$ can be defined from $\mathtt{?}$ as $\mathtt{let}\ x = \mathtt{?}\ \mathtt{in}\ \mathtt{case}\ x\ \mathtt{of}\ \mathtt{in}_1\, y.\, e_1 \mid \mathtt{in}_2\, y.\, e_2$ for fresh x, y.

The operational semantics of the language is given in Figure 2 by a reduction relation $e \longmapsto e'$. In particular, the choice operator $\mathtt{?}$ evaluates nondeterministically to any numeral $\underline{n}$ ($n \in \mathbb{N}$). We also consider evaluation contexts E, and write $E[e]$ for the term obtained by plugging e into E. It is easy to see that $e \longmapsto e'$ holds if and only if $E[e] \longmapsto E[e']$.

Typing judgements take the form $\Delta; \Gamma \vdash e : \tau$ where Γ is a typing context $x_1{:}\tau_1, \ldots, x_n{:}\tau_n$ and where Δ is a finite set of type variables that contains the free type variables of $\tau_1, \ldots, \tau_n$ and τ. The rules defining this judgement are summarized in Figure 3. The typing judgement for evaluation contexts, $\vdash E : \tau \multimap \tau'$, means that $\varnothing; \varnothing \vdash E[e] : \tau'$ holds whenever $\varnothing; \varnothing \vdash e : \tau$.

We write Type for the set of closed types τ, i.e., where $\mathit{ftv}(\tau) = \emptyset$. We write $\mathit{Val}(\tau)$ and $\mathit{Tm}(\tau)$ for the sets of closed values and terms of type τ, resp., and $\mathit{Stk}(\tau)$ for the set of τ-accepting evaluation contexts. For a typing context $\Gamma = x_1{:}\tau_1, \ldots, x_n{:}\tau_n$ with $\tau_1, \ldots, \tau_n \in \mathit{Type}$, let $\mathit{Subst}(\Gamma) = \{\gamma \in \mathit{Val}^{\vec{x}} \mid \forall 1 \le i \le n.\ \gamma(x_i) \in \mathit{Val}(\tau_i)\}$ denote the set of type-respecting value substitutions. In particular, if $\Delta; \Gamma \vdash e : \tau$ then $\varnothing; \varnothing \vdash e\delta\gamma : \tau\delta$ for any $\delta \in \mathit{Type}^{\Delta}$ and $\gamma \in \mathit{Subst}(\Gamma\delta)$, and the type system satisfies standard properties:

Lemma 1 (Canonical forms)**.**

- If $v \in \mathit{Val}(\mathbf{1})$ then v is $\langle\rangle$.
- If $v \in \mathit{Val}(\tau_1 \times \tau_2)$ then v is of the form $\langle v_1, v_2 \rangle$ for $v_i \in \mathit{Val}(\tau_i)$.
- If $v \in \mathit{Val}(\tau_1 \to \tau_2)$ then v is of the form $\lambda x.t$ for some x and e.
- If $v \in \mathit{Val}(\mu\alpha.\tau_1 + \ldots + \tau_m)$ then v is of the form $\mathtt{in}_j\, v'$ for some $1 \le j \le m$ and $v' \in \mathit{Val}(\tau_j[\mu\alpha.\tau_1 + \ldots + \tau_m/\alpha])$.

- If $v \in \mathit{Val}(\forall\alpha.\tau)$ then v is of the form $\Lambda\alpha.e$ for some α and e.

Proposition 2 (Preservation and progress)**.**

- If $e \in \mathit{Tm}(\tau)$ and $e \longmapsto e'$ then $e' \in \mathit{Tm}(\tau)$.
- If $e \in \mathit{Tm}(\tau) \setminus \mathit{Val}(\tau)$ then $e \longmapsto e'$ for some e'.

Following Lassen [12], we let $\mathtt{fix} : \forall\alpha,\beta.((\alpha\to\beta) \to (\alpha\to\beta)) \to (\alpha\to\beta)$ denote a variant of the (call-by-value) fixed point combinator from untyped lambda calculus, $\mathtt{fix} = \Lambda\alpha,\beta.\lambda f.\delta_f(\mathtt{in}\,\delta_f)$ where δ_f is the term $\lambda y.\mathtt{case}\ y\ \mathtt{of}\ \mathtt{in}\,y'.\, f(\lambda x.\mathtt{let}\ r = y'\,y\ \mathtt{in}\ r\,x)$. In Section 6 we show in what sense $\mathtt{fix}$ is a fixed point combinator. We write $\Omega : \forall\alpha.\alpha$ for the term $\Lambda\alpha.\mathtt{fix}\,\mathbf{1}\,\alpha\,(\lambda f.f)\,\langle\rangle$. Note that, for all closed types τ, reduction from $\Omega\,\tau$ is deterministic and non-terminating.

Contextual approximation. We follow Lassen's approach [12] and define contextual approximation as the largest relation that satisfies certain compatibility and adequacy properties (also see, e.g. [18, 19]). The technical advantage of this approach, compared to the more traditional one of universally quantifying over program contexts, is that in proofs there will be no need to explicitly take care of contexts and of term occurrences within contexts. In our terminology, we keep close to Pitts [18], except for suitably adapting the definitions to take the nondeterministic outcomes of evaluation into account.

The observables on which contextual approximation is based are given by may- and must-convergence. A closed term e *may-converges*, written $e\downarrow$, if $e \longmapsto^* v$ for some $v \in \mathit{Val}$, and e *may-diverges*, written $e\uparrow$, if there is an infinite reduction sequence starting from e. The *must-convergence* predicate $e \Downarrow$ is the complement of may-divergence, and it can be defined as the least predicate satisfying $e\Downarrow$ if for all e', if $e \longmapsto e'$ then $e'\Downarrow$. In addition, we say that e *must-diverges* if it does not may-converge.

Definition 3 (Type-indexed relation)**.** A *type-indexed relation* is a set of tuples $(\Delta,\Gamma,e,e',\tau)$ such that $\Delta;\Gamma \vdash e : \tau$ and $\Delta;\Gamma \vdash e' : \tau$ holds, where we write $\Delta;\Gamma \vdash e\ \mathcal{R}\ e' : \tau$ for $(\Delta,\Gamma,e,e',\tau) \in \mathcal{R}$.

Definition 4 (Precongruence)**.** A type-indexed relation $\mathcal{R}$ is *reflexive* if $\Delta;\Gamma\vdash e:\tau$ implies $\Delta;\Gamma \vdash e\ \mathcal{R}\ e : \tau$. It is *transitive* if $\Delta;\Gamma \vdash e\ \mathcal{R}\ e' : \tau$ and $\Delta;\Gamma \vdash e'\ \mathcal{R}\ e'' : \tau$ implies $\Delta;\Gamma \vdash e\,\mathcal{R}\,e'' : \tau$. A *precongruence* is a reflexive and transitive type-indexed relation $\mathcal{R}$ that is closed under the inference rules in Figure 4.

Definition 5 (May- and must-adequate relations)**.** A type-indexed relation $\mathcal{R}$ is *may-adequate* if, whenever $\varnothing;\varnothing \vdash e\,\mathcal{R}\,e' : \tau$ holds, then $e\downarrow$ implies $e'\downarrow$. It is *must-adequate* if, whenever $\varnothing;\varnothing \vdash e\,\mathcal{R}\,e' : \tau$ holds, then $e\Downarrow$ implies $e'\Downarrow$.

Definition 6 (Contextual approximations and equivalences)**.** *May-contextual approximation*, written $\lesssim^{ctx}_{\downarrow}$, is the largest may-adequate precongruence. *May-contextual equivalence*, $\cong^{ctx}_{\downarrow}$, is the symmetrization of $\lesssim^{ctx}_{\downarrow}$. Analogously, *must-contextual approximation*, written $\lesssim^{ctx}_{\Downarrow}$, is the largest must-adequate precongruence, and *must-contextual equivalence*, $\cong^{ctx}_{\Downarrow}$, is its symmetrization. *Contextual approximation*, $\lesssim^{ctx}$, and *contextual equivalence*, $\cong^{ctx}$, are given as intersections of the respective may- and must-relations, and thus $\cong^{ctx}$ is also the symmetrization of $\lesssim^{ctx}$.

$$\frac{}{\Delta;\Gamma \vdash x \mathrel{\mathcal{R}} x : \tau}\ x{:}\tau \in \Gamma \qquad \frac{}{\Delta;\Gamma \vdash \langle\rangle \mathrel{\mathcal{R}} \langle\rangle : \mathbf{1}}$$

$$\frac{\Delta;\Gamma \vdash v_1 \mathrel{\mathcal{R}} v_1' : \tau_1 \qquad \Delta;\Gamma \vdash v_2 \mathrel{\mathcal{R}} v_2' : \tau_2}{\Delta;\Gamma \vdash \langle v_1, v_2\rangle \mathrel{\mathcal{R}} \langle v_1', v_2'\rangle : \tau_1 \times \tau_2} \qquad \frac{\Delta;\Gamma, x{:}\tau_1 \vdash e \mathrel{\mathcal{R}} e' : \tau_2}{\Delta;\Gamma \vdash \lambda x.e \mathrel{\mathcal{R}} \lambda x.e' : \tau_1 \to \tau_2}$$

$$\frac{\Delta;\Gamma \vdash v \mathrel{\mathcal{R}} v' : \tau_j[\mu\alpha.\tau_1 + \ldots + \tau_n/\alpha]}{\Delta;\Gamma \vdash \mathtt{in}_j\, v \mathrel{\mathcal{R}} \mathtt{in}_j\, v' : \mu\alpha.\tau_1 + \ldots + \tau_n}\ 1 \le j \le n \qquad \frac{\Delta,\alpha;\Gamma \vdash e \mathrel{\mathcal{R}} e' : \tau}{\Delta;\Gamma \vdash \Lambda\alpha.e \mathrel{\mathcal{R}} \Lambda\alpha.e' : \forall\alpha.\tau}$$

$$\frac{\Delta;\Gamma \vdash v \mathrel{\mathcal{R}} v' : \tau_1 \times \tau_2}{\Delta;\Gamma \vdash \mathtt{proj}_i\, v \mathrel{\mathcal{R}} \mathtt{proj}_i\, v' : \tau_i} \qquad \frac{\Delta;\Gamma \vdash v \mathrel{\mathcal{R}} v' : \tau' \to \tau \qquad \Delta;\Gamma \vdash e \mathrel{\mathcal{R}} e' : \tau'}{\Delta;\Gamma \vdash v\, e \mathrel{\mathcal{R}} v'\, e' : \tau}$$

$$\frac{\Delta;\Gamma \vdash v \mathrel{\mathcal{R}} v' : \tau \qquad \ldots \qquad \Delta;\Gamma, x_j{:}\tau_j[\tau/\alpha] \vdash e_j \mathrel{\mathcal{R}} e_j' : \tau' \qquad \ldots}{\Delta;\Gamma \vdash \mathtt{case}\ v\ \mathtt{of}\,(\ldots \mid \mathtt{in}_j\, x_j.\, e_j \mid \ldots) \mathrel{\mathcal{R}} \mathtt{case}\ v'\ \mathtt{of}\,(\ldots \mid \mathtt{in}_j\, x_j.\, e_j \mid \ldots) : \tau'}\ \tau = \mu\alpha.\tau_1 + \ldots + \tau_n$$

$$\frac{\Delta;\Gamma \vdash v \mathrel{\mathcal{R}} v' : \forall\alpha.\tau}{\Delta;\Gamma \vdash v\,\tau' \mathrel{\mathcal{R}} v'\,\tau' : \tau[\tau'/\alpha]}\ ftv(\tau') \subseteq \Delta \qquad \frac{}{\Delta;\Gamma \vdash \mathtt{?} \mathrel{\mathcal{R}} \mathtt{?} : \mathtt{nat}}$$

Figure 4: Compatibility properties of type-indexed relations

That this largest (may-, must-) adequate precongruence exists can be shown as in [18], by proving that the relation $S = \bigcup\{R \mid R \text{ compatible and (may-, must-) adequate}\}$ is an adequate precongruence.

In principle, to establish an equivalence $\Delta;\Gamma \vdash e \cong^{ctx} e' : \tau$ it suffices to find some may- and must-adequate congruence $\mathcal{R}$ that contains the tuple $(\Delta, \Gamma, e, e', \tau)$ since $\cong^{ctx}$ is the largest such relation. However, in practice it is difficult to verify that a relation $\mathcal{R}$ has the necessary compatibility properties in Figure 4. An alternative characterization of the contextual approximation and equivalence relations can be given in terms of CIU preorders [16], which we define next.

Definition 7 (CIU preorders)**.** *May- and must-CIU preorder*, written $\lesssim_{\downarrow}^{ciu}$ and $\lesssim_{\Downarrow}^{ciu}$ resp., are the type-indexed relations defined as follows: for all e, e' with $\Delta;\Gamma \vdash e : \tau$ and $\Delta;\Gamma \vdash e' : \tau$,

- $\Delta;\Gamma \vdash e \lesssim_{\downarrow}^{ciu} e' : \tau \quad\Leftrightarrow\quad \forall\delta \in \mathit{Type}^{\Delta}, \gamma \in \mathit{Subst}(\Gamma\delta), E \in \mathit{Stk}(\tau\delta).\ E[e\delta\gamma]\downarrow \ \Rightarrow\ E[e'\delta\gamma]\downarrow$
- $\Delta;\Gamma \vdash e \lesssim_{\Downarrow}^{ciu} e' : \tau \quad\Leftrightarrow\quad \forall\delta \in \mathit{Type}^{\Delta}, \gamma \in \mathit{Subst}(\Gamma\delta), E \in \mathit{Stk}(\tau\delta).\ E[e\delta\gamma]\Downarrow \ \Rightarrow\ E[e'\delta\gamma]\Downarrow$

The CIU preorder is defined as the intersection of $\lesssim_{\downarrow}^{ciu}$ and $\lesssim_{\Downarrow}^{ciu}$.

Theorem 8 (CIU theorem)**.** The (may-, must-) CIU preorder coincides with (may-, must-) contextual approximation.

Using the CIU theorem, it is easy to verify that all the deterministic reductions are also valid equivalences, and that the various call-by-value eta laws hold. Moreover, we can establish the laws of Moggi's computational lambda calculus and the basic (inequational) theory of erratic choice (Figure 5). We will prove the CIU theorem in Section 4 (for the may-CIU preorder) and Section 5 (for the must-CIU preorder). The CIU theorem was also proved, using different operational techniques, for a langauge with countable nondeterminism (but no polymorphism) in [12].

$$\texttt{let}\ x = \texttt{?}\ \texttt{in}\ e \cong^{ctx} e \quad (x \notin fv(e)) \qquad \texttt{let}\ x = v\ \texttt{in}\ e \cong^{ctx} e[v/x] \qquad \texttt{let}\ x = e\ \texttt{in}\ x \cong^{ctx} e$$

$$e\ \texttt{or}\ e \cong^{ctx} e \qquad \Omega\,\tau \lesssim^{ctx}_{\downarrow} e \qquad \Omega\,\tau \lesssim^{ctx}_{\Downarrow} e$$

$$e_1\ \texttt{or}\ e_2 \cong^{ctx} e_2\ \texttt{or}\ e_1 \qquad e_1 \lesssim^{ctx}_{\downarrow} e_1\ \texttt{or}\ e_2 \qquad e_1\ \texttt{or}\ e_2 \lesssim^{ctx}_{\Downarrow} e_1$$

$$(e_1\ \texttt{or}\ e_2)\ \texttt{or}\ e_3 \cong^{ctx} e_1\ \texttt{or}\ (e_2\ \texttt{or}\ e_3) \qquad e\ \texttt{or}\ (\Omega\,\tau) \cong^{ctx}_{\downarrow} e \qquad e\ \texttt{or}\ (\Omega\,\tau) \cong^{ctx}_{\Downarrow} \Omega\,\tau$$

Figure 5: Basic may- and must-theory, where $e_1\ \texttt{or}\ e_2$ is an abbreviation for the term $\texttt{let}\ x = \texttt{?}\ \texttt{in}\ \texttt{case}\ x\ \texttt{of}\ \texttt{in}_1\, y.\, e_1\ \texttt{|}\ \texttt{in}_2\, y.\, e_2$, and e is of type τ.

3. Uniform relations

For a limit ordinal number α and a set X we define an *α-indexed uniform subset on X* to be a family $(R_\beta)_{\beta<\alpha}$ of subsets $R_\beta \subseteq X$ such that

- $R_0 = X$,
- $R_{\beta+1} \subseteq R_\beta$ for all $\beta < \alpha$, and
- $R_\lambda = \bigcap_{\beta<\lambda} R_\beta$ for every limit ordinal $\lambda < \alpha$.

Let $Rel_\alpha(X)$ denote the α-indexed uniform subsets on X.

Recursive definitions. The notions of n-equivalence, non-expansiveness and contractiveness (e.g., [5]) all generalize from the case of ω-indexed uniform subsets: Given α-indexed uniform subsets $R, S \in Rel_\alpha(X)$ and $\nu < \alpha$ we say that R and S are *ν-equivalent*, written $R \overset{\nu}{=} S$, if $R_\beta = S_\beta$ for all $\beta \leq \nu$. In particular, $R = S$ if and only if $R \overset{\nu}{=} S$ for all $\nu < \alpha$.

A function $F : Rel_\alpha(X_1) \times \cdots \times Rel_\alpha(X_n) \to Rel_\alpha(X)$ is *non-expansive* if $\vec{R} \overset{\nu}{=} \vec{S}$ implies $F(\vec{R}) \overset{\nu}{=} F(\vec{S})$, and F is *contractive* if $\vec{R} \overset{\nu}{=} \vec{S}$ implies $F(\vec{R}) \overset{\nu+1}{=} F(\vec{S})$. If $R \in Rel_\alpha(X)$ then $\triangleright R \in Rel_\alpha(X)$ is the uniform subset determined by $(\triangleright R)_{\beta+1} = R_\beta$; this operation gives rise to a contractive function on $Rel_\alpha(X)$. Henceforth, we often omit parentheses and write $\triangleright R_\beta$ for $(\triangleright R)_\beta$.

Proposition 9 (Unique fixed points)**.** If $F : Rel_\alpha(X) \to Rel_\alpha(X)$ is contractive, then F has a unique fixed point $\mathit{fix}\, r.F(r)$.

Proof. First note that F has at most one fixed point: if R, S are fixed points of F then, by the contractiveness of F, we can establish that $R = F(R) \overset{\nu}{=} F(S) = S$ holds for all $\nu < \alpha$ by induction and thus $R = S$.

Because of the uniformity conditions it is sufficient to give the components of the fixed point $\mathit{fix}\, r.F(r)$ that are indexed by successor ordinals. We set $\mathit{fix}\, r.F(r)_{\nu+1} = F(R)_{\nu+1}$ where $R \in Rel_\alpha(X)$ is defined by $R_\beta = \mathit{fix}\, r.F(r)_\beta$ for $\beta \leq \nu$ and $R_\beta = \emptyset$ for $\beta > \nu$. By induction, it is easy to see that $\mathit{fix}\, r.F(r) \in Rel_\alpha(X)$ and that $F(\mathit{fix}\, r.F(r))_\nu = \mathit{fix}\, r.F(r)_\nu$ holds for all $\nu < \alpha$, and thus $F(\mathit{fix}\, r.F(r)) = \mathit{fix}\, r.F(r)$. □

Proposition 9 is an instance of Di Gianantonio and Miculan's sheaf-theoretic fixed point theorem [7]. Indeed, an α-indexed uniform subset on X corresponds to a subobject of the constant sheaf on X in the sheaf topos on α.

Uniform relations on syntax. For $\tau, \tau' \in \mathit{Type}$ we consider the collections of β-indexed *uniform relations* between values, terms and evaluation contexts: we write $\mathit{VRel}_\beta(\tau, \tau')$ for $\mathit{Rel}_\beta(\mathit{Val}(\tau) \times \mathit{Val}(\tau'))$, we write $\mathit{SRel}_\beta(\tau, \tau')$ for $\mathit{Rel}_\beta(\mathit{Stk}(\tau) \times \mathit{Stk}(\tau'))$, and we use $\mathit{TRel}_\beta(\tau, \tau')$ for $\mathit{Rel}_\beta(\mathit{Tm}(\tau) \times \mathit{Tm}(\tau'))$. Note that a value relation may relate values of distinct types; that is essential for reasoning about relational parametricity, see, e.g., the proof of Lemma 32.

The description of the logical relations in the sections below makes use of the following (non-expansive) constructions on uniform relations:

- $R_1 \times R_2 \in \mathit{VRel}_\beta(\tau_1 \times \tau_2, \tau_1' \times \tau_2')$, for $R_1 \in \mathit{VRel}_\beta(\tau_1, \tau_1')$ and $R_2 \in \mathit{VRel}_\beta(\tau_2, \tau_2')$, is defined by $(R_1 \times R_2)_\nu = \{(\langle v_1, v_2\rangle, \langle v_1', v_2'\rangle) \mid (v_1, v_1') \in (R_1)_\nu \ \wedge \ (v_2, v_2') \in (R_2)_\nu\}$.
- $R_1 \to R_2 \in \mathit{VRel}_\beta(\tau_1 \to \tau_2, \tau_1' \to \tau_2')$, for $R_1 \in \mathit{VRel}_\beta(\tau_1, \tau_1')$ and $R_2 \in \mathit{TRel}_\beta(\tau_2, \tau_2')$, is given by $(R_1 \to R_2)_\nu = \{(\lambda x.e, \lambda x.e') \mid \forall \nu' \leq \nu.\, \forall (v, v') \in (R_1)_{\nu'}.\, (e[v/x], e'[v'/x]) \in (R_2)_{\nu'}\}$.
- $\forall r.F(r) \in \mathit{VRel}_\beta(\forall \alpha.\tau_1, \forall \alpha.\tau_1')$, for $F_{\tau,\tau'} : \mathit{VRel}_\beta(\tau, \tau') \to \mathit{TRel}_\beta(\tau_1[\tau/\alpha], \tau_1'[\tau'/\alpha])$ a family of non-expansive maps, is the uniform relation that is defined by $\forall r.F(r)_\nu = \{(\Lambda\alpha.e, \Lambda\alpha.e') \mid \forall \tau, \tau' \in \mathit{Type}, R \in \mathit{VRel}_\nu(\tau, \tau').\ (e[\tau/\alpha], e'[\tau'/\alpha] \in F_{\tau,\tau'}(R)_\nu\}$.
- $\mathtt{in}_j R \in \mathit{VRel}_\beta(\tau, \tau')$, for $\tau = \mu\alpha.\tau_1 + \ldots + \tau_m$ and $\tau' = \mu\alpha.\tau_1' + \ldots + \tau_n'$ and $R \in \mathit{VRel}_\beta(\tau_j[\tau/\alpha], \tau_j'[\tau'/\alpha])$, is given by $(\mathtt{in}_j R)_\nu = \{(\mathtt{in}_j\, v, \mathtt{in}_j\, v') \mid (v, v') \in R_\nu\}$.

4. May equational theory

In this section, we will define a logical uniform relation that is used to prove that may-CIU preorder and may-contextual approximation coincide. The key idea of the definition is the usual one of step-indexing [2], i.e., that the observables can be stratified based on step-counting in the operational semantics. Let us refer to reduction steps of the form

$$\mathtt{case}\ (\mathtt{in}_j\, v)\ \mathtt{of}\ (\ldots \mid \mathtt{in}_j\, x_j.\, e_j \mid \ldots) \longmapsto e_j[v/x_j]$$

as *unfold-fold reductions*. Following [8] we will only count such unfold-fold reductions. The advantage of this is that the interpretation of types is slightly more extensional than if we counted all reduction steps; see the precise formulation in Lemma 14 below. Hence we define

$$e \rightsquigarrow^0 e'$$

to mean that $e \longmapsto^* e'$ and *none* of the reductions in the reduction sequence is an unfold-fold reduction, and we define

$$e \rightsquigarrow^1 e'$$

to mean that $e \longmapsto^* e'$ and *exactly one* of the reductions in the reduction sequence is an unfold-fold reduction.

We shall also make use of pure reductions. To that end, we refer to reductions of the form

$$? \longmapsto \underline{n}$$

as *choice reductions*. We then define

$$e \overset{p}{\rightsquigarrow} e'$$

to mean that $e \longmapsto^* e'$ and *none* of the reductions in the reduction sequence is a choice reduction. Further, we define

$$e \overset{p}{\rightsquigarrow}{}^0 e'$$

to mean that $e \rightsquigarrow^0 e'$ and $e \overset{p}{\rightsquigarrow} e'$.

We write $e \downarrow_n$ if $e \longmapsto^* v$ for some $v \in \mathit{Val}$ and *at most* n reduction steps are unfold-fold reductions.

Logical ω-indexed uniform relation for may-approximation. In the case of may-approximation, it suffices to consider ω-indexed uniform relations. Using the constructions on relations given above, we define a relational interpretation $[\![\tau]\!]\,(\vec{r}) \in \mathit{VRel}_\omega(\tau[\vec{\tau}/\vec{\alpha}], \tau[\vec{\tau}'/\vec{\alpha}])$ by induction on the type $\vec{\alpha} \vdash \tau$, given closed types $\tau_1, \tau_1', \ldots, \tau_k, \tau_k' \in \mathit{Type}$ and relations $r_1 \in \mathit{VRel}_\omega(\tau_1, \tau_1'), \ldots, r_k \in \mathit{VRel}_\omega(\tau_k, \tau_k')$:

$$[\![\alpha_i]\!]\,(\vec{r}) = r_i \qquad [\![\tau_1 \times \tau_2]\!]\,(\vec{r}) = [\![\tau_1]\!]\,(\vec{r}) \times [\![\tau_2]\!]\,(\vec{r})$$

$$[\![\mathbf{1}]\!]\,(\vec{r}) = (\mathit{Id}_{\mathbf{1}})_{n<\omega} \qquad [\![\tau_1 \to \tau_2]\!]\,(\vec{r}) = [\![\tau_1]\!]\,(\vec{r}) \to ([\![\tau_2]\!]\,(\vec{r}))^{\perp\!\!\perp}$$

$$[\![\forall\alpha.\tau]\!]\,(\vec{r}) = \forall r.([\![\tau]\!]\,(\vec{r}, r))^{\perp\!\!\perp} \qquad [\![\mu\alpha.\tau_1 + \ldots + \tau_m]\!]\,(\vec{r}) = \mathit{fix}\, s. \bigcup_j \mathtt{in}_j(\triangleright\, [\![\tau_j]\!]\,(\vec{r}, s))$$

Here, value relations $r \in \mathit{VRel}_\omega(\tau, \tau')$ are lifted to relations $r^\perp \in \mathit{SRel}_\omega(\tau, \tau')$ on evaluation contexts and to relations $r^{\perp\!\!\perp} \in \mathit{TRel}_\omega(\tau, \tau')$ on terms by biorthogonality, much as in [9]:

$$r_n^\perp = \{(E, E') \mid \forall j \le n.\ \forall (v, v') \in r_j.\ E[v] \downarrow_j \Rightarrow\ E'[v'] \downarrow\ \}$$

$$r_n^{\perp\!\!\perp} = \{(e, e') \mid \forall j \le n.\ \forall (E, E') \in r_j^\perp.\ E[e] \downarrow_j \Rightarrow\ E'[e'] \downarrow\ \}$$

The fixed point in the interpretation of recursive types is well-defined by Proposition 9 since each $[\![\tau]\!]$ denotes a family of non-expansive functions, and thus composition with $\triangleright$ yields a contractive function. Intuitively, we want to relate two values $\mathtt{in}_1 v$ and $\mathtt{in}_1 v'$ of a recursive type if v and v' are related at the unfolded type. We cannot define the relation that way. Instead we only require that v and v' are related at one step later. This suffices because we count unfold-fold reductions, see the proof of Proposition 15 for details.

We often omit parentheses and write $[\![\tau]\!]\,\vec{r}_n$ for $([\![\tau]\!]\,\vec{r})_n$ and $[\![\tau]\!]\,\vec{r}_n^\perp$ for $([\![\tau]\!]\,\vec{r})_n^\perp$ and $[\![\tau]\!]\,\vec{r}_n^{\perp\!\!\perp}$ for $([\![\tau]\!]\,\vec{r})_n^{\perp\!\!\perp}$.

The following lemmas express basic properties of the defined relations which are often used in subsequent proofs and calculations.

Lemma 10 (Substitution)**.** If $\Delta, \alpha \vdash \tau$ and $\Delta \vdash \tau'$ then $[\![\tau[\tau'/\alpha]]\!]\,(\vec{r}) = [\![\tau]\!]\,(\vec{r}, [\![\tau']\!]\,(\vec{r}))$.

Lemma 11 (Extensiveness)**.** For all $r \in \mathit{VRel}(\tau, \tau')$, $r \subseteq r^{\perp\!\!\perp}$.

Lemma 12 (Monotonicity)**.** For all $r, s \in \mathit{VRel}(\tau, \tau')$, if $r \subseteq s$ then $r^{\perp\!\!\perp} \subseteq s^{\perp\!\!\perp}$.

Lemma 13 (Context composition)**.** If $(v, v') \in [\![\tau_1 \to \tau_2]\!]\,\vec{r}_n$ and $(E, E') \in [\![\tau_2]\!]\,\vec{r}_n^\perp$ then $(E[v\,[]], E'[v'\,[]]) \in [\![\tau_1]\!]\,\vec{r}_n^\perp$.

Proof. Let $j \le n$, $(v_1, v_1') \in [\![\tau_1]\!]\,\vec{r}_j$. Assume $E[v\,v_1] \downarrow_j$. We have $v = \lambda x.e$ and $v' = \lambda x.e'$ and $(\lambda x.e, \lambda x.e') \in [\![\tau_1 \to \tau_2]\!]\,\vec{r}_n$ for some x, e, e' and since $E[v\,v_1] \longmapsto E[e[v_1/x]]$ also $E[e[v_1/x]] \downarrow_j$. By definition, $(e[v_1/x], e'[v_1'/x]) \in [\![\tau_2]\!]\,\vec{r}_j^{\perp\!\!\perp}$. From $(E, E') \in [\![\tau_2]\!]\,\vec{r}_n^\perp$ we obtain $E'[e'[v_1'/x]] \downarrow$. Thus, $E'[v'\,v_1'] \downarrow$. □

The following lemma expresses that the term-relations are closed on the right under arbitrary pure reduction sequences and on the left under zero-step pure reduction sequences. (The lemma could be strengthened slightly by allowing some of the reductions to be non-pure, but the way it is stated now, it holds both for the may interpretations of types, and also for the must interpretations of types given in the following section.)

Lemma 14. For all $(e, e') \in [\![\tau]\!]_n^{\perp\!\!\perp}$,

- if $e_1' \overset{p}{\rightsquigarrow} e' \overset{p}{\rightsquigarrow} e_2'$, then $(e, e_1') \in [\![\tau]\!]_n^{\perp\!\!\perp}$ and $(e, e_2') \in [\![\tau]\!]_n^{\perp\!\!\perp}$;
- if $e_1 \overset{p\ 0}{\rightsquigarrow} e \overset{p\ 0}{\rightsquigarrow} e_2$, then $(e_1, e') \in [\![\tau]\!]_n^{\perp\!\!\perp}$ and $(e_2, e') \in [\![\tau]\!]_n^{\perp\!\!\perp}$.

The proof is straightforward; the use of $\overset{p\ 0}{\rightsquigarrow}$ in the second item ensures that the index of the relation does not change.

The relational interpretation extends pointwise to value substitutions: $(\gamma, \gamma') \in [\![\Gamma]\!]\, \vec{r}_n$ if $(\gamma(x), \gamma(x')) \in [\![\tau]\!]\, \vec{r}_n$ for all $x{:}\tau \in \Gamma$. Based on this interpretation we consider the following type-indexed relation:

$$\begin{aligned}\Delta; \Gamma \vdash e \lesssim_{\downarrow}^{log} e' : \tau \quad & \text{where } \Delta = \vec{\alpha} \\ \Leftrightarrow\ \forall \vec{\tau}, \vec{\tau}'. \forall \vec{r} \in VRel_\omega(\vec{\tau}, \vec{\tau}'). \forall n < \omega. \forall (\gamma, \gamma') \in [\![\Gamma]\!]\, \vec{r}_n.\ & (e[\vec{\tau}/\vec{\alpha}]\gamma, e'[\vec{\tau}'/\vec{\alpha}]\gamma') \in [\![\tau]\!]\, \vec{r}_n^{\perp\!\!\perp}\end{aligned}$$

The definition of $\lesssim_{\downarrow}^{log}$ builds in enough closure properties to prove its compatibility.

Proposition 15 (Fundamental property)**.** The relation $\lesssim_{\downarrow}^{log}$ has the compatibility properties given in Figure 4. In particular, it is reflexive: if $\Delta; \Gamma \vdash e : \tau$ then $\Delta; \Gamma \vdash e \lesssim_{\downarrow}^{log} e : \tau$.

Proof. We consider the inference rules from Figure 4 in turn.

- For the introduction of recursive types, we assume that
$$\Delta; \Gamma \vdash v \lesssim_{\downarrow}^{log} v' : \tau_j[\mu\alpha.\tau_1 + \ldots + \tau_m/\alpha],$$
and then prove that
$$\Delta; \Gamma \vdash \mathtt{in}_j\, v \lesssim_{\downarrow}^{log} \mathtt{in}_j\, v' : \mu\alpha.\tau_1 + \ldots + \tau_m.$$
For notational convenience we only consider the case of closed terms. Let τ abbreviate the type $\mu\alpha.\tau_1 + \ldots + \tau_m$. Note that
$$[\![\tau]\!]\, \vec{r} = \bigcup_j \mathtt{in}_j\big(\triangleright\, [\![\tau_j]\!]\, (\vec{r}, [\![\tau]\!]\, \vec{r})\big) = \bigcup_j \mathtt{in}_j\big(\triangleright\, [\![\tau_j[\tau/\alpha]]\!]\, (\vec{r})\big)$$
by definition and Lemma 10, and that the inclusion $[\![\tau_j[\tau/\alpha]]\!]\, (\vec{r}) \subseteq \triangleright\, [\![\tau_j[\tau/\alpha]]\!]\, (\vec{r})$ holds. It is easy to see, straight from the definition, that $(\lambda x.\mathtt{in}_j\, x, \lambda x.\mathtt{in}_j\, x) \in [\![\tau_j[\tau/\alpha] \to \tau]\!]\, \vec{r}_n$, so assuming $(E, E') \in [\![\tau]\!]\, \vec{r}_n^{\perp}$ it follows from Lemma 13 that
$$(E[(\lambda x.\mathtt{in}_j\, x)\, []], E'[(\lambda x.\mathtt{in}_j\, x)\, []]) \in [\![\tau_j[\tau/\alpha]]\!]\, \vec{r}_n^{\perp}.$$
Thus, if $E[\mathtt{in}_j\, v] \downarrow_i$ for some $i \leq n$ then $E'[(\lambda x.\mathtt{in}_j\, x)\, v']) \downarrow$ follows from $(v, v') \in [\![\tau_j[\tau/\alpha]]\!]\, \vec{r}_n^{\perp\!\!\perp}$. Therefore we can conclude $E'[\mathtt{in}_j\, v'] \downarrow$, and have shown $(\mathtt{in}_j\, v, \mathtt{in}_j\, v') \in [\![\tau]\!]\, \vec{r}_n^{\perp\!\!\perp}$. Since n was chosen arbitrarily, we have $\Delta; \Gamma \vdash \mathtt{in}_j\, v \lesssim_{\downarrow}^{log} \mathtt{in}_j\, v' : \tau$.
- For the elimination of recursive types, we assume that τ is of the form $\mu\alpha.\tau_1 + \ldots + \tau_m$, $\Delta; \Gamma, x_j{:}\tau_j[\tau/\alpha] \vdash e_j \lesssim_{\downarrow}^{log} e_j' : \tau'$ for all $1 \leq j \leq m$ and $\Delta; \Gamma \vdash v \lesssim_{\downarrow}^{log} v' : \tau$. We prove $\Delta; \Gamma \vdash \mathtt{case}\ v\ \mathtt{of}(\ldots | \mathtt{in}_j\, x_j.\, e_j | \ldots) \lesssim_{\downarrow}^{log} \mathtt{case}\ v'\ \mathtt{of}(\ldots | \mathtt{in}_j\, x_j.\, e_j' | \ldots) : \tau'$.

For simplicity we only consider the case of closed terms. By definition and by Lemma 10 we have $[\![\tau]\!]\,\vec{r} = \bigcup_j \mathtt{in}_j\big(\triangleright\,[\![\tau_j]\!]\,(\vec{r}, [\![\tau]\!]\,\vec{r})\big) = \bigcup_j \mathtt{in}_j(\triangleright\,[\![\tau_j[\tau/\alpha]]\!]\,\vec{r})$. Moreover, $(\lambda x.\mathtt{case}\ x\ \mathtt{of}(\ldots \mid \mathtt{in}_j\, x_j.\, e_j \mid \ldots), \lambda x.\mathtt{case}\ x\ \mathtt{of}(\ldots \mid \mathtt{in}_j\, x_j.\, e'_j \mid \ldots)) \in [\![\tau \to \tau']\!]\,\vec{r}_n$ for any n. To see this, assume $k \le n$, let $(a, a') \in [\![\tau]\!]\,\vec{r}_n$ and $(E, E') \in [\![\tau']\!]\,\vec{r}_n^{\perp}$ such that $E[\mathtt{case}\ a\ \mathtt{of}(\ldots \mid \mathtt{in}_j\, x_j.\, e_j \mid \ldots)]\downarrow_k$. This implies that $k > 0$ and by the above observation we have $a = \mathtt{in}_j a_j$ and $a' = \mathtt{in}_j a'_j$ for some $(a_j, a'_j) \in [\![\tau_j[\tau/\alpha]]\!]\,\vec{r}_{k-1}$. From $E[\mathtt{case}\ a\ \mathtt{of}(\ldots \mid \mathtt{in}_j\, x_j.\, e_j \mid \ldots)]\downarrow_k$ we obtain $E[e_j[a_j/x_j]]\downarrow_{k-1}$, and thus the assumption on e_j and e'_j gives $E'[e'_j[a'_j/x_j]]\downarrow$. From this we can conclude that $E'[\mathtt{case}\ a'\ \mathtt{of}(\ldots \mid \mathtt{in}_j\, x_j.\, e'_j \mid \ldots)]\downarrow$ holds.

To prove the case, assume next that $(E, E') \in [\![\tau']\!]\,\vec{r}_n^{\perp}$. From Lemma 13 we obtain $(E[(\lambda x.\mathtt{case}\ x\ \mathtt{of}(\ldots \mid \mathtt{in}_j\, x_j.\, e_j \mid \ldots))\,[]], E'[(\lambda x.\mathtt{case}\ x\ \mathtt{of}(\ldots \mid \mathtt{in}_j\, x_j.\, e'_j \mid \ldots))\,[]]) \in [\![\tau]\!]\,\vec{r}_n^{\perp}$. Now, since we know $(v, v') \in [\![\tau]\!]\,\vec{r}_n^{\perp\perp}$ by assumption, we obtain that $E[\mathtt{case}\ v\ \mathtt{of}(\ldots \mid \mathtt{in}_j\, x_j.\, e_j \mid \ldots)]\downarrow_n$ implies $E[\mathtt{case}\ v'\ \mathtt{of}(\ldots \mid \mathtt{in}_j\, x_j.\, e'_j \mid \ldots)]\downarrow$ as required.

- For choice, we assume $\Delta \vdash \Gamma$ and show $\Delta;\Gamma \vdash \mathtt{?} \lesssim_{\downarrow}^{log} \mathtt{?} : \mathtt{nat}$. Suppose $(E, E') \in [\![\mathtt{nat}]\!]\,\vec{r}_n^{\perp}$ and $E[\mathtt{?}]\downarrow_j$ for some $j \le n$. Then $E[\mathtt{?}] \longmapsto E[\underline{k}]$ and $E[\underline{k}]\downarrow_j$ for some $k \in \mathbb{N}$. By induction on k we obtain that $(\underline{k}, \underline{k}) \in [\![\mathtt{nat}]\!]\,\vec{r}_n$, and thus $E'[\underline{k}]\downarrow$. Hence $E'[\mathtt{?}]\downarrow$.

The proofs for the remaining rules are similar. □

Corollary 16. If $v \in Val(\tau)$ then for all $n < \omega$, $(v, v) \in [\![\tau]\!]_n$.

Proof. We prove this by induction on the value v.

- Suppose $\tau = \tau_1 \to \tau_2$ and $v = \lambda x.e$. Fix n and let $i \le n$. For arbitrary $(u, u') \in [\![\tau_1]\!]_i$ we have to prove $(e[u/x], e[u'/x]) \in [\![\tau_2]\!]_i^{\perp\perp}$. Since $\varnothing; x:\tau_1 \vdash e:\tau_2$ using Proposition 15 we have $\varnothing; x:\tau_1 \vdash e \lesssim_{\downarrow}^{log} e:\tau_2$. If we instantiate this with i and the substitution $x \mapsto (u, u')$ we get what is required.
- Suppose $\tau = \forall\alpha.\tau$ and $v = \Lambda\alpha.e$. Fix n, pick $\tau, \tau' \in Type$ and $R \in VRel_n(\tau, \tau')$. We have to show $(e[\tau/\alpha], e[\tau'/\alpha]) \in [\![\tau]\!]\,R_n^{\perp\perp}$, but this again follows straightforwardly from Proposition 15.

The other cases follows straightforwardly from the induction hypothesis. The case for $\mathtt{in}_j$ also requires Lemma 10. □

Theorem 17 (Coincidence)**.** $\Delta;\Gamma \vdash e \lesssim_{\downarrow}^{log} e' : \tau$ if and only if $\Delta;\Gamma \vdash e \lesssim_{\downarrow}^{ciu} e' : \tau$.

Proof. For the direction from left to right, let $\delta \in Type^{\Delta}$, $\gamma \in Subst(\Gamma\delta)$ and $E \in Stk(\tau\delta)$, and assume $E[e\delta\gamma]\downarrow$. Then $E[e\delta\gamma]\downarrow_n$ for some n. We must show $E[e'\delta\gamma]\downarrow$. As a consequence of Proposition 15 and Corollary 16, $(\gamma, \gamma) \in [\![\Gamma\delta]\!]_n$ and $(E, E) \in [\![\tau\delta]\!]_n^{\perp}$. By definition of $\Delta;\Gamma \vdash e \lesssim_{\downarrow}^{log} e' : \tau$ and Lemma 10 we have $(e\delta\gamma, e'\delta\gamma) \in [\![\tau\delta]\!]_n^{\perp\perp}$, and thus $E[e\delta\gamma]\downarrow_n$ gives $E[e'\delta\gamma]\downarrow$.

For the direction from right to left, first note that the logical relation is closed under may-CIU approximation; more precisely, if $\Delta;\Gamma \vdash e \lesssim_{\downarrow}^{log} e' : \tau$ and $\Delta;\Gamma \vdash e' \lesssim_{\downarrow}^{ciu} e'' : \tau$ then $\Delta;\Gamma \vdash e \lesssim_{\downarrow}^{log} e'' : \tau$. This observation follows from the definition of $(\cdot)^{\perp\perp}$ used in $\Delta;\Gamma \vdash e \lesssim_{\downarrow}^{log} e' : \tau$ and the definition of CIU approximation. Now assume that $\Delta;\Gamma \vdash e \lesssim_{\downarrow}^{ciu} e' : \tau$. By Proposition 15, $\Delta;\Gamma \vdash e \lesssim_{\downarrow}^{log} e : \tau$, and thus $\Delta;\Gamma \vdash e \lesssim_{\downarrow}^{log} e' : \tau$. □

$$\frac{\Delta;\Gamma \vdash v \mathrel{\mathcal{R}} v' : \tau \qquad \Delta;\Gamma, x{:}\tau \vdash e \mathrel{\mathcal{R}} e' : \tau'}{\Delta;\Gamma \vdash e[v/x] \mathrel{\mathcal{R}} e'[v'/x] : \tau'} \qquad \frac{\Delta, \alpha;\Gamma \vdash e \mathrel{\mathcal{R}} e' : \tau'}{\Delta;\Gamma[\tau/\alpha] \vdash e \mathrel{\mathcal{R}} e' : \tau'[\tau/\alpha]}\ \Delta \vdash \tau$$

Figure 6: Substitutivity properties of type-indexed relations

Proof of CIU Theorem 8(1). We first show that $\lesssim_{\downarrow}^{ciu}$ is contained in $\lesssim_{\downarrow}^{ctx}$. By definition, $\lesssim_{\downarrow}^{ctx}$ is the largest may-adequate precongruence, thus it is sufficient to establish that $\lesssim_{\downarrow}^{ciu}$ is a may-adequate precongruence. From the definition it is immediate that $\lesssim_{\downarrow}^{ciu}$ is may-adequate, reflexive and transitive. By Theorem 17, $\lesssim_{\downarrow}^{ciu}$ coincides with $\lesssim_{\downarrow}^{log}$ which is compatible by Proposition 15.

For the other direction, following Pitts [19], we first consider the special case where $\varnothing;\varnothing \vdash e \lesssim_{\downarrow}^{ctx} e' : \tau$. To prove $\varnothing;\varnothing \vdash e \lesssim_{\downarrow}^{ciu} e' : \tau$, note that $\varnothing;\varnothing \vdash E[e] \lesssim_{\downarrow}^{ctx} E[e'] : \tau'$ holds for all evaluation contexts E such that $\vdash E : \tau \multimap \tau'$ since $\lesssim_{\downarrow}^{ctx}$ is reflexive and compatible. Hence, that $E[e]\downarrow$ implies $E[e']\downarrow$ follows since $\lesssim_{\downarrow}^{ctx}$ is may-adequate.

The general case reduces to this special case since may-contextual approximation has the substitutivity properties given in Figure 6. For the first of these, assume $\Delta;\Gamma \vdash v \lesssim_{\downarrow}^{ctx} v' : \tau$ and $\Delta;\Gamma, x{:}\tau \vdash e \lesssim_{\downarrow}^{ctx} e' : \tau'$. From the definition of may-CIU approximation it is easy to see

$$\Delta;\Gamma \vdash e[v/x] \lesssim_{\downarrow}^{ciu} (\lambda x.e)\, v : \tau' \quad \text{and} \quad \Delta;\Gamma \vdash (\lambda x.e')\, v' \lesssim_{\downarrow}^{ciu} e'[v'/x] : \tau' \ .$$

Since we have already shown that $\lesssim_{\downarrow}^{ciu}$ is contained in $\lesssim_{\downarrow}^{ctx}$, and since $\Delta;\Gamma \vdash (\lambda x.e)\, v \lesssim_{\downarrow}^{ctx} (\lambda x.e')\, v' : \tau'$ by compatibility, we can conclude $\Delta;\Gamma \vdash e[v/x] \lesssim_{\downarrow}^{ctx} e'[v'/x] : \tau'$ by transitivity. The second substitutivity property is proved similarly, using a weakening property of may-contextual approximation. □

Using the logical relation, we now prove some simple extensionality properties for may contextual approximation and equivalence. We will use these properties in the parametricity example in Section 6.

Lemma 18. If $v \in \mathit{Val}\,(\forall\alpha.\sigma)$ then $\forall\tau,\tau' \in \mathit{Type}, \forall R \in \mathit{VRel}\,(\tau,\tau'), \forall n < \omega, (v\,\tau, v\,\tau') \in [\![\sigma]\!]\, R_n{}^{\perp\!\!\perp}$.

Proof. Take $n < \omega, j \leq n, (E, E') \in [\![\sigma]\!]\, R_j^{\perp}$ and assume $E\,[v\,\tau]\downarrow_j$ which is equivalent to $E[(\lambda x.x\,\tau)v]\downarrow_j$. It is easy to see that $(E\,[(\lambda x.x\,\tau)[]]\,, E'\,[(\lambda x.x\,\tau')[]]) \in [\![\forall\alpha.\sigma]\!]\, R_j^{\perp}$ and using Proposition 15 we have $\forall n < \omega, (v, v) \in [\![\forall\alpha.\alpha\times\alpha\to\alpha]\!]_n^{\perp\!\!\perp}$ which concludes the proof. □

Lemma 19 (Application)**.** If $(e, e') \in [\![\tau_1]\!]\, \vec{r}_n^{\perp\!\!\perp}$ and $(v, v') \in [\![\tau_1 \to \tau_2]\!]\, \vec{r}_n$ then $(v\,e, v'\,e') \in [\![\tau_2]\!]\, \vec{r}_n^{\perp\!\!\perp}$.

Proof. For any $(E, E') \in [\![\tau_2]\!]\, \vec{r}_n^{\perp}$, $(E[v\,[]], E'[v'\,[]]) \in [\![\tau_1]\!]\, \vec{r}_n^{\perp}$ by Lemma 13. Thus, if $E[v\,e]\downarrow_j$ for $j \leq n$ then $E'[v'\,e']\downarrow$. □

Lemma 20. If $v, u \in \mathit{Val}\,(\forall\alpha.\sigma)$, $n < \omega$ and $\forall\tau,\tau' \in \mathit{Type}, \forall R \in \mathit{VRel}(\tau,\tau'), (v\,\tau, u\,\tau') \in [\![\sigma]\!]^{\perp\!\!\perp} R_n$ then $(v, u) \in [\![\forall\alpha.\sigma]\!]_n$

Lemma 21. If $\tau, \sigma \in \mathit{Type}$, $n < \omega$, $(f, f') \in [\![\tau \to \sigma]\!]\, R_n{}^{\perp\!\!\perp}$ and $(e, e') \in [\![\tau]\!]\, R_n{}^{\perp\!\!\perp}$ then

$$\big((\lambda x.x\,e)\, f, (\lambda x.x\,e')\, f'\big) \in [\![\sigma]\!]\, R_n{}^{\perp\!\!\perp}.$$

Proof. This follows from Lemma 13 and Lemma 19. □

Lemma 22 (Functional extensionality)**.** Let $\tau, \sigma \in \mathit{Type}, f, g \in \mathit{Val}(\tau \to \sigma)$ and assume $\forall u \in \mathit{Val}(\tau), f\,u \cong^{ctx}_{\downarrow} g\,u$. Then $f \cong^{ctx}_{\downarrow} g$.

Proof. We will show directly that $\forall n < \omega, (f, g) \in [\![\tau \to \sigma]\!]_n$. To this end take $n < \omega, j \leq n$ and $(v, u) \in [\![\tau]\!]_j$. By the canonical forms lemma $f = \lambda x.e$ and $g = \lambda x.e'$ for some e and e'. We must show $(e[v/x], e'[u/x]) \in [\![\sigma]\!]^{\perp\perp}_j$. So take a $k \leq j$, $(E, E') \in [\![\sigma]\!]^{\perp}_k$ and assume $E[e[v/x]] \downarrow_k$ which is equivalent to $E[f\,v] \downarrow_k$. Proposition 15 shows that $\forall m < \omega, (f, f) \in [\![\sigma \to \tau]\!]^{\perp\perp}_m$ and then using Lemma 21 we can conclude that $E'[f\,u] \downarrow$. Using the assumption and Theorem 17 we get $E'[g\,u] \downarrow$, which concludes the proof. □

Note that extensionality property above is stated for *values* of function type; a more general extensionality property for *expressions* of function type fails. To show that, we first define some abbreviations.

Let $\mathbf{2}$ be the type $\mu\alpha.\mathbf{1} + \mathbf{1}$ and let $\mathbf{true} = \mathtt{in}_1\langle\rangle$ and $\mathbf{false} = \mathtt{in}_2\langle\rangle$ be values of type $\mathbf{2}$. By the canonical forms lemma these two are the only closed values of this type. Let $\Omega_{\mathbf{2}} = \Omega\,\mathbf{2}$. Note that reduction from $\Omega_{\mathbf{2}}$ is deterministic and non-terminating. We first define `if` and `ifz` constructs as

$$\mathtt{if}\,p\,\mathtt{then}\,e\,\mathtt{else}\,e' = \mathtt{let}\ y = p\ \mathtt{in}\ \mathtt{case}\ p\ \mathtt{of}\ \mathtt{in}_1\,x.\,e\,|\,\mathtt{in}_2\,x.\,e'$$

$$\mathtt{ifz}\ p\ \mathtt{then}\ e\ \mathtt{else}\ e' = \mathtt{let}\ y = p\ \mathtt{in}\ \mathtt{case}\ \underline{n}\ \mathtt{of}\ \mathtt{in}_1\,x.\,e\,|\,\mathtt{in}_2\,x.\,e'$$

where x and y are variables not free in e or e'.

Now, to exhibit the failure of a more general extensionality property, let $e = \lambda x.\mathtt{proj}_1\,x\ \mathtt{or}\ \mathtt{proj}_2\,x$ and $e' = \mathtt{ifz}\ ?\ \mathtt{then}\ \lambda x.\mathtt{proj}_1\,x\ \mathtt{else}\ \lambda x.\mathtt{proj}_2\,x$ be two terms of type $\mathbf{2} \times \mathbf{2} \to \mathbf{2}$.

Then it is easy to see that $\forall u \in \mathit{Val}(\mathbf{2} \times \mathbf{2}), (\lambda x.x\,u)\,e \cong^{ctx}_{\downarrow} (\lambda x.x\,u)\,e'$. But on the other hand there is an evaluation context distinguishing the two terms e and e'. The idea is to call the resulting value twice with the same pair and diverge if it produces the same value twice, but return a value if the results of the two calls differ. To this end we first define the function $\mathtt{xor} = \lambda x.\lambda y.\mathtt{if}\,x\,\mathtt{then}\,(\mathtt{if}\,y\,\mathtt{then}\,\mathbf{false}\,\mathtt{else}\,\mathbf{true})\,\mathtt{else}\,y$ and then we define

$$\begin{aligned} E = {} & \mathtt{let}\ x = [\,]\ \mathtt{in} \\ & \mathtt{let}\ y = x\,\langle\mathbf{true}, \mathbf{false}\rangle\ \mathtt{in} \\ & \mathtt{let}\ z = x\,\langle\mathbf{true}, \mathbf{false}\rangle\ \mathtt{in} \\ & \mathtt{let}\ w = x\,\mathtt{xor}\,y\ \mathtt{in} \\ & \mathtt{if}\,w\,\mathtt{then}\,w\,\mathtt{else}\,\Omega_{\mathbf{2}} \end{aligned}$$

We then have $E[e] \downarrow$ but on the other hand, $E[e']$ always diverges, therefore e and e' are not contextually equal.

A similar counter-example can also be exhibited for must-contextual equivalence. Indeed, if we define the function $\mathtt{xnor} = \lambda x.\lambda y.\mathtt{if}\,x\,\mathtt{then}\,y\,\mathtt{else}\,(\mathtt{if}\,y\,\mathtt{then}\,\mathbf{false}\,\mathtt{else}\,\mathbf{true})$ and the evaluation context

$$\begin{aligned} E' = {} & \mathtt{let}\ x = [\,]\ \mathtt{in} \\ & \mathtt{let}\ y = x\,\langle\mathbf{true}, \mathbf{false}\rangle\ \mathtt{in} \\ & \mathtt{let}\ z = x\,\langle\mathbf{true}, \mathbf{false}\rangle\ \mathtt{in} \\ & \mathtt{let}\ w = x\,\mathtt{xnor}\,y\ \mathtt{in} \\ & \mathtt{if}\,w\,\mathtt{then}\,w\,\mathtt{else}\,\Omega_{\mathbf{2}}. \end{aligned}$$

we have that $\forall u \in \mathit{Val}(\mathbf{2} \times \mathbf{2}), (\lambda x.x\,u)\,e \cong^{ctx}_{\Downarrow} (\lambda x.x\,u)\,e'$, but on the other hand $E'[e'] \Downarrow$ but not $E'[e] \Downarrow$.

Finally, we state the expected extensionality property for values of polymorphic type.

Lemma 23 (Extensionality for $\forall$)**.** Let $u, v \in \mathit{Val}\,(\forall\alpha.\sigma)$ and assume $\forall\tau \in \mathit{Type}, u\,\tau \cong^{ctx}_{\downarrow} v\,\tau$. Then $u \cong^{ctx}_{\downarrow} v$.

The proof of this lemma is essentially the same as the proof of Lemma 22.

5. Must equational theory

To define the logical relation for must-approximation, we need to stratify the observables again. We define stratified relations counting all steps ($\not\downarrow$) and one counting only unfold-fold reductions ($\Downarrow$). The latter is used for indexing the logical relations, the former for relating the latter to must-approximation.

For terms e and ordinals β we define $e \not\downarrow_\beta$ by induction on β: $e \not\downarrow_\beta$ if for all e' such that $e \longmapsto e'$ there exists $\nu < \beta$ and $e' \not\downarrow_\nu$. The essential observation is that $\not\downarrow_\beta$ indeed captures must-convergent behaviour.

Lemma 24 (Stratified must-convergence)**.** $e \Downarrow$ if and only if $e \not\downarrow_\beta$ for some $\beta < \omega_1$ (for ω_1 the least uncountable ordinal).

Proof. The proof from left to right is by induction on $e \Downarrow$. By induction hypothesis there exists ordinals $\nu(e') < \omega_1$ for each term e' such that $e \longmapsto e'$. Let $\beta = \bigcup \nu(e')$, then $\beta + 1 < \omega_1$ (since there are only countably many such e' and each $\nu(e')$ is countable) and $e \not\downarrow_{\beta+1}$. The direction from right to left is by induction on β. □

For terms e and ordinals β we define $e \Downarrow_\beta$ by induction on β: $e \Downarrow_\beta$ if for all e' such that $e \rightsquigarrow^1 e'$ there exists $\nu < \beta$ and $e' \Downarrow_\nu$.

Using Lemma 24, we can show:

Lemma 25. $e \Downarrow$ implies $e \Downarrow_\beta$ for some $\beta < \omega_1$.

Logical ω_1-indexed uniform relation for must-approximation. Proposition 24 indicates that logical relations for must-approximation need to be indexed over ω_1. The lifting of value relations $r \in \mathit{VRel}_{\omega_1}(\tau, \tau')$ to relations $r^\perp \in \mathit{SRel}_{\omega_1}(\tau, \tau')$ on evaluation contexts and to relations $r^{\perp\!\perp} \in \mathit{TRel}_{\omega_1}(\tau, \tau')$ on terms is defined with respect to must termination.

$$r^\perp_\beta = \{(E, E') \mid \forall \nu \leq \beta.\ \forall (v, v') \in r_\nu.\ E[v] \Downarrow_\nu \Rightarrow\ E'[v'] \Downarrow\ \}$$
$$r^{\perp\!\perp}_\beta = \{(e, e') \mid \forall \nu \leq \beta.\ \forall (E, E') \in r^\perp_\nu.\ E[e] \Downarrow_\nu \Rightarrow\ E'[e'] \Downarrow\ \}$$

Except for this difference, the relational interpretation $[\![\tau]\!]\,(\vec{r}) \in \mathit{VRel}_{\omega_1}(\tau[\vec{\tau}/\vec{\alpha}], \tau[\vec{\tau}'/\vec{\alpha}])$ is literally the same as in Section 4 and defined by induction on the type $\vec{\alpha} \vdash \tau$, given closed types $\tau_1, \tau_1', \ldots, \tau_k, \tau_k' \in \mathit{Type}$ and relations $r_1 \in \mathit{VRel}_{\omega_1}(\tau_1, \tau_1'), \ldots, r_k \in \mathit{VRel}_{\omega_1}(\tau_k, \tau_k')$:

$$[\![\alpha_i]\!]\,(\vec{r}) = r_i \qquad [\![\tau_1 \times \tau_2]\!]\,(\vec{r}) = [\![\tau_1]\!]\,(\vec{r}) \times [\![\tau_2]\!]\,(\vec{r})$$
$$[\![\mathbf{1}]\!]\,(\vec{r}) = (\mathit{Id}_{\mathbf{1}})_{\beta<\omega_1} \qquad [\![\tau_1 \to \tau_2]\!]\,(\vec{r}) = [\![\tau_1]\!]\,(\vec{r}) \to [\![\tau_2]\!]\,(\vec{r})^{\perp\!\perp}$$
$$[\![\forall\alpha.\tau]\!]\,(\vec{r}) = \forall r.[\![\tau]\!]\,(\vec{r}, r)^{\perp\!\perp} \qquad [\![\mu\alpha.\tau_1 + \ldots + \tau_m]\!]\,(\vec{r}) = \mathit{fix}\,s.\bigcup_j \mathtt{in}_j(\triangleright\,[\![\tau_j]\!]\,(\vec{r}, s))$$

$$\frac{\Delta;\Gamma \vdash v\, v' \lesssim_{\downarrow}^{ctx} v' : \tau_1 \to \tau_2}{\Delta;\Gamma \vdash \mathtt{fix}\,\tau_1\tau_2\, v \lesssim_{\downarrow}^{ctx} v' : \tau_1 \to \tau_2} \qquad \frac{\Delta;\Gamma \vdash v\, v' \lesssim_{\Downarrow}^{ctx} v' : \tau_1 \to \tau_2}{\Delta;\Gamma \vdash \mathtt{fix}\,\tau_1\tau_2\, v \lesssim_{\Downarrow}^{ctx} v' : \tau_1 \to \tau_2}$$

Figure 7: Recursion induction: least prefixed point property of `fix`

Logical must-approximation is defined as follows:

$$\Delta;\Gamma \vdash e \lesssim_{\Downarrow}^{log} e' : \tau \;\text{ where } \Delta = \vec{\alpha}$$
$$\Leftrightarrow\; \forall \vec{\tau}, \vec{\tau}'.\, \forall \vec{r} \in VRel_{\omega_1}(\vec{\tau}, \vec{\tau}').\, \forall \beta < \omega_1.\, \forall (\gamma, \gamma') \in [\![\Gamma]\!]\, \vec{r}_\beta.\; (e[\vec{\tau}/\vec{\alpha}]\gamma, e'[\vec{\tau}'/\vec{\alpha}]\gamma') \in [\![\tau]\!]\, \vec{r}_\beta^{\perp\perp}$$

Proposition 26 (Fundamental property)**.** The relation $\lesssim_{\Downarrow}^{log}$ has the compatibility properties given in Figure 4. In particular, it is reflexive: if $\Delta;\Gamma \vdash e : \tau$ then $\Delta;\Gamma \vdash e \lesssim_{\Downarrow}^{log} e : \tau$.

Proof. The proof is similar to the one for Proposition 15. We give only the case for choice, where we assume $\Delta \vdash \Gamma$ and prove $\Delta;\Gamma \vdash \mathtt{?} \lesssim_{\Downarrow}^{log} \mathtt{?} : \mathtt{nat}$. Suppose $(E, E') \in [\![\mathtt{nat}]\!]\, \vec{r}_\beta^{\perp}$ and $E[\mathtt{?}] \Downarrow_\beta$. We are to show that $E'[\mathtt{?}] \Downarrow$, for which it suffices to show that $E'[\underline{k}] \Downarrow$, for all $k \in \mathbb{N}$. Let $k \in \mathbb{N}$ be arbitrary. Then $E[\mathtt{?}] \longmapsto E[\underline{k}]$ and so $E[\underline{k}] \Downarrow_\beta$. Induction on k shows that $(\underline{k}, \underline{k}) \in [\![\mathtt{nat}]\!]\, \vec{r}_\beta$ and hence the required follows by the assumption on (E, E'). □

Corollary 27. If $v \in Val(\tau)$ then for all $\nu < \omega_1$, $(v, v) \in [\![\tau]\!]_\nu$.

We omit the proof, as it is analogous to the proof of Corollary 16.

Theorem 28 (Coincidence)**.** $\Delta;\Gamma \vdash e \lesssim_{\Downarrow}^{log} e' : \tau$ if and only if $\Delta;\Gamma \vdash e \lesssim_{\Downarrow}^{ciu} e' : \tau$.

Proof. The proof is completely analogous to that of Theorem 17. For the direction from left to right one uses the relationship between $\Downarrow$ and $\Downarrow_\beta$ given by Lemma 25 and then appeals to Proposition 26 and Corollary 27. The direction from right to left uses the fact that $\lesssim_{\Downarrow}^{log}$ is closed under must-CIU approximation. □

Proof of CIU Theorem 8(2). The proof is analogous to that of Theorem 8(1). From the definition, $\lesssim_{\Downarrow}^{ciu}$ is a must-adequate reflexive and transitive relation, by Proposition 26 and Theorem 28 it is also compatible, and thus contained in $\lesssim_{\Downarrow}^{ctx}$. From this containment and the closure of $\lesssim_{\Downarrow}^{ciu}$ under beta conversion it follows that $\lesssim_{\Downarrow}^{ctx}$ has the substitutivity properties in Figure 6. Thus it suffices to prove the containment of $\lesssim_{\Downarrow}^{ctx}$ in $\lesssim_{\Downarrow}^{ciu}$ for closed terms, which is clear by the compatibility and must-adequacy of $\lesssim_{\Downarrow}^{ctx}$. □

6. Applications

This section illustrates how the logical relation characterization of contextual approximation can be used to derive interesting examples and further proof principles. We consider three such applications: a recursion-induction principle for recursively defined functions, syntactic minimal invariance of a recursive type, and an application of relational parametricity to characterize the elements of the type $\forall\alpha.\alpha \times \alpha \to \alpha$.

Recursion-induction. Recall from the introduction that $\mathtt{fix} : \forall\alpha,\beta.((\alpha\to\beta)\to(\alpha\to\beta))\to(\alpha\to\beta)$ is given by the term $\Lambda\alpha,\beta.\lambda f.\delta_f(\mathtt{in}\,\delta_f)$ where δ_f is an abbreviation for the term $\lambda y.\mathtt{case}\ y\ \mathtt{of}\ \mathtt{in}\,y'.\,f(\lambda x.(\lambda r.r\,x)(y'\,y))$. We now prove that $\mathtt{fix}$ is a *least* prefixed point combinator. More precisely, we prove (1) the soundness of the recursion-induction rules in Figure 7; and (2) Proposition 29, which says that $\mathtt{fix}$ behaves as a fixed point combinator for a large class of functionals, including all those of the form $\lambda f.u$, for u a value. (Observe that this class of functionals includes those needed for defining a standard fixed point expression $\mathtt{fix}\,f(x).\,e$ via an application of the fixed point combinator.)

Our recursion-induction rules are mild generalizations of the rules given by Lassen [12] who proved similar results (for a language without polymorphism), when v was restricted to be of the form $\lambda f.u$, for some value u.

We only include the proof for $\lesssim^{ctx}_{\Downarrow}$ and for notational simplicity we assume that the contexts Δ and Γ are empty. We assume the premise of the rule, and to show the conclusion we first prove that $(h,v')\in[\![\tau_1\to\tau_2]\!]_\beta$ where h is $\lambda x.(\lambda r.r\,x)\,(\delta_v\,(\mathtt{in}\,\delta_v))$, for all $\beta<\omega_1$. The result then follows from the agreement of the logical relation with contextual approximation and transitivity, since $\mathtt{fix}\,\tau_1\tau_2\,v\cong^{ctx} v\,h\lesssim^{ctx}_{\Downarrow} v\,v'\lesssim^{ctx}_{\Downarrow} v'$.

To prove $(h,v')\in[\![\tau_1\to\tau_2]\!]_\beta$ we proceed by induction on β and assume that $(h,v')\in[\![\tau_1\to\tau_2]\!]_\nu$, for all $\nu<\beta$; we are then to show that $(h,v')\in[\![\tau_1\to\tau_2]\!]_\beta$. From the canonical forms lemma it follows that v' must be of the form $\lambda x.e'$ for some e'. So let $\beta_1\le\beta$ and $(u,u')\in[\![\tau_1]\!]_{\beta_1}$, then it remains to show $((\lambda r.r\,u)(\delta_v\,(\mathtt{in}\,\delta_v)),e'[u'/x])\in[\![\tau_2]\!]^{\perp\!\perp}_{\beta_1}$.

Suppose $\beta_2\le\beta_1$, $(E,E')\in[\![\tau_2]\!]^{\perp}_{\beta_2}$ and $E[(\lambda r.r\,u)(\delta_v\,(\mathtt{in}\,\delta_v))]\Downarrow_{\beta_2}$; we are to show $E'[e'[u'/x]]\Downarrow$. By (the must-analogue of) Lemma 13 and the fundamental property of the logical relation applied to v we obtain $(E[(\lambda r.r\,u)\,((\lambda x.v\,x)\,[])],E'[(\lambda r.r\,u')\,((\lambda x.v\,x)\,[])])\in[\![\tau_1\to\tau_2]\!]^{\perp}_{\beta_2}$. Then, since $\delta_v\,(\mathtt{in}\,\delta_v)\rightsquigarrow^1 v\,h$ and $(\lambda x.v\,x)\,h\longmapsto v\,h$, we have $E[(\lambda r.r\,u)(v\,h)]\Downarrow_{\beta_3}$ for $\beta_3<\beta_2\le\beta$, and hence also $E'[(\lambda r.r\,u')\,(v\,v')]\Downarrow$ by induction hypothesis.

By the premise and Theorem 28 we have that $v\,v'$ CIU-approximates v', and thus we get $E'[(\lambda r.r\,u')\,v']\Downarrow$. Finally, since $(\lambda r.r\,u')\,v'\longmapsto^* e'[u'/x]$ we obtain the required $E'[e'[u'/x]]\Downarrow$.

Proposition 29. Let $\tau,\tau'\in \mathit{Type}$, $f\in \mathit{Val}((\tau\to\tau')\to\tau\to\tau')$. If for all $g\in\mathit{Val}(\tau\to\tau')$ there exists a value f_g such that $f\,g\overset{p}{\rightsquigarrow} f_g$, then $\mathtt{fix}\,\tau\,\tau'\,f$ is a fixed point of f, i.e., $f(\mathtt{fix}\,\tau\,\tau'\,f)\cong^{ctx}_{\Downarrow}\mathtt{fix}\,\tau\,\tau'\,f$ and $f(\mathtt{fix}\,\tau\,\tau'\,f)\cong^{ctx}_{\downarrow}\mathtt{fix}\,\tau\,\tau'\,f$.[1]

Proof. Fix $\tau,\tau'\in\mathit{Type}$ and $f\in\mathit{Val}((\tau\to\tau')\to\tau\to\tau')$. Let $\mathtt{fix}_{\tau,\tau'}=\lambda f.\delta_f\,\mathtt{in}\delta_f$. It is then easy to see that $\mathtt{fix}_{\tau,\tau'}f\overset{p}{\rightsquigarrow} f\,h$ where $h=\lambda x.(\lambda r.r\,x)(\mathtt{fix}_{\tau,\tau'}\,f)$. Similarly, $f\,(\mathtt{fix}_{\tau,\tau'}\,f)\overset{p}{\rightsquigarrow} f\,(f\,h)$.

Let $E\in\mathit{Stk}(\tau')$ and $v\in\mathit{Val}(\tau)$. We have the following sequence of equivalences

$$\begin{aligned} E[h\,v]\Downarrow &\iff E[(\lambda r.r\,v)\,(\mathtt{fix}_{\tau,\tau'}\,f)]\Downarrow\\ &\iff E[(\lambda r.r\,v)\,(f\,h)]\Downarrow\\ &\iff \big(\forall f',f\,h\longmapsto^* f'\Rightarrow E[f'\,v]\Downarrow\big)\end{aligned}$$

[1] We have abused notation slightly by writing $\mathtt{fix}\,\tau\,\tau'\,f$ instead of $\mathtt{let}\ x=\mathtt{fix}\,\tau\ \mathtt{in}\ \mathtt{let}\ y=x\,\tau'\ \mathtt{in}\ y\,f$, but the former is more readable.

and similarly, for may equivalence,

$$\begin{aligned} E[h\,v]\downarrow &\iff E[(\lambda r.r\,v)\,(\mathtt{fix}_{\tau,\tau'}\,f)]\downarrow \\ &\iff E[(\lambda r.r\,v)\,(f\,h)]\downarrow \\ &\iff \left(\exists f'_v, f\,h \longmapsto^* f'_v \wedge E[f'\,v]\downarrow\right) \end{aligned}$$

Suppose further that $f\,h \overset{p}{\rightsquigarrow} f_h$, for some value f_h. This implies that $f\,h$ reduces to a unique value. Then the above equivalences reduce to $E[h\,v]\Downarrow \iff E[f_h\,v]\Downarrow$ and $E[h\,v]\downarrow \iff E[f_h\,v]\downarrow$. Since v and E were arbitrary, we can use Lemma 22 and its must-analogue to conclude that h is may and must equivalent to f_h and thus to $f\,h$, which concludes the proof. □

Syntactic minimal invariance. Consider the type $\tau = \mu\alpha.\mathtt{nat} + \alpha \to \alpha$. Let $id = \lambda x.x$ and consider the term

$$f \;\equiv\; \lambda h, x.\mathtt{case}\ x\ \mathtt{of}\ \mathtt{in}_1\,y.\,\mathtt{in}_1\,y\ \mid\ \mathtt{in}_2\,g.\,\mathtt{in}_2\,\lambda y.h(g(h\,y))\ .$$

We shall show that $\mathtt{fix}\,\tau\tau\,f \cong^{ctx} id : \tau \to \tau$. This equivalence corresponds to the characterization of solutions to recursive domain equations as minimal invariants in domain-theoretic work [17], from which Pitts derives several (co-)induction principles. Our proof is similar to the one by Dreyer, Ahmed, and Birkedal [8] for a language without nondeterminism.

By the soundness of the call-by-value beta- and eta-laws for contextual equivalence (Figure 5) and the transitivity of $\lesssim^{ctx}$, it is easy to see that $f\,id \cong^{ctx} id : \tau \to \tau$. The recursion-induction principle therefore yields $\mathtt{fix}\,\tau\tau\,f \lesssim^{ctx} id : \tau \to \tau$.

For the reverse approximation we first show $id \lesssim^{log}_{\Downarrow} h : \tau \to \tau$ where h is again the term $\lambda x.(\lambda r.r\,x)(\delta_f\,(\mathtt{in}\,\delta_f))$. We show this by proving $(id, h) \in [\![\tau \to \tau]\!]_\beta$ for all $\beta < \omega_1$ by induction on β. (The case for may-approximation is similar.)

Thus it suffices to show, for all $\nu \le \beta$, for all $(v, v') \in [\![\tau]\!]_\nu$, $(id\,v, h\,v') \in [\![\tau]\!]^{\perp\!\perp}_\nu$.

Since $[\![\tau]\!] = \mathtt{in}_1(\triangleright [\![\mathtt{nat}]\!]) \cup \mathtt{in}_2(\triangleright [\![\tau \to \tau]\!])$ there are two cases to consider:

- Case $(v, v') \in \mathtt{in}_1(\triangleright [\![\mathtt{nat}]\!])_\nu$. Then there exist $u, u' \in Val(\mathtt{nat})$ such that $v = \mathtt{in}_1\,u$, $v' = \mathtt{in}_1\,u'$ and $(u, u') \in [\![\mathtt{nat}]\!]_{\nu'}$, for all $\nu' < \nu \le \beta$. Given $(E, E') \in [\![\tau]\!]^\perp_\nu$ such that $E[id\,v]\Downarrow_\nu$, it suffices to show that $E[h\,v']\Downarrow$, which follows using the must-analogues of Lemmas 14 and 11 since $h\,v' \overset{p}{\rightsquigarrow} v'$ and $(v, v') \in [\![\tau]\!]_\nu$ by assumption.
- Case $(v, v') \in \mathtt{in}_2(\triangleright [\![\tau \to \tau]\!])_\nu$. Then there exist $g, g' \in Val(\tau \to \tau)$ such that $v = \mathtt{in}_2\,g$, $v' = \mathtt{in}_2\,g'$ and $(g, g') \in [\![\tau \to \tau]\!]_{\nu'}$ for all $\nu' < \nu \le \beta$. We are to show that $(id\,v, h\,v') \in [\![\tau]\!]^{\perp\!\perp}_\nu$. Since $h\,v' \overset{p}{\rightsquigarrow} \mathtt{in}_2(\lambda y.h(g'(h\,y)))$ and $id\,v \overset{p}{\rightsquigarrow}{}^0\ v$, by must-analogue of Lemma 14, it suffices to show $(\mathtt{in}_2(g), \mathtt{in}_2(\lambda y.h(g'(h\,y)))) \in [\![\tau]\!]^{\perp\!\perp}_\nu$. Hence it suffices to show $(g, \lambda y.h(g'(h\,y))) \in [\![\tau \to \tau]\!]^{\perp\!\perp}_{\nu_1}$, for all $\nu_1 < \nu$. Pick $\nu_1 < \nu$. By the must-analogue of Lemma 14 it suffices to show, for all $\nu_2 \le \nu_1$,

  $$\forall (u, u') \in [\![\tau]\!]_{\nu_2}\,.\,(g(u), h(g'(h(u')))) \in [\![\tau]\!]^{\perp\!\perp}_{\nu_2}.$$

  To this end, let $(u, u') \in [\![\tau]\!]_{\nu_2}$ and pick $\nu_3 \le \nu_2$ and suppose that $(E, E') \in [\![\tau]\!]^\perp_{\nu_3}$. By the induction hypothesis and the must-analogue of Lemma 14, we get $(u, h(u')) \in [\![\tau]\!]^{\perp\!\perp}_{\nu_3}$. Hence by the must-analogue of Lemma 21, we get $(g(u), g'(h(u'))) \in [\![\tau]\!]^{\perp\!\perp}_{\nu_3}$, and thus it suffices to show that $(E, E'[h\,[]]) \in [\![\tau]\!]^\perp_{\nu_3}$. So let $\nu_4 \le \nu_3$ and take

$(w, w') \in [\![\tau]\!]_{\nu_4}$. We are to show that if $E[w] \Downarrow_{\nu_4}$, then $E'[h\, w'] \Downarrow$. By the induction hypothesis and the must-analogue of Lemma 14, $(w, h\, w') \in [\![\tau]\!]_{\nu_4}^{\perp\!\!\perp}$, from which the required follows by the assumption on (E, E').

By Theorem 28 and the CIU theorem, $id \lesssim_{\Downarrow}^{log} h : \tau \to \tau$ implies $id \lesssim_{\Downarrow}^{ctx} h : \tau \to \tau$. Since $id \cong^{ctx} f\, id : \tau \to \tau$ and $f\, h \cong^{ctx} \mathtt{fix}\,\tau\tau\, f : \tau \to \tau$ we obtain $id \lesssim_{\Downarrow}^{ctx} \mathtt{fix}\,\tau\tau\, f : \tau \to \tau$ by compatibility and transitivity of must-contextual equivalence.

Parametricity. We will now characterize the elements of the type $\forall\alpha.\alpha \times \alpha \to \alpha$, using relational parametricity. The main result is expressed as Theorem 33; we first start with some lemmas. We only state and prove the results for must-contextual equivalence; for may-contextual equivalence the properties and proofs are analogous.

Lemma 30. Let $v \in \mathit{Val}\,(\forall\alpha.\alpha \times \alpha \to \alpha)$. If there exists a $\tau \in \mathit{Type}$ such that $v\,\tau$ may-diverges then $v \cong_{\Downarrow}^{ctx} (\Lambda\alpha.\Omega\,(\alpha \times \alpha \to \alpha))$.

Proof. Let τ be such that $v\,\tau$ may-diverges. By the must-analogue of Lemma 18 we have $\forall\tau' \in \mathit{Type}, \forall R, \forall\nu < \omega_1,\ (v\,\tau', v\,\tau) \in [\![\alpha \times \alpha \to \alpha]\!]\, R_\nu{}^{\perp\!\!\perp}$. This implies that for all $\tau' \in \mathit{Type}$, $v\,\tau'$ may-diverges (because the empty context is always related to itself, for instance).

Using the must-analogue of Lemma 20 we can thus conclude that $v \lesssim_{\Downarrow}^{log} \Lambda\alpha.\Omega\,(\alpha\times\alpha \to \alpha)$ and $\Lambda\alpha.\Omega\,(\alpha \times \alpha \to \alpha) \lesssim_{\Downarrow}^{log} v$. Theorems 28 and 8 then finish the proof. □

Lemma 31. Let $v \in \mathit{Val}\,(\forall\alpha.\alpha \times \alpha \to \alpha)$. If for all $\tau \in \mathit{Type}$, the expression $v\,\tau$ must-converges, and there exist a τ and $u \in \mathit{Val}(\tau \times \tau)$ such that $(\lambda x.x\, u)\,(v\,\tau)$ may-diverges, then for all $\tau' \in \mathit{Type}$ and for all $u' \in \mathit{Val}(\tau' \times \tau')$, $(\lambda x.x\, u')\,(v\,\tau') \cong_{\Downarrow}^{ctx} \Omega\,\tau'$.

Proof. Let $\tau' \in \mathit{Type}, u' \in \mathit{Val}\,(\tau' \times \tau')$. By the canonical forms lemma $u = \langle u_1, u_2\rangle$ for some $u_1, u_2 \in \mathit{Val}(\tau)$ and $u' = \langle u_1', u_2'\rangle$ for some $u_1', u_2' \in \mathit{Val}(\tau')$. Let $R_\nu = \{(u_1', u_1), (u_2', u_2)\}$ for $\nu < \omega_1$. It is easy to see that $(u', u) \in [\![\alpha \times \alpha]\!]\, R_\nu$. The must-analogues of Lemmas 21 and 18 then imply that $((\lambda x.x\, u')\,(v\,\tau'), (\lambda x.x\, u)\,(v\,\tau)) \in [\![\alpha]\!]\, R_\nu{}^{\perp\!\!\perp}$. This in particular means that $(\lambda x.x\, u')\,(v\,\tau')$ may-diverges. Since $\tau' \in \mathit{Type}$ and $u' \in \mathit{Val}(\tau' \times \tau')$ were arbitrary, we have that for all $\tau' \in \mathit{Type}$ and $u' \in \mathit{Val}(\tau' \times \tau')$, $(\lambda x.x\, u')\,(v\,\tau') \cong_{\Downarrow}^{ctx} \Omega\,\tau'$.

□

Lemma 32. Let $v \in \mathit{Val}\,(\forall\alpha.\alpha \times \alpha \to \alpha)$. If for all $\tau \in \mathit{Type}$ and for all $u \in \mathit{Val}(\tau \times \tau)$, the expression $(\lambda x.x\, u)\,(v\,\tau)$ must-converges, one of the following three cases holds

(1) $\forall\tau \in \mathit{Type}, \forall t, s \in \mathit{Val}(\tau), (\lambda x.x\,\langle t, s\rangle)\,(v\,\tau) \cong_{\Downarrow}^{ctx} t$
(2) $\forall\tau \in \mathit{Type}, \forall t, s \in \mathit{Val}(\tau), (\lambda x.x\,\langle t, s\rangle)\,(v\,\tau) \cong_{\Downarrow}^{ctx} s$
(3) $\forall\tau \in \mathit{Type}, \forall t, s \in \mathit{Val}(\tau), (\lambda x.x\,\langle t, s\rangle)\,(v\,\tau) \cong_{\Downarrow}^{ctx} t\ \mathtt{or}\ s$

Proof. Let $\tau \in \mathit{Type}$. Must-analogues of Lemmas 18, 21 and the definitions of relational actions show that

$$\forall\nu < \omega_1, \forall R \in \mathit{VRel}(\mathbf{2}, \tau), \forall (b, w) \in (R \times R)_\nu, ((\lambda x.x\, b)\,(v\,\mathbf{2}), (\lambda x.x\, w)\,(v\,\tau)) \in R_\nu{}^{\perp\!\!\perp} \quad (6.1)$$

and

$$\forall\nu < \omega_1, \forall S \in \mathit{VRel}(\tau, \mathbf{2}), \forall (w, b) \in (S \times S)_\nu, ((\lambda x.x\, w)\,(v\,\tau), (\lambda x.x\, b)\,(v\,\mathbf{2})) \in S_\nu{}^{\perp\!\!\perp}. \quad (6.2)$$

By assumption there exists a $s \in \mathit{Val}(\mathbf{2})$, such that $(\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\tau) \longmapsto^* s$. By the canonical forms lemma, s can only be **true** or **false** and based on this, we consider three different options.

In all the cases let $t, s \in \mathit{Val}(\tau)$ and define $R = \{(\mathbf{true}, t), (\mathbf{false}, s)\}$ and $S = \{(t, \mathbf{true}), (s, \mathbf{false})\}$. Note that the cases don't depend on t, s, R or S.

- $(\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\mathbf{2}) \longmapsto^* \mathbf{true}$ but $\neg\,((\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\mathbf{2}) \longmapsto^* \mathbf{false})$. In this case, we wish to show that $(\lambda x.x\,\langle t, s\rangle)\,(v\,\tau) \cong^{ctx}_{\Downarrow} t$ and we again show this by showing that the two terms are must-CIU equivalent.

  Let $E \in \mathit{Stk}(\tau)$ and assume $E\,[(\lambda x.x\,\langle t, s\rangle)\,(v\,\tau)] \Downarrow$. This implies there exists a $\nu < \omega_1$, such that $E\,[(\lambda x.x\,\langle t, s\rangle)\,(v\,\tau)] \Downarrow_\nu$. We must show $E[t] \Downarrow$. Suppose instead that $\neg(E[t] \Downarrow)$. Then $\forall \beta < \omega_1, (E, (\lambda x.\mathtt{if}\ x\ \mathtt{then}\ \Omega_{\mathbf{2}}\ \mathtt{else}\ x)\,[]) \in {S_\beta}^\perp$. Instantiating (6.2) with the above defined S and any $\beta \geq \nu$ shows that

  $$(\lambda x.\mathtt{if}\ x\ \mathtt{then}\ \Omega_{\mathbf{2}}\ \mathtt{else}\ x)\,((\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\mathbf{2})) \Downarrow,$$

  but we have assumed that $(\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\mathbf{2})] \longmapsto^* \mathbf{true}$. This therefore leads to a contradiction, stemming from the assumption that $E[t]$ may-diverges. This shows one direction of may-CIU approximation.

  For the other, again let $E \in \mathit{Stk}(\tau)$ and now assume $E[t] \Downarrow$. It follows that for all $\beta < \omega_1$, $((\lambda x.\mathtt{if}\ x\ \mathtt{then}\ x\ \mathtt{else}\ \Omega_{\mathbf{2}})\,[], E) \in {R_n}^\perp$. We now instantiate (6.1) with our particular R and $\nu < \omega_1$, such that $(\lambda x.\mathtt{if}\ x\ \mathtt{then}\ x\ \mathtt{else}\ \Omega_{\mathbf{2}})\ ((\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\mathbf{2})) \Downarrow_\nu$. Such a ν exists since we have assumed that $(\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\mathbf{2})$ must-converges and $(\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\mathbf{2}) \longmapsto^* \mathbf{true}$ but it does not reduce to **false** and so this implies $E\,[(\lambda x.x\,\langle t, s\rangle)\,(v\,\tau)] \Downarrow$, which concludes this part of the proof.
- $(\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\mathbf{2}) \longmapsto^* \mathbf{false}$ but $\neg\,((\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\mathbf{2}) \longmapsto^* \mathbf{true})$. In this case we show that $(\lambda x.x\,\langle t, s\rangle)\,(v\,\tau) \cong^{ctx}_{\Downarrow} s$. The proof of this is completely analogous to the one for the previous case, so we omit the details here.
- $(\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\mathbf{2}) \longmapsto^* \mathbf{false}$ *and* $(\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\mathbf{2}) \longmapsto^* \mathbf{true}$. In this case, we wish to show that $(\lambda x.x\,\langle t, s\rangle)\,(v\,\tau) \cong^{ctx}_{\Downarrow} t\ \mathtt{or}\ s$. We again do this by showing must-CIU equivalence in two steps.

  Let $E \in \mathit{Stk}(\tau)$ and assume $E\,[(\lambda x.x\,\langle t, s\rangle)\,(v\,\tau)] \Downarrow$. This implies there exists a $\nu < \omega_1$, such that $E\,[(\lambda x.x\,\langle t, s\rangle)\,(v\,\tau)] \Downarrow_\nu$. We must show $E[t\ \mathtt{or}\ s] \Downarrow$. Suppose instead that $\neg(E[t\ \mathtt{or}\ s] \Downarrow)$. This implies $\neg(E[t] \Downarrow)$ or $\neg(E[s] \Downarrow)$ (or both). Without loss of generality suppose $\neg(E[t] \Downarrow)$. This implies $\forall \beta < \omega_1, (E, (\lambda x.\mathtt{if}\ x\ \mathtt{then}\ \Omega_{\mathbf{2}}\ \mathtt{else}\ x)\,[]) \in {S_\beta}^\perp$. Instantiating (6.2) with the above defined S and any $\beta \geq \nu$ leads to a contradiction, since it implies that

  $$(\lambda x.\mathtt{if}\ x\ \mathtt{then}\ \Omega_{\mathbf{2}}\ \mathtt{else}\ x)\,(\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\mathbf{2}) \Downarrow$$

  but since $(\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\mathbf{2}) \longmapsto^* \mathbf{true}$, this cannot be.

  For the other direction, let $E \in \mathit{Stk}(\tau)$ and now assume $E[t\ \mathtt{or}\ s] \Downarrow$. This implies that $E[t] \Downarrow$ and $E[s] \Downarrow$, which further implies that $\forall \beta < \omega_1, ([], E) \in {R_n}^\perp$. If we instantiate (6.1) with our particular R and $\nu < \omega_1$, such that

  $$(\lambda x.x\,\langle \mathbf{true}, \mathbf{false}\rangle)\,(v\,\mathbf{2}) \Downarrow_\nu,$$

  (such a ν again exists by assumption) we have $E\,[(\lambda x.x\,\langle t, s\rangle)\,(v\,\tau)] \Downarrow$ which concludes the proof. □

Theorem 33. If $v \in \mathit{Val}\,(\forall\alpha.\alpha \times \alpha \to \alpha)$ then exactly one of the following holds

- $v \cong^{ctx}_{\Downarrow} \Lambda\alpha.\Omega\,(\alpha \times \alpha \to \alpha)$
- $\forall\tau \in \mathit{Type}, \forall t \in \mathit{Val}(\tau \times \tau), (\lambda x.x\,t)\,(v\,\tau) \cong^{ctx}_{\Downarrow} \Omega\,\tau$
- $\forall\tau \in \mathit{Type}, \forall t, s \in \mathit{Val}(\tau), (\lambda x.x\,\langle t, s\rangle)\,(v\,\tau) \cong^{ctx}_{\Downarrow} t$
- $\forall\tau \in \mathit{Type}, \forall t, s \in \mathit{Val}(\tau), (\lambda x.x\,\langle t, s\rangle)\,(v\,\tau) \cong^{ctx}_{\Downarrow} s$
- $\forall\tau \in \mathit{Type}, \forall t, s \in \mathit{Val}(\tau), (\lambda x.x\,\langle t, s\rangle)\,(v\,\tau) \cong^{ctx}_{\Downarrow} t\ \mathtt{or}\ s$

If, further, $\forall\tau \in \mathit{Type}, \exists v_\tau \in \mathit{Val}(\tau \times \tau \to \tau)$, such that $v\,\tau \overset{p}{\rightsquigarrow} v_\tau$ then one of the following holds

- $v \cong^{ctx}_{\Downarrow} \Lambda\alpha.\lambda x.\Omega\,\alpha$
- $v \cong^{ctx}_{\Downarrow} \Lambda\alpha.\lambda x.\mathtt{proj}_1\,x$
- $v \cong^{ctx}_{\Downarrow} \Lambda\alpha.\lambda x.\mathtt{proj}_2\,x$
- $v \cong^{ctx}_{\Downarrow} \Lambda\alpha.\lambda x.\mathtt{proj}_1\,x\ \mathtt{or}\ \mathtt{proj}_2\,x.$

Proof. The first part of the theorem only summarizes Lemmas 30, 31, 32.

For the second part, we consider cases as in the first part and as all of them are analogous, we only show the last one. Using the must-analogue of Lemma 23 we only need to show

$$\forall\tau \in \mathit{Type}, v\,\tau \cong^{ctx}_{\Downarrow} \lambda x.\mathtt{proj}_1\,x\ \mathtt{or}\ \mathtt{proj}_2\,x$$

It is easy to show that $v\,\tau \cong^{ctx}_{\Downarrow} v_\tau$, but note that the fact that $v\,\tau$ reduces to v_τ using only pure reductions is crucial, as it implies that this v_τ is the unique value of $v\,\tau$. Using transitivity of $\cong^{ctx}_{\Downarrow}$ it thus suffices to show $v_\tau \cong^{ctx}_{\Downarrow} \lambda x.\mathtt{proj}_1\,x\ \mathtt{or}\ \mathtt{proj}_2\,x$. From the first part we have that

$$\forall t, s \in \mathit{Val}(\tau), (\lambda x.x\,\langle t, s\rangle)\,(v\,\tau) \cong^{ctx}_{\Downarrow} t\ \mathtt{or}\ s$$

and it is also immediate that $(\lambda x.\mathtt{proj}_1\,x\ \mathtt{or}\ \mathtt{proj}_2\,x)\,\langle t, s\rangle \cong^{ctx}_{\Downarrow} t\ \mathtt{or}\ s$ which together imply

$$\forall t, s \in \mathit{Val}(\tau), v_\tau\,\langle t, s\rangle \cong^{ctx}_{\Downarrow} (\lambda x.\mathtt{proj}_1\,x\ \mathtt{or}\ \mathtt{proj}_2\,x)\,\langle t, s\rangle.$$

The canonical forms lemma shows that such pairs are the only possible values of type $\tau \times \tau$ and so the must-analogue of Lemma 22 implies that $v_\tau \cong^{ctx}_{\Downarrow} \lambda x.\mathtt{proj}_1\,x\ \mathtt{or}\ \mathtt{proj}_2\,x$, as required. □

Note that the example in Section 4, used to demonstrate the lack of extensionality for expression of function type, also demonstrates that it is not the case that v is contextually equivalent to one of the functions listed in the above theorem without some further restrictions, such as the one used for the second part of the theorem.

7. Comparison to conference paper

A preliminary version of this paper was presented at the 20th Annual Conference on Computer Science Logic (CSL'11), 12-15 September 2011. The present version corrects some mistakes in the proof of syntactic minimal invariance in the earlier conference paper. This was done by changing the counting of steps so that the only steps that count in the indexing of the logical relations are unfold-fold reductions. That suffices for well-definedness of the logical relation, and means that the approximation relations are closed under pure zero-step reductions on the left and under pure arbitrary reductions on the right, which was implicitly used in the wrong proof in the conference paper. This change means that

the precise formulation of several lemmas have changed. Moreover, we have changed the parametricity example to a more interesting one involving nondeterminism.

Acknowledgements

We gratefully acknowledge the comments and suggestions from the referees. In particular we thank them for discovering the problem with the earlier proof of syntactic minimal invariance and for suggesting a better example of relational parametricity.

References

[1] G. Agha, I. A. Mason, S. F. Smith, and C. L. Talcott. A foundation for actor computation. *J. Funct. Program.*, 7(1):1–72, 1997.
[2] A. W. Appel and D. A. McAllester. An indexed model of recursive types for foundational proof-carrying code. *ACM Trans. Program. Lang. Syst.*, 23(5):657–683, 2001.
[3] K. R. Apt and G. D. Plotkin. Countable nondeterminism and random assignment. *J. ACM*, 33(4):724–767, 1986.
[4] L. Birkedal and R. Harper. Relational interpretations of recursive types in an operational setting. *Inf. Comput.*, 155(1-2):3–63, 1999.
[5] L. Birkedal, B. Reus, J. Schwinghammer, K. Støvring, J. Thamsborg, and H. Yang. Step-indexed Kripke models over recursive worlds. In *POPL*, pages 119–132, 2011.
[6] P. Di Gianantonio, F. Honsell, and G. D. Plotkin. Uncountable limits and the lambda calculus. *Nord. J. Comput.*, 2(2):126–145, 1995.
[7] P. Di Gianantonio and M. Miculan. Unifying recursive and co-recursive definitions in sheaf categories. In *FOSSACS*, pages 136–150, 2004.
[8] D. Dreyer, A. Ahmed, and L. Birkedal. Logical step-indexed logical relations. *Logical Methods in Computer Science*, 7(2), 2011.
[9] D. Dreyer, G. Neis, and L. Birkedal. The impact of higher-order state and control effects on local relational reasoning. In *ICFP*, pages 143–156, 2010.
[10] P. Johann, A. Simpson, and J. Voigtländer. A generic operational metatheory for algebraic effects. In *LICS*, pages 209–218, 2010.
[11] J. Laird. Bidomains and full abstraction for countable nondeterminism. In *FOSSACS*, pages 352–366, 2006.
[12] S. B. Lassen. *Relational Reasoning about Functions and Nondeterminism*. PhD thesis, University of Aarhus, 1998.
[13] S. B. Lassen and A. Moran. Unique fixed point induction for McCarthy's amb. In *MFCS*, pages 198–208, 1999.
[14] S. B. Lassen and C. Pitcher. Similarity and bisimilarity for countable non-determinism and higher-order functions. *Electr. Notes Theor. Comput. Sci.*, 10, 1997.
[15] P. B. Levy. Infinitary Howe's method. In *CMCS*, pages 85–104, 2006.
[16] I. A. Mason and C. L. Talcott. Equivalence in functional languages with effects. *J. Funct. Program.*, 1(3):287–327, 1991.
[17] A. M. Pitts. Relational properties of domains. *Inf. Comput.*, 127(2):66–90, 1996.
[18] A. M. Pitts. Typed operational reasoning. In B. C. Pierce, editor, *Advanced Topics in Types and Programming Languages*, chapter 7, pages 245–289. MIT Press, 2005.
[19] A. M. Pitts. Step-indexed biorthogonality: a tutorial example. In *Modelling, Controlling and Reasoning About State*, Dagstuhl Seminar Proceedings, 2010.
[20] D. Sabel and M. Schmidt-Schauß. A call-by-need lambda calculus with locally bottom-avoiding choice: context lemma and correctness of transformations. *Math. Struct. Comp. Sci.*, 18(3):501–553, 2008.