Relational Interpretations of Recursive Types in an Operational Setting

Lars Birkedal Robert Harper

April 24, 1998

CMU-CS-98-125

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted for publication to *Information and Computation.* A summary of this paper appeared in TACS '97.

This research was supported in part by the Isaac Newton Institute for Mathematical Sciences at Cambridge University. Robert Harper is supported by the National Science Foundation under Grant No. CCR-95-2674. Lars Birkedal is supported in part by the Danish National Research Council and in part by the National Science Foundation under Grant No. CCR-9409997.

Keywords: Type theory, operational semantics, logical relations, equational logics, compiler correctness, continuations.

Abstract

Relational interpretations of type systems are useful for establishing properties of programming languages. For languages with recursive types it is usually difficult to establish the existence of a relational interpretation. The usual approach is to give a denotational semantics of the language in a domain-theoretic model given by an inverse limit construction, and to construct relations over the model by a similar inverse limit construction. However, in passing to a denotational semantics we incur the obligation to prove its adequacy with respect to the operational semantics of the language, which is itself often proved using a relational interpretation of types! In this paper we investigate the construction of relational interpretations of recursive types in a purely operational setting, drawing on recent ideas from domain theory and operational semantics as a guide. We establish a *syntactic minimal invariance* property for an ML-like language with a recursive type that is a syntactic analogue of Freyd's universal characterization of the canonical solution of a domain equation. As Pitts has shown in the setting of domans, minimal invariance suffices to establish the existence of relational interpretations of recursive types. We give two applications of this construction. First, we derive a notion of *logical equivalence* for expressions of the language that we show coincides with contextual equivalence and which, by virtue of its construction, validates useful induction and coinduction principles for reasoning about the recursive type. Second, we give a relational proof of correctness of the continuation-passing transformation, which is used in some compilers for functional languages. The proof relies on the construction of a family of simulation relations whose existence follows from syntactic minimal invariance.

1 Introduction

The interpretation of types by relations is a fundamental technique in the study of type systems (see, for example, Mitchell's survey [14] and monograph [15] for examples and references to the literature). The general idea is to associate to each type a relation over a suitable value space in such a way that well-typed terms are related appropriately by the interpretation. In many cases the existence of a relational interpretation is established by induction on the structure of types, but in more complex languages with impredicative polymorphism or (not necessarily positive) recursive types, more sophisticated methods are required.

In the case of impredicative polymorphism the method of candidates introduced by Girard [8] may be used to construct a relational interpretation. For recursive types the usual approach is to pass to a domain-theoretic model of the language and to exploit the structure of the model to build the required system of relations. In practice the model (such as Scott's D_∞) is obtained as the inverse limit of a system of domains, and the required relational interpretation is obtained by exploiting the structure of the model arising from this construction.

The denotational approach has been successfully used for a number of problems, including Reynolds' proof of correctness of the continuation-passing transformation used in some compilers for functional languages [20]. A disadvantage of this approach is that one must also prove the correctness (adequacy) of the denotational semantics of the language, which is itself often established using a relational interpretation of types [19, 17]! Moreover, since the construction is carried out for a specific model of the language, it is not clear *a priori* to what extent the specific model affects the result.

The latter question was recently addressed by Pitts [17] who showed that Freyd's universal characterization of the solution of a domain equation by the *minimal invariant* property [6, 5, 7] is sufficient to validate the construction of a wide class of relational interpretations of recursive types. The starting point for the present work is the observation that for a sufficiently rich language with recursive functions and recursive types the minimal invariance property of the model is expressible entirely in terms of the language itself by an equation stating that a particular recursively-defined function is denotationally equivalent to the identity function on the recursive type. This opens the door to the construction of relational interpretations without the passage to a denotational semantics. The key is to establish the minimal invariance property up to contextual, rather than denotational, equivalence. With this *syntactic minimal invariance* property in hand we may exploit

Pitts's results to construct relational interpretations over contextual equivalence classes of expressions entirely at the level of the language itself.

Since contextual equivalence is a congruence, it induces a compositional interpretation that may be seen as a form of denotational semantics, *albeit* one that is adequate by construction. This suggests that our approach may be seen as a particular instance of the standard construction. However, as Mason, Smith, and Talcott have shown [11], the interpretation induced by taking contextual equivalence classes does not yield a domain in the conventional sense since, for example, not all chains have least upper bounds. Thus the operational approach to interpreting recursive types as relations differs fundamentally from the denotational method.

We study the construction of relational interpretations for an ML-like language, $\mathcal{L}$, with recursive functions and one recursive type. The operational semantics of the language specifies a call-by-value, or "eager", evaluation strategy, as in Standard ML [13]. We make no restrictions on the occurrence of the recursively-defined type in its definition — both positive and negative occurrences are permitted.

The proof of syntactic minimal invariance for $\mathcal{L}$ relies on a characterization of contextual equivalence given by Mason, Smith, and Talcott [11], called *experimental equivalence*. The primary interest in this notion of equivalence is that it coincides with contextual equivalence and supports a relatively straightforward proof of syntactic minimal invariance. Other, equivalent, characterizations are also available, but these do not appear to significantly simplify the argument.

We give two examples of the use of relational interpretations to analyze properties of the language $\mathcal{L}$. First, we derive another characterization of contextual equivalence, called *logical equivalence*, that validates induction and coinduction principles for reasoning about values of the recursive type. We illustrate the use of logical equivalence with two small examples based on similar examples given by Pitts [16]. Second, we give a relational proof of correctness of the continuation-passing (cps) transform introduced by Fischer [4] and Plotkin [18] and studied by Reynolds [20]. The proof relies on the construction of a relational interpretation of $\mathcal{L}$ that establishes a correspondence between the evaluation of a program and its continuation-passing transform. This generalizes Reynolds' result [20] to the case of a typed language with an arbitrary recursive type, while avoiding the need to consider a denotational semantics for $\mathcal{L}$.

This paper is organized as follows. In Section 2 we define the syntax of the language $\mathcal{L}$, define the operational semantics and show some standard typing properties, including type soundness. Then in Section 3 we define the notion of experimental equivalence, with which we shall be working in the remainder of the paper. The main result of this section is the proof of syntactic minimal invariance based on a technique introduced by Mason, Talcott, and Smith [11]. In Section 4 we define a universe of admissible relations over contextual equivalence classes of closed expressions. We also define relational operators corresponding to the type constructors of the language and show that they preserve admissibility. The relational constructors are used in Section 5 where we construct a relational interpretation of types using the method described above. In Section 6 we show how to use our method to give a relational interpretation which coincides with contextual equivalence. In Section 7 we apply the method to give a proof of correctness of the cps transformation. Finally, in Section 8 we discuss related work, and in Section 9 we conclude.

2 The Language

The language, $\mathcal{L}$, is a simply-typed fragment of ML with one top-level recursive type. We let x and f range over a set Var of program variables. The syntax of the language is given by the following grammar:

$$
\begin{array}{llll}
\textit{Types} & \tau & ::= & 0 \mid 1 \mid \rho \mid \tau_1 + \tau_2 \mid \tau_1 \times \tau_2 \mid \tau_1 \rightharpoonup \tau_2 \\
\textit{Expressions} & e & ::= & v \mid \mathsf{in}\ e \mid \mathsf{out}\ e \mid \mathsf{inl}_\tau\ e \mid \mathsf{inr}_\tau\ e \mid \mathsf{case}(e_1, e_2, e_3) \mid \\
 & & & (e_1, e_2) \mid \mathsf{fst}\ e \mid \mathsf{snd}\ e \mid e_1\, e_2 \\
\textit{Values} & v & ::= & * \mid \mathsf{in}\ v \mid \mathsf{inl}_\tau\ v \mid \mathsf{inr}_\tau\ v \mid \\
 & & & (v_1, v_2) \mid \mathsf{fix}\ f(x{:}\tau).e \\
\textit{Evaluation} & E & ::= & \_ \mid \mathsf{in}\ E \mid \mathsf{out}\ E \mid \mathsf{inl}_\tau\ E \mid \mathsf{inr}_\tau\ E \mid \mathsf{case}(E, e, e') \mid \\
\textit{Contexts} & & & (E, e) \mid (v, E) \mid \mathsf{fst}\ E \mid \mathsf{snd}\ E \mid E\, e \mid v\, E
\end{array}
$$

The $\mathcal{L}$ *raw terms* are given by the syntax trees generated by the grammar above, with e as start symbol, modulo α-equivalence, as usual. Alpha-equivalence is denoted $\equiv_\alpha$. Observe that ρ is a type constant. Distinguish a fixed type expression τ_ρ, the intuition being that ρ is a recursive type isomorphic to τ_ρ; in and out are used to mediate the isomorphism.

A *finite* map is a map with finite domain. We use $\emptyset$ to denote the map whose domain is the empty set. The domain and range of a finite map f are denoted $\mathrm{Dom}(f)$ and $\mathrm{Rng}(f)$, respectively. When f and g are finite maps, $f + g$ is the finite map whose domain is $\mathrm{Dom}(f) \cup \mathrm{Dom}(g)$ and whose value is $g(x)$, if $x \in \mathrm{Dom}(g)$, and $f(x)$ otherwise. $f \downarrow A$ means the restriction of f to A, and $f \setminus\!\!\setminus A$ means f restricted to the complement of A. We use

$[x_1 : y_1, \ldots, x_n : y_n]$ to denote the finite map which maps x_i to y_i, for all $1 \le i \le n$.

We denote the set of all types by Type. A *typing context* is a finite map from variables to types; we use Γ to range over typing contexts. If $x \notin \mathrm{Dom}(\Gamma)$, then $\Gamma[x : \tau]$ denotes the typing context $\Gamma + [x : \tau]$. A typing judgment has the form $\Gamma \vdash e : \tau$. The typing rules are given in Figure 1. We write $\vdash e : \tau$ for $\emptyset \vdash e : \tau$. The $\mathcal{L}$ *terms* is the set of raw terms e for which there exists, for each e, a typing context Γ and a type τ such that $\Gamma \vdash e : \tau$.

Note that, even though there is no explicit introduction rule for the type 0, there are terms of this type, for instance $(\mathsf{fix}\ f(x{:}1).f\,x)\,*$.

The set of expressions of type τ with free variables given types by Γ, denoted $\mathrm{Exp}_\tau(\Gamma)$ is defined as follows.

$$\mathrm{Exp}_\tau(\Gamma) \stackrel{\mathrm{def}}{=} \{\, e \mid \Gamma \vdash e : \tau \,\}$$

Further define

$$\mathrm{Exp}_\tau \stackrel{\mathrm{def}}{=} \mathrm{Exp}_\tau(\emptyset)$$

Likewise, we define sets for values as follows.

$$\mathrm{Val}_\tau(\Gamma) \stackrel{\mathrm{def}}{=} \{\, v \mid \Gamma \vdash v : \tau \,\}$$

and

$$\mathrm{Val}_\tau \stackrel{\mathrm{def}}{=} \mathrm{Val}_\tau(\emptyset)$$

Substitution of an expression e' for free occurrences of x in e is written $[e'/x]e$. The parallel substution of $e_1, \ldots e_n$ for $x_1, \ldots, x_n$ in e is written $[e_1, \ldots, e_n/x_1, \ldots, x_n]e$. We let $\mathrm{FV}(e)$ denote the set of free variables in e. We use $\lambda x{:}\tau.e$ as an abbreviation for $\mathsf{fix}\ f(x{:}\tau).e$ where f is some variable satisfying $f \notin \mathrm{FV}(e)$.

2.1 Contexts

The $\mathcal{L}$ contexts, ranged over by C, are the syntax tree generated by the grammar for e augmented by the clause

$$C \quad ::= \quad \cdots \mid \mathsf{p}$$

where p ranges over some fixed set of parameters. Note that the syntax trees of $\mathcal{L}$ terms are contexts, namely the ones with no occurrence of parameters. $[C/\mathsf{p}]C'$ denotes the context obtained from context C' by replacing all occurrences of p in C' with C. This may involve capture of variables.

$$\Gamma \vdash x : \tau \quad (\Gamma(x) = \tau) \tag{T-VAR}$$

$$\Gamma \vdash * : 1 \tag{T-ONE}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2} \tag{T-PROD}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \mathsf{fst}\ e : \tau_1} \tag{T-FST}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \mathsf{snd}\ e : \tau_2} \tag{T-SND}$$

$$\frac{\Gamma \vdash e : \tau_1}{\Gamma \vdash \mathsf{inl}_{\tau_2}\ e : \tau_1 + \tau_2} \tag{T-INL}$$

$$\frac{\Gamma \vdash e : \tau_2}{\Gamma \vdash \mathsf{inr}_{\tau_1}\ e : \tau_1 + \tau_2} \tag{T-INR}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 + \tau_2 \qquad \Gamma \vdash e_2 : \tau_1 \rightharpoonup \tau \qquad \Gamma \vdash e_3 : \tau_2 \rightharpoonup \tau}{\Gamma \vdash \mathsf{case}(e_1, e_2, e_3) : \tau} \tag{T-CASE}$$

$$\frac{\Gamma[f : \tau_1 \rightharpoonup \tau_2][x : \tau_1] \vdash e : \tau_2}{\Gamma \vdash \mathsf{fix}\ f(x{:}\tau_1).e : \tau_1 \rightharpoonup \tau_2} \quad (f, x \notin \mathrm{Dom}(\Gamma)) \tag{T-FIX}$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \rightharpoonup \tau \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\, e_2 : \tau} \tag{T-APP}$$

$$\frac{\Gamma \vdash e : \rho}{\Gamma \vdash \mathsf{out}\ e : \tau_\rho} \tag{T-OUT}$$

$$\frac{\Gamma \vdash e : \tau_\rho}{\Gamma \vdash \mathsf{in}\ e : \rho} \tag{T-IN}$$

Figure 1: Typing Rules

Lemma 2.1 *If* $C_1 \equiv_\alpha C_2$ *then* $[C_1/\mathsf{p}]C' \equiv_\alpha [C_2/\mathsf{p}]C'$.

Proof By induction on C'. $\square$

By Lemma 2.1, the operation of substituting for a parameter in a context induces a well-defined operation on α-equivalence classes of $\mathcal{L}$ contexts.

Notation 2.2 *Most of the time we will only use contexts involving a single parameter which we will write as* $\_$. *We write* $C\{\_\}$ *to indicate that* C *is a context containing no parameters other than* $\_$ *(note that it may contain no parameters at all). If* e *is an* $\mathcal{L}$ *term, then* $C\{e\}$ *denotes the raw term resulting from choosing a representative syntax tree for* e*, substituting it for the parameter in* c *and forming the* α*-equivalence class of the resulting* $\mathcal{L}$ *syntax tree (which by the remarks above is independent of the choice of representative for* e*).*

2.2 Typed Contexts

We will assume given a function that assigns types to parameters. We write $\_\tau$ to indicate that a parameter $\_$ has type τ.

The relation $\Gamma \vdash C : \tau$ is inductively generated by axioms and rules just like those defining $\Gamma \vdash e : \tau$ together with the following axiom for parameters.

$$\Gamma \vdash \_\tau : \tau \tag{T-PAR}$$

The set of contexts of type τ with free variables given types by Γ, denoted $\mathrm{Ctx}_\tau(\Gamma)$ is defined as follows.

$$\mathrm{Ctx}_\tau(\Gamma) \stackrel{\text{def}}{=} \{\, C \mid \Gamma \vdash C : \tau \,\}$$

$$\mathrm{Ctx}_\tau \stackrel{\text{def}}{=} \mathrm{Ctx}_\tau(\emptyset)$$

2.3 Evaluation

The operational semantics will be given by term rewriting and will be defined for all closed terms (not only those of ground type).

The set of *evaluation contexts* are the syntax trees generated by the grammar for E. Note that this is clearly a subset of the set of contexts (with parameters including $\_$). Hence we shall use the notation associated with contexts for evaluation contexts also. In addition, we define

$$\mathrm{ECtx}_\tau(\Gamma) \stackrel{\text{def}}{=} \{\, E \mid \Gamma \vdash E : \tau \,\}$$

and

$$\mathrm{ECtx}_\tau \stackrel{\mathrm{def}}{=} \mathrm{ECtx}_\tau(\emptyset)$$

Note that evaluation contexts are not capturing. Hence we have the following lemma.

Lemma 2.3 *For all $e \in Exp_\tau$ and for all $E\{\_\tau\} \in ECtx_{\tau'}$, $E\{e\} = [e/x]E\{x\}$*

Proof By induction on E. $\square$

Redices are generated by the following grammar.

$$\begin{array}{lll} \textit{Redices} \quad r & ::= & (\mathsf{fix}\ f(x{:}\tau).e)\ v \mid \mathsf{fst}\ (v_1, v_2) \mid \mathsf{snd}\ (v_1, v_2) \mid \\ & & \mathsf{out}\ (\mathsf{in}\ v) \mid \mathsf{case}(\mathsf{inl}_\tau\ v, e_1, e_2) \mid \mathsf{case}(\mathsf{inr}_\tau\ v, e_1, e_2) \end{array}$$

Note that the set of redices is a subset of the set of expressions. We define

$$\mathrm{Rexp}_\tau(\Gamma) \stackrel{\mathrm{def}}{=} \{\, r \mid \Gamma \vdash r : \tau \,\}$$

and

$$\mathrm{Rexp}_\tau \stackrel{\mathrm{def}}{=} \mathrm{Rexp}_\tau(\emptyset)$$

Lemma 2.4 *For all $e \in Exp_\tau \setminus Val_\tau$, there exists a unique pair of evaluation context, E, and redex, r, such that $e = E\{r\}$.*

Proof By induction on e. $\square$

The reduction rules for redices are as follows.

$$\begin{array}{lllr} (\mathsf{fix}\ f(x{:}\tau).e)\ v & \leadsto & [\mathsf{fix}\ f(x{:}\tau).e, v/f, x]e & (\textsc{r-beta}) \\ \mathsf{fst}\ (v_1, v_2) & \leadsto & v_1 & (\textsc{r-fst}) \\ \mathsf{snd}\ (v_1, v_2) & \leadsto & v_2 & (\textsc{r-snd}) \\ \mathsf{out}\ (\mathsf{in}\ v) & \leadsto & v & (\textsc{r-out}) \\ \mathsf{case}(\mathsf{inl}_\tau\ v, e_1, e_2) & \leadsto & e_1\, v & (\textsc{r-case-inl}) \\ \mathsf{case}(\mathsf{inr}_\tau\ v, e_1, e_2) & \leadsto & e_2\, v & (\textsc{r-case-inr}) \end{array}$$

Further, we define, for closed expressions e and e', $e \mapsto e'$ if and only if $e = E\{r\}$ and $e' = E\{e_1\}$ and $r \leadsto e_1$.

Definition 2.5 *The reflexive and transitive closure of $\mapsto$ is denoted $\mapsto^*$. For $n \geq 0$, we define $e \mapsto^n e'$ iff $e = e_0 \mapsto e_1 \mapsto \cdots e_{n-1} \mapsto e_n = e'$. Further, we write $e \Uparrow$ iff whenever $e \mapsto^* e'$, there exists an e'' such that $e' \mapsto e''$. Finally, we write $e \Downarrow$ iff there exists a v such that $e \mapsto^* v$.*

Note that evaluation is only defined for closed expressions and that during evaluation we will only ever substitute closed values for variables.

Lemma 2.6 (Evaluation is deterministic) *If $e \mapsto e'$ and $e \mapsto e''$, then $e' = e''$.*

Proof Follows by Lemma 2.4. $\square$

Lemma 2.7 1. *For all τ and all $v \in Val_\tau$: $v \Downarrow$.*

2. *For all $e \in Exp_\tau$, if $e \mapsto e'$, then $e \in Exp_\tau \setminus Val_\tau$.*

Lemma 2.8 *For all $E\{\_{\tau_1}\} \in ECtx_{\tau_2}$, and for all $e \in Exp_{\tau_1} \setminus Val_{\tau_1}$, if $E\{e\} \mapsto E\{e'\}$, then there exists $E_1\{\_{\tau_3}\} \in ECtx_{\tau_1}$ and $r \in Rexp_{\tau_3}$ and $e_1 \in Exp_{\tau_3}$ such that $e = E_1\{r\}$ and $e' = E_1\{e_1\}$ and $r \mapsto e_1$.*

Lemma 2.9 1. *If $\Gamma[x : \tau] \vdash e : \tau'$ and $\Gamma \vdash e' : \tau$, then $\Gamma \vdash [e'/x]e : \tau'$.*

2. *If $\vdash E\{e\} : \tau$ then there exists a τ_e such that $\vdash e : \tau_e$ and $\vdash E\{e'\} : \tau$ for all e' such that $\vdash e' : \tau_e$.*

Theorem 2.10 (Preservation)

If $e \mapsto e'$ and $\vdash e : \tau$, then $\vdash e' : \tau$.

Proof By the definition of the evaluation relation and Lemma 2.9. $\square$

Lemma 2.11 (Canonical Forms) *Suppose that $\vdash v : \tau$. Then*

- $\tau \neq 0$.
- *If $\tau = 1$, then $v = *$.*
- *If $\tau = \rho$, then $v = \mathsf{in}\ v'$ for some $v' \in Val_{\tau_\rho}$.*
- *If $\tau = \tau_1 + \tau_2$, then either $v = \mathsf{inl}_{\tau_2}\ v'$ for some $v' \in Val_{\tau_1}$ or $v = \mathsf{inr}_{\tau_1}\ v'$ for some $v' \in Val_{\tau_2}$.*
- *If $\tau = \tau_1 \times \tau_2$, then $v = (v_1, v_2)$ for some $v_1 \in Val_{\tau_1}$ and some $v_2 \in Val_{\tau_2}$.*
- *If $\tau = \tau_1 \rightharpoonup \tau_2$, then $v = \mathsf{fix}\ f(x{:}\tau_1).e$ for some variables f and x, and some $e \in Exp_{\tau_2}([f : \tau_1 \rightharpoonup \tau_2, x : \tau_1])$.*

Proof By inspection of the typing rules and the definition of closed values. □

Theorem 2.12 (Progress) *If* $\vdash e : \tau$*, then either* e *is a value or there exists an* e' *such that* $e \mapsto e'$*.*

Proof By induction on $\vdash e : \tau$. □

Lemma 2.13 (Uniformity of Evaluation) *For all* $e \in Exp_{\tau_1} \setminus Val_{\tau_1}$ *and for all* $E\{\_{\tau_1}\} \in ECtx_{\tau_2}$*, if* $E\{e\} \mapsto E\{e'\}$*, then* $\forall E'\{\_{\tau_1}\} \in ECtx_{\tau_2} : E'\{e\} \mapsto E'\{e'\}$*.*

Proof By the definition of the evaluation relation $e \mapsto e'$ and the definition of the reduction rules. □

Lemma 2.14 *For all* $e, e' \in Exp_\tau \setminus Val_{\tau_1}$ *and for all* $E\{\_\tau\} \in ECtx_{\tau'}$*, if* $E\{e\} \mapsto E\{e'\}$*, then also* $e \mapsto e'$*.*

Lemma 2.15 *If* $e \in Exp_\tau$ *and* $e \Uparrow$*, then* $\forall E\{\_\tau\} \in ECtx_{\tau'} : E\{e\} \Uparrow$*.*

3 Experimental Equivalence

For closed expressions of base type 1, we define a notion of Kleene approximation and Kleene equivalence as follows.

Definition 3.1 (Kleene Approximation and Equivalence) *For all* $e, e' \in Exp_1$*, we define* $e \preceq^k e'$ *iff* $e \mapsto^* * \Rightarrow e' \mapsto^* *$ *and* $e \approx^k e'$ *iff* $e \mapsto^* * \iff e' \mapsto^* *$*.*

For closed expressions we define notions of experimental approximation and experimental equivalence as follows.

Definition 3.2 (Experimental Approximation and Equivalence) *For all* $e, e' \in Exp_\tau$*, we define*

$$\begin{array}{lcl} \vdash e \preceq e' : \tau & \iff & \forall E\{\_\tau\} \in ECtx_1 : E\{e\} \preceq^k E\{e'\} \\ \vdash e \approx e' : \tau & \iff & \forall E\{\_\tau\} \in ECtx_1 : E\{e\} \approx^k E\{e'\} \end{array}$$

Lemma 3.3 $\vdash e \approx e' : \tau \iff (\vdash e \preceq e' : \tau \quad \wedge \quad \vdash e' \preceq e : \tau)$

Notation 3.4 *When τ is clear from context we write $e \preceq e'$ for $\vdash e \preceq e' : \tau$ and $e \approx e'$ for $\vdash e \approx e' : \tau$.*

We now establish some basic properties of experimental equivalence and evaluation.

Lemma 3.5 *If $\vdash e_1 \approx e_2 : \tau$ then $e_1 \Downarrow$ iff $e_2 \Downarrow$.*

Lemma 3.6 *For all $e \in Exp_{\tau_1}$ and for all $E\{\_{\tau_1}\} \in ECtx_{\tau}$, $\vdash E\{e\} \approx (\lambda x{:}\tau.E\{x\})\, e : \tau$.*

Lemma 3.7 ($\mapsto \subseteq \approx$) *For all $e, e' \in Exp_{\tau}$, if $e \mapsto e'$, then $\vdash e \approx e' : \tau$.*

Lemma 3.8 *Experimental equivalence, $\approx$, is an equivalence relation. That is, the following three properties hold.*

1. *If $\vdash e_1 \approx e_2 : \tau$ and $\vdash e_2 \approx e_3 : \tau$, then $\vdash e_1 \approx e_3 : \tau$.*
2. *If $e \in Exp_{\tau}$, then $\vdash e \approx e : \tau$.*
3. *If $\vdash e_1 \approx e_2 : \tau$, then $\vdash e_2 \approx e_1 : \tau$.*

Lemma 3.9

1. *If $\vdash e \approx (e_1, e_2) : \tau_1 \times \tau_2$ then $e \Downarrow$ iff $e_1 \Downarrow$ and $e_2 \Downarrow$.*
2. *If $\vdash e \approx (e_1, e_2) : \tau_1 \times \tau_2$ and $\vdash e_1 \approx e_1' : \tau_1$ and $\vdash e_2 \approx e_2' : \tau_2$, then $\vdash e \approx (e_1', e_2') : \tau_1 \times \tau_2$.*
3. *If $\vdash e \approx (e_1, e_2) : \tau_1 \times \tau_2$ and $e \Downarrow$, then $\vdash \mathsf{fst}\, e \approx e_1 : \tau_1$ and $\vdash \mathsf{snd}\, e \approx e_2 : \tau_2$.*

Lemma 3.10

1. *If $\vdash e \approx \mathsf{inl}_{\tau_2}\, e' : \tau_1 + \tau_2$ then $e \Downarrow$ iff $e' \Downarrow$. If $\vdash e \approx \mathsf{inr}_{\tau_1}\, e' : \tau_1 + \tau_2$, then $e \Downarrow$ iff $e' \Downarrow$.*
2. *If $\vdash e \approx \mathsf{inl}_{\tau_2}\, e' : \tau_1 + \tau_2$ and $\vdash e' \approx e'' : \tau_1$, then $\vdash e \approx \mathsf{inl}_{\tau_2}\, e'' : \tau_1 + \tau_2$. If $\vdash e \approx \mathsf{inr}_{\tau_1}\, e' : \tau_1 + \tau_2$ and $\vdash e' \approx e'' : \tau_2$, then $\vdash e \approx \mathsf{inr}_{\tau_1}\, e'' : \tau_1 + \tau_2$.*
3. *If $\vdash e \approx \mathsf{inl}_{\tau_2}\, e' : \tau_1 + \tau_2$ and $e \Downarrow$, then there exists a v' such that $\vdash e \approx \mathsf{inl}_{\tau_2}\, v' : \tau_1 + \tau_2$ and $\vdash e' \approx v' : \tau_1$. If $\vdash e \approx \mathsf{inr}_{\tau_1}\, e' : \tau_1 + \tau_2$ and $e \Downarrow$, then there exists a v' such that $\vdash e \approx \mathsf{inr}_{\tau_1}\, v' : \tau_1 + \tau_2$ and $\vdash e' \approx v' : \tau_2$.*

4. $\vdash \mathsf{inl}_{\tau_2}\, e \approx \mathsf{inl}_{\tau_2}\, e' : \tau_1 + \tau_2$ *iff* $\vdash e \approx e' : \tau_1$. $\vdash \mathsf{inr}_{\tau_1}\, e \approx \mathsf{inr}_{\tau_1}\, e' : \tau_1 + \tau_2$ *iff* $\vdash e \approx e' : \tau_2$.

Lemma 3.11

1. *If* $\vdash e \approx \mathsf{in}\, e' : \rho$, *then* $e \Downarrow$ *iff* $e' \Downarrow$.
2. *If* $\vdash e \approx \mathsf{in}\, e' : \rho$ *and* $e' \Uparrow$ *then* $e \Uparrow$.
3. *If* $\vdash e \approx \mathsf{in}\, e' : \rho$ *and* $\vdash e' \approx e'' : \tau_\rho$, *then* $\vdash e \approx \mathsf{in}\, e'' : \rho$.
4. $\vdash \mathsf{in}\, e \approx \mathsf{in}\, e' : \rho$ *iff* $\vdash e \approx e' : \tau_\rho$.

We shall now prove a somewhat technical lemma to the effect (Corollary 3.13) that we can restrict the set of evaluation contexts to consider when proving $\vdash e \preceq e' : \tau$. It turns out that we can restrict attention to the evaluation contexts for which the hole occurs in an atomic testing context.

The *atomic testing contexts*, ranged over by T are the syntax tree generated by the following grammar

$$T \quad ::= \quad \_{1} \mid \mathsf{out}\ \_{\rho} \mid \mathsf{case}(\_{\tau_1+\tau_2}, e_1, e_2) \mid \mathsf{fst}\ \_{\tau_1\times\tau_2} \mid \mathsf{snd}\ \_{\tau_1\times\tau_2} \mid \_{\tau_1\rightharpoonup\tau_2}\, v$$

We define

$$\mathrm{TCtx}_\tau(\Gamma) \stackrel{\mathrm{def}}{=} \{\, T \mid \Gamma \vdash T : \tau \,\}$$

and

$$\mathrm{TCtx}_\tau \stackrel{\mathrm{def}}{=} \mathrm{TCtx}_\tau(\emptyset)$$

We shall be a little more pedantic than elsewhere in the following lemma as it is a little delicate.

Lemma 3.12 $\forall n \in N : \forall \tau \in \mathrm{Type} : \forall v, v' \in \mathit{Val}_\tau$: *if*

$$\forall \tau' \in \mathrm{Type} : \forall E\{\_{\tau'}\} \in \mathit{ECtx}_1 : \forall T\{\_{\tau}\} \in \mathit{TCtx}_{\tau'} : E\{T\{v\}\} \preceq^k E\{T\{v'\}\} \tag{1}$$

then

$$\begin{array}{l}\forall \tau' \in \mathrm{Type} : \forall z \in \mathrm{Var} : \forall e \in \mathit{Exp}_{\tau'}([z:\tau]) : \forall E\{\_{\tau'}\} \in \mathit{ECtx}_1([z:\tau]) : \\ \quad ([v/z](E\{e\}) \mapsto^n *) \Rightarrow ([v'/z](E\{e\}) \mapsto^* *)\end{array} \tag{2}$$

Proof By induction on n.

Basis ($n = 0$)*:* Let $\tau \in \mathrm{Type}$ and $v, v' \in \mathrm{Val}_\tau$ be arbitrary. Assume (1). We are to show (2) with 0 substituted for n. Let $\tau' \in \mathrm{Type}$, $z \in \mathrm{Var}$, $e \in \mathrm{Exp}_{\tau'}([z:\tau])$, and $E\{\_{\tau'}\} \in \mathrm{ECtx}_1([z:\tau])$ be arbitrary. Assume $[v/z](E\{e\}) \mapsto^0 *$. Then $[v/z](E\{e\}) = *$. Thus there are two cases to consider.

1. $\tau' = 1$, $E = \_{1}$, and $e = *$

2. $\tau' = 1$, $E = \_{1}$, $e = z$, $\tau = 1$, and $v = *$

SubCase 1: Then also $[v'/z](E\{e\}) = *$ and hence $[v'/z](E\{e\}) \mapsto^* *$, as required.

SubCase 2: By assumption (1) with $\tau' = \tau$, $E\{\_{\tau'}\} = \_{\tau}$, and $T\{\_{\tau}\} = \_{1}$, we have that $(v \mapsto^* *) \Rightarrow (v' \mapsto^* *)$. As $v = *$, indeed $(v \mapsto^* *)$, whence we can conclude that $v' \mapsto^* *$. Thus $v' = *$ and hence also $[v'/z](E\{e\}) = *$, so $[v'/z](E\{e\}) \mapsto^* *$, as required.

Inductive Step: We assume that the lemma holds for $n \geq 0$ and show for $n + 1$. Let $\tau \in \text{Type}$ and $v, v' \in \text{Val}_\tau$ be arbitrary. Assume (1). We are to show (2) with $(n + 1)$ substituted for n. Let $\tau' \in \text{Type}$, $z \in \text{Var}$, $e \in \text{Exp}_{\tau'}([z : \tau])$, and $E\{\_{\tau'}\} \in \text{ECtx}_1([z : \tau])$ be arbitrary. Assume $[v/z](E\{e\}) \mapsto^{n+1} *$. Since there is at least one reduction step, we can proceed by cases on the first reduction step.

Case R-BETA: Then there are two cases

1. $\tau = \tau_1 \rightharpoonup \tau_2$ and $E\{e\} = E'\{z\, v_1\}$ for some $E'\{\_{\tau_2}\} \in \text{ECtx}_1([z : \tau])$ and some $v_1 \in \text{Val}_{\tau_1}([z : \tau])$

2. $E\{e\} = E'\{\mathsf{fix}\, f(x{:}\tau_1).e_0\, v_1\}$ for some $E'\{\_{\tau_2}\} \in \text{ECtx}_1([z : \tau])$, some $v_1 \in \text{Val}_{\tau_1}([z : \tau])$, and some $\mathsf{fix}\, f(x{:}\tau_1).e_0 \in \text{Val}_{\tau_1 \rightharpoonup \tau_2}([z : \tau])$

SubCase 1: Then v is of the form $\mathsf{fix}\, f(x{:}\tau_1).e_0$. Thus

$$\begin{array}{rcll} [v/z](E\{e\}) & = & [v/z](E'\{z\, v_1\}) & \\ & = & [v/z](E'\{(\mathsf{fix}\, f(x{:}\tau_1).e_0)\, v_1\}) & \\ & \mapsto & [v/z](E'\{[v, v_1/f, x]e_0\}) & (3) \\ & \mapsto^n & * & (4) \end{array}$$

Now by the induction hypothesis we have (with $\tau = \tau$, $v = v$ and $v' = v'$ and noting that (1) holds by assumption) that

$$\begin{array}{l} \forall \tau' \in \text{Type} : \forall z \in \text{Var} : \forall e \in \text{Exp}_{\tau'}([z : \tau]) : \forall E\{\_{\tau'}\} \in \text{ECtx}_1([z : \tau]) : \\ \quad ([v/z](E\{e\}) \mapsto^n *) \Rightarrow ([v'/z](E\{e\}) \mapsto^* *) \end{array} \tag{5}$$

Letting $\tau' = \tau_2$, $z = z$, $e = [v, v_1/f, x]e_0$, and $E\{\_{\tau'}\} = E'\{\_{\tau_2}\}$ in (5) and using (3) and (4), we conclude that

$$[v'/z]E'\{[v, v_1/f, x]e_0\} \mapsto^* * \tag{6}$$

By (6) and recalling that $v = \mathsf{fix}\ f(x{:}\tau_1).e_0$ we get that

$$[v'/z]E'\{v\,v_1\} \mapsto [v'/z]E'\{[v, v_1/f, x]e_0\} \mapsto^* * \tag{7}$$

By assumption (1) on (7) with $\tau' = \tau_2$, $E\{\_{\tau'}\} = [v/z](E'\{\_{\tau_2}\}) \in \mathrm{ECtx}_1$, and $T\{\_\tau\} = \_\tau\,([v/z]v_1) \in \mathrm{TCtx}_{\tau_2}$ we get

$$[v'/z]E'\{v'\,v_1\} \mapsto^* * \tag{8}$$

Hence, as $[v'/z]E\{e\} = [v'/z]E'\{v'\,v_1\}$, we have the required by (8).

SubCase 2: Then

$$\begin{aligned}
[v/z](E\{e\}) &= [v/z](E'\{\mathsf{fix}\ f(x{:}\tau_1).e_0\,v_1\}) \\
&\mapsto [v/z](E'\{[\mathsf{fix}\ f(x{:}\tau_1).e_0, v_1/f, x]e_0\}) && (9) \\
&\mapsto^n * && (10)
\end{aligned}$$

Now by the induction hypothesis we have (with $\tau = \tau$, $v = v$ and $v' = v'$ and noting that (1) holds by assumption) that

$$\begin{aligned}
&\forall \tau' \in \mathrm{Type} : \forall z \in \mathrm{Var} : \forall e \in \mathrm{Exp}_{\tau'}([z:\tau]) : \forall E\{_{\tau'}\} \in \mathrm{ECtx}_1([z:\tau]) : \\
&\quad ([v/z](E\{e\}) \mapsto^n *) \Rightarrow ([v'/z](E\{e\}) \mapsto^* *)
\end{aligned} \tag{11}$$

Letting $\tau' = \tau_2$, $z = z$, $e = [\mathsf{fix}\ f(x{:}\tau_1).e_0, v_1/f, x]e_0$, and $E\{\_{\tau'}\} = E'\{\_{\tau_2}\}$ in (5) and using (9) and (10), we conclude that

$$[v'/z](E'\{[\mathsf{fix}\ f(x{:}\tau_1).e_0, v_1/f, x]e_0\}) \mapsto^* * \tag{12}$$

We have that

$$\begin{aligned}
[v'/z](E\{e\}) &= [v'/z](E'\{(\mathsf{fix}\ f(x{:}\tau_1).e_0)\,v_1\}) && (13) \\
&\mapsto [v'/z](E'\{[\mathsf{fix}\ f(x{:}\tau_1).e_0, v_1/f, x]e_0\}) && (14)
\end{aligned}$$

Hence, combining (12) and (13) and (14), we have that

$$[v'/z](E\{e\}) \mapsto^* *$$

as required.

Case R-OUT, R-CASE-INL, R-CASE-INR, R-FST, or R-SND: The proof for each of these cases proceeds analogously to the previous case, with two subcases for each case, and using the corresponding atomic testing contexts. □

Corollary 3.13 $\forall \tau \in \text{Type} : \forall v, v' \in \mathit{Val}_\tau :$ *if*

$$\forall \tau' \in \text{Type} : \forall E\{\_\tau'\} \in \mathit{ECtx}_1 : \forall T\{\_\tau\} \in \mathit{TCtx}_{\tau'} : E\{T\{v\}\} \preceq^k E\{T\{v'\}\} \tag{15}$$

then

$$\vdash v \preceq v' : \tau$$

Proof Let $\tau \in$ Type, and $v, v' \in \mathrm{Val}_\tau$ be arbitrary. Assume (15). Let $E\{\_\tau\} \in \mathrm{ECtx}_1$ be arbitrary. We are to show that

$$E\{v\} \mapsto^* * \Rightarrow E\{v'\} \mapsto^* * \tag{16}$$

From our assumption (15) we get by Lemma 3.12

$$\begin{aligned}&\forall \tau' \in \text{Type} : \forall z \in \text{Var} : \forall e \in \text{Exp}_{\tau'}([z : \tau]) : \forall E\{\_\tau'\} \in \text{ECtx}_1([z : \tau]) : \\ &\quad ([v/z](E\{e\}) \mapsto^* *) \Rightarrow ([v'/z](E\{e\}) \mapsto^* *)\end{aligned} \tag{17}$$

Letting $\tau' = \tau$, $e = z$ and $E\{\_\tau'\} = E\{\_\tau\}$ in (17), we get (16), as required. $\square$

The next corollary is an immediate consequence of Corollary 3.13 and is the formulation which we will often make use of in the following.

Corollary 3.14

1. *To show* $\vdash v \preceq v' : \tau_1 \rightharpoonup \tau_2$, *it suffices to show*

$$\forall E\{\_{\tau_1 \rightharpoonup \tau_2}\, v_1\} \in \mathit{ECtx}_1 : E\{v\} \preceq^k E\{v'\}$$

2. *To show* $\vdash v \preceq v' : \tau_1 \times \tau_2$, *it suffices to show*

$$\forall E\{\mathsf{fst}\ \_{\tau_1 \times \tau_2}\} \in \mathit{ECtx}_1 : E\{v\} \preceq^k E\{v'\}$$

and

$$\forall E\{\mathsf{snd}\ \_{\tau_1 \times \tau_2}\} \in \mathit{ECtx}_1 : E\{v\} \preceq^k E\{v'\}$$

3. *To show* $\vdash v \preceq v' : \tau_1 + \tau_2$, *it suffices to show*

$$\forall E\{\mathsf{case}(\_{\tau_1 + \tau_2}, e_1, e_2)\} \in \mathit{ECtx}_1 : E\{v\} \preceq^k E\{v'\}$$

4. *To show* $\vdash v \preceq v' : \rho$, *it suffices to show*

$$\forall E\{\mathsf{out}\ \_\rho\} \in \mathit{ECtx}_1 : E\{v\} \preceq^k E\{v'\}$$

Proof Follows by Corollary 3.13 and the definition of atomic contexts. $\square$

3.1 Compactness of Evaluation

In this section we show that a fix-term is approximated, in the experimental approximation pre-order, by its finite unrollings. Further, we show that to fill a context is a monotone operation with respect to the experimental pre-order and we use this to show that a fix-term is the least upper bound of its finite unrollings. These properties are also referred to as compactness of evaluation. Finally, we show that to fill a context is a continous operation with respect to the approximation pre-order. We shall only be concerned with closed fix-terms, as this suffices for our purposes.

Our development of compactness of evaluation follows the approach of Pitts [16, Section 5] quite closely but there are some technical differences due to the fact that we use a reduction semantics rather than a natural semantics as employed by Pitts. We have chosen this formulation, using cofinal sets, because it fits nicely with our formulation of admissible relations, for which a formulation based on cofinal sets suffices (see Section 4).

Throughout this section we shall consider a particular fixed term $F = \mathsf{fix}\ f(x{:}\tau_1).e$ satisfying $F \in \mathrm{Exp}_{\tau_1 \rightharpoonup \tau_2}$, and use the following abbreviations:

$$\begin{aligned} F_0 &\stackrel{\text{def}}{=} \mathsf{fix}\ f^0(x{:}\tau_1).e \stackrel{\text{def}}{=} \mathsf{fix}\ f(x{:}\tau_1).f\ x \\ F_{n+1} &\stackrel{\text{def}}{=} \mathsf{fix}\ f^{n+1}(x{:}\tau_1).e \stackrel{\text{def}}{=} \lambda x{:}\tau_1.[F_n/f]e \\ F_\omega &\stackrel{\text{def}}{=} F \end{aligned}$$

Note that we here simply define some abbreviations of expressions already in the language. This is opposed to introducing new labelled expressions and new notions of reduction for labelled expressions as, e.g., done by Gunter [9].

We will only consider contexts involving parameters of type $\tau_1 \rightharpoonup \tau_2$. We write $C\{\vec{\mathsf{p}}\}$ for such a context whose parameters are included in the list $\vec{\mathsf{p}}$ (note that we do not required that all the parameters in $\vec{\mathsf{p}}$ occur in C). Given an k-tuple $\vec{n} = (n_1, \ldots, n_k)$ of natural numbers, then we make the following abbreviations.

$$\begin{aligned} C\{F_{\vec{n}}\} &\stackrel{\text{def}}{=} C\{F_{n_1}, \ldots, F_{n_k}\} \\ C\{F_{\vec{\omega}}\} &\stackrel{\text{def}}{=} C\{F_\omega, \ldots, F_\omega\} \end{aligned}$$

The length of a list of parameter $\vec{\mathsf{p}}$ will be denoted $|\vec{\mathsf{p}}|$.

Definition 3.15 *For each k, we partially order the set N^k by*

$$\vec{n} \leq \vec{n}' \iff (n_1 \leq n'_1 \wedge \cdots \wedge n_k \leq n'_k)$$

Definition 3.16 *A subset $I \subseteq N^k$ is said to be cofinal in N^k if and only if, for all $\vec{n} \in N^k$, $\exists \vec{n'} \in I : \vec{n} \leq \vec{n'}$. We write $\mathcal{P}_{\mathrm{cof}}(N^k)$ for the set of all such cofinal subsets of N^k.*

We say that a context C is a value if it follows the grammar for values v augmented by the obvious clause for parameters. We introduce the following definitions of sets of value contexts

$$\mathrm{VCtx}_\tau(\Gamma) \stackrel{\mathrm{def}}{=} \{ C \in \mathrm{Ctx}_\tau(\Gamma) \mid C \text{ is a value or } C \text{ is a parameter} \}$$

$$\mathrm{VCtx}_\tau \stackrel{\mathrm{def}}{=} \mathrm{VCtx}_\tau(\emptyset)$$

We use V to range over value contexts. We say that a value context is *proper* if it is not a parameter.

Remark 3.17 *Note that, if $V\{\_{\tau}\} \in VCtx_{\tau'}$ is a proper value context and $e \in Exp_\tau$, then $V\{e\}$ is a value. Also, if $V\{\_{\tau}\} \in VCtx_{\tau'}$ and $v \in Val_\tau$, then $V\{v\}$ is a value.*

Notation 3.18 *We abbreviate $V\{F_{\vec{m}}\}$ and $V\{F_{\vec{\omega}}\}$ analogously to $C\{F_{\vec{m}}\}$ and $C\{F_{\vec{\omega}}\}$.*

Definition 3.19 *If $C\{\vec{\mathsf{p}}\}$ is a context and $V\{\vec{\mathsf{p}}'\}$ is a value context, then we write $C\{\vec{\mathsf{p}}\} \Downarrow^F V\{\vec{\mathsf{p}}'\}$ to mean that for all $I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|})$*

$$\{ \vec{m}\vec{m}' \mid \vec{m} \in I \wedge C\{F_{\vec{m}}\} \mapsto^* V\{F_{\vec{m}'}\} \} \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|})$$

Note that the relation $C\{\vec{\mathsf{p}}\} \Downarrow^F V\{\vec{\mathsf{p}}'\}$ is preserved under renaming of the parameters $\vec{\mathsf{p}}$ and, independently, the parameters $\vec{\mathsf{p}}'$.

Lemma 3.20 *If $C\{\vec{\mathsf{p}}\}$ is a context and $V\{\vec{\mathsf{p}}'\}$ is a value context, then*

$$C\{\vec{\mathsf{p}}\} \Downarrow^F V\{\vec{\mathsf{p}}'\} \iff C\{\vec{\mathsf{p}}\vec{\mathsf{q}}\} \Downarrow^F V\{\vec{\mathsf{p}}'\vec{\mathsf{q}}'\}$$

Proof By definition of $\Downarrow^F$ and simple properties of cofinal subsets of N^k. $\square$

Lemma 3.21

1. *If $V\{\vec{\mathsf{p}}\}$ is a proper value context, then $V\{\vec{\mathsf{p}}\} \Downarrow^F V\{\vec{\mathsf{p}}\}$.*
2. *If $E'\{V\}\{\vec{\mathsf{p}}\} \Downarrow^F V''\{\vec{\mathsf{p}}''\}$ and $V'\{\vec{\mathsf{p}}\vec{\mathsf{p}}'\}$ is a value context, then $E'\{\mathsf{fst}\ (V, V')\}\{\vec{\mathsf{p}}\vec{\mathsf{p}}'\} \Downarrow^F V''\{\vec{\mathsf{p}}''\}$.*

3. *If* $E'\{V\}\{\vec{\mathsf{p}}\} \Downarrow^F V''\{\vec{\mathsf{p}}''\}$ *and* $V'\{\vec{\mathsf{p}}\vec{\mathsf{p}}'\}$ *is a value context, then* $E'\{\mathsf{snd}\ (V, V')\}\{\vec{\mathsf{p}}\vec{\mathsf{p}}'\} \Downarrow^F V''\{\vec{\mathsf{p}}''\}$.

4. *If* $E'\{V\}\{\vec{\mathsf{p}}\} \Downarrow^F V'\{\vec{\mathsf{p}}'\}$, *then* $E'\{\mathsf{out}\ (\mathsf{in}\ V)\}\{\vec{\mathsf{p}}\} \Downarrow^F V'\{\vec{\mathsf{p}}'\}$.

5. *If* $E'\{e_1\, v\}\{\vec{\mathsf{p}}\} \Downarrow^F V\{\vec{\mathsf{p}}'\}$ *and* $e_2 = C_2\{F_{\vec{\omega}}\}$ *for some* $C_2\{\vec{\mathsf{p}}\vec{\mathsf{p}}'\}$, *then* $E'\{\mathsf{case}(\mathsf{inl}_{\tau_2}\ v, e_1, e_2)\}\{\vec{\mathsf{p}}\vec{\mathsf{p}}'\} \Downarrow^F V\{\vec{\mathsf{p}}'\}$

6. *If* $E'\{e_2\, v\}\{\vec{\mathsf{p}}\} \Downarrow^F V\{\vec{\mathsf{p}}'\}$, *and* $e_1 = C_1\{F_{\vec{\omega}}\}$ *for some* $C_1\{\vec{\mathsf{p}}\vec{\mathsf{p}}'\}$, *then* $E'\{\mathsf{case}(\mathsf{inr}_{\tau_1}\ v, e_1, e_2)\}\{\vec{\mathsf{p}}\vec{\mathsf{p}}'\} \Downarrow^F V\{\vec{\mathsf{p}}'\}$

Proof Item 1 is immediate. We show item 2; items 3–6 are similar.

Let $C = E'\{V\}$ and let $C' = E'\{\mathsf{fst}\ (V, V')\}$. By the assumption and Lemma 3.20,

$$C\{\vec{\mathsf{p}}\vec{\mathsf{p}}'\} \Downarrow^F V''\{\vec{\mathsf{p}}''\} \tag{18}$$

Assume $I \in \mathcal{P}_{\mathrm{cof}}(N^{|\mathsf{p}|+|\mathsf{p}'|})$. Then we are to show that

$$I' \stackrel{\mathrm{def}}{=} \{\, \vec{m}\vec{m}' \mid \vec{m} \in I \wedge C'\{F_{\vec{m}}\} \mapsto^* V''\{F_{\vec{m}'}\}\,\}$$

is a cofinal subset of $N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|+|\vec{\mathsf{p}}''|}$. But $C'\{F_{\vec{m}}\} \mapsto C\{F_{\vec{m}}\}$ so by determinism of evaluation, $C'\{F_{\vec{m}}\} \mapsto^* V''\{F_{\vec{m}'}\}$ if and only if $C\{F_{\vec{m}}\} \mapsto^* V''\{F_{\vec{m}'}\}$. Hence I' equals the set

$$\{\, \vec{m}\vec{m}' \mid \vec{m} \in I \wedge C\{F_{\vec{m}}\} \mapsto^* V''\{F_{\vec{m}'}\}\,\}$$

which by (18) is a cofinal subset of $N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|+|\vec{\mathsf{p}}''|}$, as required. □

Lemma 3.22 (Compactness of Evaluation) *For all* $C\{\vec{\mathsf{p}}\} \in Ctx_\tau$, *if* $C\{F_{\vec{\omega}}\} \mapsto^* v$, *then there exists a* $V\{\vec{\mathsf{p}}'\} \in VCtx_\tau$ *such that* $v = V\{F_{\vec{\omega}}\}$ *and* $C\{\vec{\mathsf{p}}\} \Downarrow^F V\{\vec{\mathsf{p}}'\}$.

Proof By induction on the length, n, of $C\{F_{\vec{\omega}}\} \mapsto^* v$.

Basis $(n = 0)$*:* Pick $V = C$. If C is a parameter, then the required is immediate (recall that F_ω is a value). Otherwise, C is a proper value context and the required follows by Lemma 3.21, item 1.

Inductive Step: We assume it holds for n and show for $n+1$. To this end assume $C\{F_{\vec{\omega}}\} \mapsto^{n+1} v$. We proceed by cases on the first reduction step.

Case R-FST: Then $C\{F_{\vec{\omega}}\} = E\{\mathsf{fst}\ (v_1, v_2)\}$ with $E = E'\{F_{\vec{\omega}}\}$, $v_1 = V_1\{F_{\vec{\omega}}\}$, and $v_2 = V_2\{F_{\vec{\omega}}\}$ for some $E'\{\vec{\mathsf{p}_1}\}$, $V_1\{\vec{\mathsf{p}_1}\}$, and $V_2\{\vec{\mathsf{p}_1}\vec{\mathsf{p}_2}\}$ with

$\vec{\mathsf{p}} = \vec{\mathsf{p}_1}\vec{\mathsf{p}_2}$. Moreover, $E\{\mathsf{fst}\ (v_1, v_2)\} \mapsto E\{v_1\} \mapsto^n v$. Note that $E\{v_1\}$ is of the form $C_1'\{F_{\vec{\omega}}\}$ where $C_1'\{\vec{\mathsf{p}_1}\} = E'\{V_1\}\{\vec{\mathsf{p}_1}\}$. Hence we can apply induction on n to yield that there exists a $V\{\vec{\mathsf{p}}'\}$ such that $v = V\{F_{\vec{\omega}}\}$ and $C_1'\{\vec{\mathsf{p}_1}\} \Downarrow^F V\{\vec{\mathsf{p}}'\}$. By Lemma 3.21, item 2, also $C\{\vec{\mathsf{p}}\} \Downarrow^F V\{\vec{\mathsf{p}}'\}$, as required.

Case R-SND, R-OUT, R-CASE-INL, R-CASE-INR: All analogous to preceding case, using corresponding item in Lemma 3.21.

Case R-BETA: Then $C\{F_{\vec{\omega}}\} = E\{(\mathsf{fix}\ f'(x'{:}\tau').e')\,v'\}$ for some f', x', τ', e', v', and E. There are two cases, depending on whether $F = \mathsf{fix}\ f'(x'{:}\tau').e'$ or not.

SubCase I: Assume $F = \mathsf{fix}\ f'(x'{:}\tau').e'$. Then

$$\begin{array}{rcl} C\{F_{\vec{\omega}}\} & = & E\{(\mathsf{fix}\ f(x{:}\tau).e)\,v'\} \\ & \mapsto & E\{[\mathsf{fix}\ f(x{:}\tau).e, v'/f, x]e\} \\ & \mapsto^n & v \end{array}$$

where $E = E'\{F_{\vec{\omega}}\}$ and $v' = V'\{F_{\vec{\omega}}\}$ for some $E'\{\vec{\mathsf{p}}\}$ and $V'\{\vec{\mathsf{p}}\}$. We have that

$$E\{[\mathsf{fix}\ f(x{:}\tau).e, v'/f, x]e\} = E'\{[p, V'/f, x]e\}\{F_{\vec{\omega}}\}$$

Let $C'\{\vec{\mathsf{p}}p\} = E'\{[p, V'/f, x]e\}$. Then we have that $C'\{F_{\vec{\omega}}\} \mapsto^n v$ so by induction on n there exists a $V\{\vec{\mathsf{p}}'\}$ such that $v = V\{\vec{\mathsf{p}}'\}$ and

$$C'\{\vec{\mathsf{p}}p\} \Downarrow^F V\{\vec{\mathsf{p}}'\} \tag{19}$$

We aim to show that

$$C\{\vec{\mathsf{p}}\} \Downarrow^F V\{\vec{\mathsf{p}}'\} \tag{20}$$

Let $I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|})$ be arbitrary. We are to show that

$$I_1 \stackrel{\mathrm{def}}{=} \{\, \vec{m}\vec{m}' \mid \vec{m} \in I \wedge C\{F_{\vec{m}}\} \mapsto^* V\{F_{\vec{m}\vec{m}'}\}\,\}$$

is a cofinal subset of $N^{|\vec{\mathsf{p}}|}$. Define

$$I_2 \stackrel{\mathrm{def}}{=} \{\, n\vec{m} \mid \vec{m} \in I \wedge n = n_k \wedge C\{F_{\vec{m}}\} \mapsto C'\{F_{n\vec{m}}\}\,\}$$

Clearly, I_2 is cofinal since I is cofinal. By (19) we therefore have that

$$I_3 \stackrel{\mathrm{def}}{=} \{\, \vec{m}\vec{m}' \mid \vec{m} \in I_2 \wedge C'\{F_{\vec{m}}\} \mapsto^* V\{F_{\vec{m}'}\}\,\}$$

is cofinal. Now it is easy to see that $I_3 \subseteq I_1$ and thus, since I_3 is cofinal, I_1 is cofinal. Since I was arbitrary, we have (20) as desired.

SubCase II: Assume $F \neq \mathsf{fix}\ f'(x'{:}\tau').e'$. Then

$$\begin{array}{rcl} C\{F_{\vec{\omega}}\} & = & E\{(\mathsf{fix}\ f'(x'{:}\tau').e')\, v'\} \\ & \mapsto & E\{[\mathsf{fix}\ f'(x'{:}\tau').e', v'/f, x]e'\} \\ & \mapsto^n & v \end{array}$$

and $E\{[\mathsf{fix}\ f'(x'{:}\tau').e', v'/f, x]e'\}$ is of the form $C_1\{F_{\vec{\omega}}\}$ for some $C_1\{\vec{\mathsf{p}}\vec{\mathsf{p}_1}\}$. By induction we get that

$$C_1\{\vec{\mathsf{p}}\vec{\mathsf{p}_1}\} \Downarrow^F V\{\vec{\mathsf{p}}'\} \tag{21}$$

Let $I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|})$ be arbitrary. Let

$$I_1 \stackrel{\mathrm{def}}{=} \{\, \vec{m}\vec{m}' \mid \vec{m} \in I \wedge \vec{m}' \in N^{\vec{\mathsf{p}_1}} \,\}$$

Then I_1 is a cofinal subset of $N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}_1}|}$ since I is cofinal. Hence, by (21),

$$I_2 \stackrel{\mathrm{def}}{=} \{\, \vec{m}\vec{m}'\vec{m}'' \mid \vec{m}\vec{m}' \in I_1 \wedge C_1\{F_{\vec{m}\vec{m}'}\} \mapsto^* V\{F_{\vec{m}''}\}\,\}$$

is cofinal and thus it is easy to see that also

$$I_3 \stackrel{\mathrm{def}}{=} \{\, \vec{m}\vec{m}'' \mid \vec{m} \in I \wedge C\{F_{\vec{m}}\} \mapsto^* V\{F_{\vec{m}''}\}\,\}$$

is cofinal, as required. □

The following lemma expresses that the finite unrollings of a fix-term form a chain with respect to the approximation order and that the fix-term itself is an upper bound of this sequence. We shall soon see that it is in fact the *least* upper bound.

Lemma 3.23 *For all $i \in N$, $\vdash F_i \preceq F_{i+1} : \tau_1 \rightharpoonup \tau_2$ and $\vdash F_i \preceq F_\omega : \tau_1 \rightharpoonup \tau_2$.*

Proof Both properties are shown by induction on i. □

To show that a fix-term is the *least* upper bound of its finite unrollings we shall need that the operation of filling a context is a monotone operation with respect to the experimental pre-order (in other words, the experimental pre-order is a pre-congruence). To this end we shall first generalize the experimental pre-order to open expressions in the following way.

Definition 3.24 *An* expression substitution γ for a type environment Γ *is a finite map from variables to closed expressions satisfying the following two conditions.*

1. $\mathrm{Dom}(\gamma) = \mathrm{Dom}(\Gamma)$.
2. $\forall x \in \mathrm{Dom}(\gamma) : \emptyset \vdash \gamma(x) : \Gamma(x)$.

Definition 3.25 *A* value substitution γ for Γ *is an expression substitution for* Γ *satisfying* $\forall x \in \mathrm{Dom}(\Gamma) : \gamma(x) \Downarrow$.

Definition 3.26 *Let* γ *and* γ' *be expression substitutions for* Γ*. Then* γ *approximates* γ'*, written* $\vdash \gamma \preceq \gamma' : \Gamma$*, if and only if* $\forall x \in \mathrm{Dom}(\Gamma) : \vdash \gamma(x) \preceq \gamma'(x) : \Gamma(x)$*. Likewise, we write* $\vdash \gamma \approx \gamma' : \Gamma$*, if and only if* $\forall x \in \mathrm{Dom}(\Gamma) : \vdash \gamma(x) \approx \gamma'(x) : \Gamma(x)$.

Note that this definition also expresses when a *value* substitution γ approximates another value substituition γ' (both for some Γ) as a value substition is just a special expression substitution (we need a notion of expression substition in Section 3.2, which is why we have chosen this formulation).

Definition 3.27 (Open Experimental Approximation and Equivalence) *For all* e *and* e'*, if* $\Gamma \vdash e : \tau$ *and* $\Gamma \vdash e' : \tau$*, then we define* $\Gamma \vdash e \preceq e' : \tau$ *if, and only if, for all value substitutions* γ *and* γ' *for* Γ *satisfying* $\vdash \gamma \preceq \gamma' : \Gamma$, $\vdash \gamma(e) \preceq \gamma'(e') : \tau$*. Moreover, we define* $\Gamma \vdash e \approx e' : \tau$ *if and only if* $\Gamma \vdash e \preceq e' : \tau$ *and* $\Gamma \vdash e' \preceq e : \tau$.

An alternative definition of open experimental approximation would be to say that $\Gamma \vdash e \preceq e' : \tau$ if and only if, for all expression substitutions γ for Γ, $\vdash \gamma(e) \preceq \gamma(e') : \tau$. However, we need the more general definition (specifically it is used in proving (22) below).

Lemma 3.28 *If* $[f : \tau_1 \rightharpoonup \tau_2, x : \tau_1] \vdash e \preceq e' : \tau_2$ *then* $\vdash \mathsf{fix}\, f(x{:}\tau_1).e \preceq \mathsf{fix}\, f(x{:}\tau_1).e' : \tau_1 \rightharpoonup \tau_2$.

Proof By induction on i, it is easy to show that, for all $i \in N$,

$$\vdash \mathsf{fix}\, f^i(x{:}\tau_1).e \preceq \mathsf{fix}\, f^i(x{:}\tau_1).e' : \tau_1 \rightharpoonup \tau_2 \tag{22}$$

By Corollary 3.14, it suffices to show,

$$\forall E\{_{-\tau_1 \rightharpoonup \tau_2} v_1\} \in \mathrm{ECtx}_1 : E\{\mathsf{fix}\, f(x{:}\tau_1).e\} \preceq^k E\{\mathsf{fix}\, f(x{:}\tau_1).e'\}$$

So assume, $E\{(\mathsf{fix}\, f(x{:}\tau_1).e)\,(v_1)\} \mapsto^* *$. Let $C\{\_{\tau_1 \rightharpoonup \tau_2}\} = E\{\_{\tau_1 \rightharpoonup \tau_2}\, v_1\}$. Then, by Lemma 3.22, there exists a $V\{\vec{\mathsf{p}}'\}$ such that $V\{F_{\vec{\omega}}\} = *$ and

$$C\{\vec{\mathsf{p}}\} \Downarrow^F V\{\vec{\mathsf{p}}'\} \tag{23}$$

Let $I = N$, clearly a cofinal set. Then by (23),

$$I' \stackrel{\text{def}}{=} \{\, i \in I \mid C\{\mathsf{fix}\, f^i(x{:}\tau_1).e\} \mapsto^* * \,\}$$

is a cofinal subset of $N|\vec{\mathsf{p}}|$. Hence I' is in particular non-empty, i.e., there exists $i \in I'$ such that $C\{\mathsf{fix}\, f^i(x{:}\tau_1).e\} \mapsto^* *$. Thus, by definition of I and C, there exists an $i \in N$ such that $E\{(\mathsf{fix}\, f^i(x{:}\tau_1).e)\,(v_1)\} \mapsto^* *$. Hence, by (22), we also have $E\{(\mathsf{fix}\, f^i(x{:}\tau_1).e')\,(v_1)\} \mapsto^* *$. Then by Lemma 3.23, we get $E\{(\mathsf{fix}\, f(x{:}\tau_1).e')\,(v_1)\} \mapsto^* *$, as required. □

Lemma 3.29 *If $\Gamma, \Gamma' \vdash e \preceq e' : \tau$, $C\{\_{\tau}\} \in Ctx_{\tau'}(\Gamma)$, $\Gamma \vdash C\{e\} : \tau'$, and $\Gamma \vdash C\{e'\} : \tau'$, then $\Gamma \vdash C\{e\} \preceq C\{e'\} : \tau'$.*

Proof By induction on C. In the case for $C = \mathsf{fix}\, f(x{:}\tau).C'$, use Lemma 3.28; all the other cases follow easily (either directly by the assumptions or by induction and using Lemmas 3.7–3.11 and composition of evaluation contexts). □

The following corollary expresses the monotonicity of contexts with respect to the experimental pre-order — in other words, the experimental pre-order is a pre-congruence. We shall subsequently show that contexts are not only monotone, but also continuous (in an appropriate sense).

Corollary 3.30 (Context Monotonicity) *If $\vdash e \preceq e' : \tau_1$ and $C\{\_{\tau_1}\} \in Ctx_{\tau}$, then $\vdash C\{e\} \preceq C\{e'\} : \tau$.*

Proof Follows immediately by Lemma 3.29. □

Lemma 3.31 *If $\vdash e_1 \preceq e_1' : \tau_1, \ldots, \vdash e_k \preceq e_k' : \tau_k$ and $C\{\_1, \ldots, \_k\} \in Ctx_{\tau}$ with $\_i$ of type τ_i, for all $1 \leq i \leq k$, then $\vdash C\{e_1, \ldots, e_k\} \preceq C\{e_1', \ldots, e_k'\} : \tau$.*

Proof By repeated application of Corollary 3.30 and transitivity of $\preceq$. □

Corollary 3.32 (Experimental equivalence is a congruence relation) *If $\vdash e_1 \approx e_1' : \tau_1 \ldots, \vdash e_k \approx e_k' : \tau_k$ and $C\{\_1, \ldots, \_k\} \in Ctx_{\tau}$ with $\_i$ of type τ_i, for all $1 \leq i \leq k$, then $\vdash C\{e_1, \ldots, e_k\} \approx C\{e_1', \ldots, e_k'\} : \tau$.*

Proof Follows immediately by Lemma 3.31. $\square$

Before embarking on the theorem from which it follows that a fix-term is the least upper bound of its finite unrollings (Theorem 3.36), the proof of which will make use of context monotonicity, we shall first make another use of context monotonicity. We shall show that experimental approximation can be used to give an alternative characterization of the usual definition of contextual equivalence — via this alternative characterization, the proof principles for establishing experimental equivalence that are developed in this paper can be used also to establish results of contextual equivalence. This theorem (Theorem 3.34) is reminiscent of the CIU Theorem of Mason, Smith and Talcott [11].

Definition 3.33 (Contextual Approximation and Equivalence) *If* $\Gamma \vdash e : \tau$ *and* $\Gamma \vdash e' : \tau$, *we define*

$$\begin{array}{lcl} \vdash e \preceq^c e' : \tau & \iff & \forall C\{\_\tau\} \in Ctx_1 : C\{e\} \preceq^k C\{e'\} \\ \vdash e \approx^c e' : \tau & \iff & \forall C\{\_\tau\} \in Ctx_1 : C\{e\} \approx^k C\{e'\} \end{array}$$

Theorem 3.34 *If* $\Gamma \vdash e : \tau$ *and* $\Gamma \vdash e' : \tau$, *then* $\vdash e \preceq^c e' : \tau$ *iff* $\Gamma \vdash e \preceq e' : \tau$.

Proof

$\Rightarrow$ Assume $\vdash e \preceq^c e' : \tau$ Further assume $\vdash \gamma \preceq \gamma' : \Gamma$ and that $E\{\gamma(e)\} \mapsto^* *$. We are to show that $E\{\gamma'(e')\} \mapsto^* *$. Assume that $\mathrm{Dom}(\Gamma) = \{x_1, \ldots, x_n\}$ and let $\tau_i = \Gamma(x_i)$. Without loss of generality (by the definition of value substitution and by considering the following part of the proof) we may assume that $\gamma(x_i) \in \mathrm{Val}_{\tau_i}$ and $\gamma'(x_i) \in \mathrm{Val}_{\tau_i}$. Let $v_i = \gamma(x_i)$ and $v_i' = \gamma'(x_i)$, for all $1 \leq i \leq n$. Let

$$C\{\_\tau\} = \lambda x_1{:}\tau_1.\cdots\lambda x_n{:}\tau_n.E\{\_\tau\}\, v_1 \cdots v_n$$

Then $C\{e\} \mapsto^* *$. Hence by the assumption that $\vdash e \preceq^c e' : \tau$, also $C\{e'\} \mapsto^* *$. Let

$$C'\{\_\tau_1, \ldots, \_\tau_n\} = \lambda x_1{:}\tau_1.\cdots\lambda x_n{:}\tau_n.E\{e'\}\, \_\tau_1 \cdots \_\tau_n$$

Then we have that $C'\{v_1, \ldots, v_n\} \mapsto^* *$. By Lemma 3.31,

$$\vdash C'\{v_1, \ldots, v_n\} \preceq C'\{v_1', \ldots, v_n'\} : 1$$

Hence it follows that $C'\{v_1', \ldots, v_n'\} \mapsto^* *$, and thus $E\{\gamma'(e')\} \mapsto^* *$, as required.

$\Leftarrow$ Assume $\Gamma \vdash e \preceq e' : \tau$. Then, by Lemma 3.29, $\vdash C\{e\} \preceq C\{e'\} : 1$. From this it follows (with $E = \_{}_1$) that $C\{e\} \preceq^k C\{e'\}$, as required. $\square$

Corollary 3.35 *If* $\Gamma \vdash e : \tau$ *and* $\Gamma \vdash e' : \tau$*, then* $\vdash e \approx^c e' : \tau$ *iff* $\Gamma \vdash e \approx e' : \tau$.

Proof Immediate from Theorem 3.34. $\square$

Theorem 3.36 *For all* $C\{\vec{\mathsf{p}}\} \in Ctx_\tau$*, the following three propositions are equivalent.*

1. $\vdash C\{F_{\vec{\omega}}\} \preceq e : \tau$
2. $\exists I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|}) : \forall \vec{m} \in I : \quad \vdash C\{F_{\vec{m}}\} \preceq e : \tau$
3. $\forall I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|}) : \forall \vec{m} \in I : \quad \vdash C\{F_{\vec{m}}\} \preceq e : \tau$

Proof We show that 1 is equivalent to 2 and that 1 is equivalent to 3 from which the theorem follows.

$1 \Rightarrow 2$: Let $I = \{\vec{m} \in N^{|\vec{\mathsf{p}}|} \mid m_1 = m_2 = \cdots = m_{|\vec{\mathsf{p}}|}\}$. I is clearly cofinal. By Lemmas 3.31 and 3.23 we get $\forall \vec{m} \in I : \quad \vdash C\{F_{\vec{m}}\} \preceq C\{F_{\vec{\omega}}\} : \tau$ so by transitivity the desired follows.

$2 \Rightarrow 1$: Let $E\{\_{}_\tau\} \in \mathrm{ECtx}_1$ be arbitrary. We are to show that

$$E\{C\{F_{\vec{\omega}}\}\} \preceq^k E\{e\}$$

So assume $E\{C\{F_{\vec{\omega}}\}\} \mapsto^* *$. Then by Lemma 3.22, there exists a $V\{\vec{\mathsf{p}}'\}$ such that $V\{F_{\vec{\omega}}\} = *$ and

$$E\{C\}\{\vec{\mathsf{p}}\} \Downarrow^F V\{\vec{\mathsf{p}}'\} \tag{24}$$

Clearly, $V = *$. By the assumption that $I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|})$ and (24) we get that

$$I' \stackrel{\mathrm{def}}{=} \{\vec{m} \in I \mid E\{C\}\{F_{\vec{m}}\} \mapsto^* *\}$$

is a cofinal subset of $N^{|\vec{\mathsf{p}}|}$. Hence I' is in particular non-empty, i.e., there exists $\vec{m} \in I'$ such that $E\{C\}\{F_{\vec{m}}\} \mapsto^* *$. Now, $E\{C\}\{F_{\vec{m}}\} = E\{C\{F_{\vec{m}}\}\}$ so we have $\exists \vec{m} \in I' : E\{C\{F_{\vec{m}}\}\} \mapsto^* *$. Finally, since I' is a subset of I we get by the assumption 2 that $E\{e\} \mapsto^* *$, as required.

$1 \Rightarrow 3$: Let $I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|})$ be arbitrary. The required follows by Lemmas 3.31 and 3.23 and transitivity. (as in $1 \Rightarrow 2$ above).

$3 \Rightarrow 1$: Easy, since clearly $3 \Rightarrow 2$ and we have already shown $2 \Rightarrow 1$ above. $\square$

Corollary 3.37 (Context Continuity) *For all $C\{\vec{\mathsf{p}}\} \in Ctx_\tau$,*

$$\vdash C\{F_{\vec{\omega}}\} \preceq e : \tau \iff \forall n \in N : \quad \vdash C\{F_n, \ldots, F_n\} \preceq e : \tau$$

Proof Follows by Theorem 3.36. $\square$

Let $C\{\vec{\mathsf{p}}\} = -_{\tau_1 \rightharpoonup \tau_2}$ in Corollary 3.37. Then the corollary together with Lemma 3.23 intuitively says that F_ω is the least upper bound of the chain of its finite unrollings: By Lemma 3.23,

$$F_0 \preceq F_1 \preceq F_2 \preceq \cdots$$

is a chain with upper bound F_ω. By Corollary 3.37, if e is an upper bound of the same chain, then $F_\omega \preceq e$, so F_ω is a least upper bound of the chain:

$$F_\omega = \bigsqcup \{ F_0, F_1, F_2, \ldots \}$$

Furthermore, by Corollary 3.30,

$$C\{F_0\} \preceq C\{F_1\} \preceq C\{F_2\} \preceq \cdots$$

is again a chain with upper bound $C\{F_\omega\}$, and by Corollary 3.37, if e is an upper bound of the same chain, then $C\{F_\omega\} \preceq e$, so $C\{F_\omega\}$ is a least upper bound of the chain:

$$C\{F_\omega\} = \bigsqcup \{ C\{F_0\}, C\{F_1\}, C\{F_2\}, \ldots \}$$

In other words, to fill a context is a continuous operation for chains of finite unrollings of fix terms with respect to the approximation order.

As explained by Mason, Smith, and Talcott [11] arbitrary chains of terms do not always have a least upper bound. This leads Mason, Smith, and Talcott to develop a notion of ordering between sets of terms, for which arbitrary chains *do* have a least upper bound, [11, Lemma 4.31]. Here, however, we shall only ever consider chains of the form

$$C\{F_0\} \preceq C\{F_1\} \preceq C\{F_2\} \preceq \cdots$$

for some given closed fix-term F and thus the chains, which we shall consider, will always have a least upper bound. Hence we do not need to develop more complicated notions of approximation à la the set ordering developed by Mason, Smith, and Talcott [11].

3.2 Syntactic Projections

In this section we introduce syntactic projection terms which are the syntactic counterpart of the semantic projection functions known from domain theory. These syntactic projections will be used in the construction of the desired relations in Section 5.

Let π be a variable. For all types τ, we define terms $\Pi_\tau : \tau \rightharpoonup \tau$ (given $\pi : \rho \rightharpoonup \rho$) by induction on τ as follows.

$$\begin{array}{rcl}
\Pi_\rho & \stackrel{\text{def}}{=} & \lambda x{:}\rho.\pi\, x \\
\Pi_0 & \stackrel{\text{def}}{=} & \lambda x{:}0.x \\
\Pi_1 & \stackrel{\text{def}}{=} & \lambda x{:}1.x \\
\Pi_{\tau_1 \times \tau_2} & \stackrel{\text{def}}{=} & \lambda x{:}\tau_1 \times \tau_2.(\Pi_{\tau_1}\ (\mathsf{fst}\ x), \Pi_{\tau_2}\ (\mathsf{snd}\ x)) \\
\Pi_{\tau_1 + \tau_2} & \stackrel{\text{def}}{=} & \lambda x{:}\tau_1 + \tau_2.\mathsf{case}(x, \lambda x{:}\tau_1.\mathsf{inl}_{\tau_2}\ (\Pi_{\tau_1}\ x), \lambda x{:}\tau_2.\mathsf{inr}_{\tau_1}\ (\Pi_{\tau_2}\ x)) \\
\Pi_{\tau_1 \rightharpoonup \tau_2} & \stackrel{\text{def}}{=} & \lambda f{:}\tau_1 \rightharpoonup \tau_2.\lambda x{:}\tau_1.\Pi_{\tau_2}\ (f\ (\Pi_{\tau_1}\ x))
\end{array}$$

Note that π is possibly free in these so defined terms. Further, define terms $\pi^i : \rho \rightharpoonup \rho$, for all $i \geq 0$, by induction on i as follows.

$$\begin{array}{rcl}
\pi^0 & \stackrel{\text{def}}{=} & \mathsf{fix}\ \pi(x{:}\rho).\pi\, x \\
\pi^{i+1} & \stackrel{\text{def}}{=} & \lambda x{:}\rho.[\pi^i/\pi]\mathsf{in}\ (\Pi_{\tau_\rho}\ (\mathsf{out}\ x))
\end{array}$$

and define

$$\pi^\infty \stackrel{\text{def}}{=} \mathsf{fix}\ \pi(x{:}\rho).\mathsf{in}\ (\Pi_{\tau_\rho}\ (\mathsf{out}\ x)) : \rho \rightharpoonup \rho$$

Observe that π^i and also π^∞ are values.

Note that the π^i's are the finite unrollings of the fix-term π^∞ so, as explained in the previous subsection, π^∞ is the least upper bound of the chain of π^i's. The π^∞ term corresponds to the least fixed point $\mathit{fix}(\delta)$ of the continuous function $\delta(e) = iF(e,e)i^{-1}$ in [17, Definition 3.2]. We shall show that π^∞ is experimentally equivalent to the identity function (more precisely, the term $\lambda x{:}\rho.x$); this corresponds to the minimal invariant property in [17, Definition 3.2].

Example Assume $\tau_\rho = 1 + \rho$. Intuitively, our recursive type then corresponds to the type of natural numbers. Then π^∞ is equal to

$$\mathsf{fix}\ \pi(x{:}\rho).\mathsf{in}\ ((\lambda x{:}1+\rho.\mathsf{case}(x, \lambda x{:}1.\mathsf{inl}_\rho\ ((\lambda x{:}1.x)\ x), \lambda x{:}\rho.\mathsf{inr}_1\ ((\lambda x{:}\rho.\pi\, x)\ x)))\ (\mathsf{out}\ x))$$

Intuitively, it is clear that this is equivalent to the identify function. □

For all τ and all $i \geq 0$, we define

$$\Pi_\tau^i \stackrel{\text{def}}{=} [\pi^i/\pi]\Pi_\tau : \tau \rightharpoonup \tau$$

Finally, for all τ, we define

$$\Pi_\tau^\infty \stackrel{\text{def}}{=} [\pi^\infty/\pi]\Pi_\tau : \tau \rightharpoonup \tau$$

The following Lemmas 3.38–3.41 express that the above definitions do indeed define terms.

Lemma 3.38 *For all τ, $[\pi : \rho \rightharpoonup \rho] \vdash \Pi_\tau : \tau \rightharpoonup \tau$.*

Proof By induction on τ. □

Lemma 3.39 $\vdash \pi^\infty : \rho \rightharpoonup \rho$.

Proof By Lemma 3.38 and strengthening lemma for typing. □

Lemma 3.40 *For all τ, $\vdash \Pi_\tau^\infty : \tau \rightharpoonup \tau$.*

Proof By Lemmas 3.38 and 3.39 and substitution for typing. □

Lemma 3.41 *For all τ, for all $i \geq 0$, $\vdash \Pi_\tau^i : \tau \rightharpoonup \tau$; and for all $i \geq 0$, $\vdash \pi^i : \rho \rightharpoonup \rho$.*

Proof Simultaneously by induction on (i, τ) ordered lexicographically. □

We aim to show that π^∞ is operationally equivalent to the identity function $\lambda x{:}\rho.x$. To this end we need a series of simple lemmas which we now proceed to establish .

Lemma 3.42 *If $\vdash e \approx * : 1$ then*

1. *For all $i \geq 0$, $\vdash \Pi_1^i\, e \approx * : 1$.*
2. $\vdash \Pi_1^\infty\, e \approx * : 1$

Lemma 3.43 *If $\vdash e \approx (v_1, v_2) : \tau_1 \times \tau_2$, then*

1. *For all $i \geq 0$, $\vdash \Pi_{\tau_1 \times \tau_2}^i\, e \approx (\Pi_{\tau_1}^i\, v_1, \Pi_{\tau_2}^i\, v_2) : \tau_1 \times \tau_2$.*

2. $\vdash \Pi^{\infty}_{\tau_1 \times \tau_2}\, e \approx (\Pi^{\infty}_{\tau_1}\, v_1, \Pi^{\infty}_{\tau_2}\, v_2) : \tau_1 \times \tau_2$.

Proof We show 1; 2 is similar. Let $i \geq 0$ be arbitrary. Assume $\vdash e \approx (v_1, v_2) : \tau_1 \times \tau_2$. Then by Lemma 3.5, $e \Downarrow$, i.e., there exists a v such that $e \mapsto^* v$. By Canonical Forms Lemma (Lemma 2.11), $v = (v_1', v_2')$ for some v_1' and v_2'. Hence

$$\Pi^{i}_{\tau_1 \times \tau_2}\, e = (\lambda x{:}\tau_1 \times \tau_2.(\Pi^{i}_{\tau_1}\, (\mathsf{fst}\ x), \Pi^{i}_{\tau_2}\, (\mathsf{snd}\ x)))\, (e) \mapsto^* (\Pi^{i}_{\tau_1}\, v_1', \Pi^{i}_{\tau_2}\, v_2')$$

Thus by Lemma 3.7 $\vdash \Pi^{i}_{\tau_1 \times \tau_2}\, e \approx (\Pi^{i}_{\tau_1}\, v_1', \Pi^{i}_{\tau_2}\, v_2') : \tau_1 \times \tau_2$ and $\vdash e \approx (v_1', v_2') : \tau_1 \times \tau_2$. By transitivity of $\approx$, we get have $\vdash (v_1, v_2) \approx (v_1', v_2') : \tau_1 \times \tau_2$. By Lemma 3.9 it then follows that $\vdash v_1 \approx v_1' : \tau_1$ and $\vdash v_2 \approx v_2' : \tau_2$. Hence it follows, by composition of evaluation contexts, that $\vdash \Pi^{i}_{\tau_1}\, v_1 \approx \Pi^{i}_{\tau_1}\, v_1' : \tau_1$ and $\vdash \Pi^{i}_{\tau_2}\, v_2 \approx \Pi^{i}_{\tau_2}\, v_2' : \tau_2$. Hence, by Lemma 3.9, $\vdash \Pi^{i}_{\tau_1 \times \tau_2}\, (e) \approx (\Pi^{i}_{\tau_1}\, v_1, \Pi^{i}_{\tau_2}\, v_2) : \tau_1 \times \tau_2$, as required. $\square$

Lemma 3.44

1. *If* $\vdash e \approx \mathsf{inl}_{\tau_2}\ v : \tau_1 + \tau_2$, *then*
   (a) *For all* $i \geq 0$, $\vdash \Pi^{i}_{\tau_1 + \tau_2}\, e \approx \mathsf{inl}_{\tau_2}\ (\Pi^{i}_{\tau_1}\, v) : \tau_1 + \tau_2$.
   (b) $\vdash \Pi^{\infty}_{\tau_1 + \tau_2}\, e \approx \mathsf{inl}_{\tau_2}\ (\Pi^{\infty}_{\tau_1}\, v) : \tau_1 + \tau_2$.
2. *If* $\vdash e \approx \mathsf{inr}_{\tau_1}\ v : \tau_1 + \tau_2$, *then*
   (a) *For all* $i \geq 0$, $\vdash \Pi^{i}_{\tau_1 + \tau_2}\, e \approx \mathsf{inr}_{\tau_1}\ (\Pi^{i}_{\tau_2}\, v) : \tau_1 + \tau_2$.
   (b) $\vdash \Pi^{\infty}_{\tau_1 + \tau_2}\, e \approx \mathsf{inr}_{\tau_1}\ (\Pi^{\infty}_{\tau_2}\, v) : \tau_1 + \tau_2$.

Lemma 3.45 *If* $\vdash e \approx v : \tau_1 \rightharpoonup \tau_2$, *then*

1. *For all* $i \geq 0$, $\vdash \Pi^{i}_{\tau_1 \rightharpoonup \tau_2}\, e \approx \lambda x{:}\tau_1.\Pi^{i}_{\tau_2}\, (v\, (\Pi^{i}_{\tau_1}\, x)) : \tau_1 \rightharpoonup \tau_2$.
2. $\vdash \Pi^{\infty}_{\tau_1 \rightharpoonup \tau_2}\, e \approx \lambda x{:}\tau_1.\Pi^{\infty}_{\tau_2}\, (v\, (\Pi^{\infty}_{\tau_1}\, x)) : \tau_1 \rightharpoonup \tau_2$.

Proof We show 1, 2 is similar. Let $i \geq 0$ be arbitrary. Assume $\vdash e \approx v : \tau_1 \rightharpoonup \tau_2$. Then by Lemmas 3.5, 3.7, and 3.8, there exists a v' such that $e \mapsto^* v'$ and $\vdash v \approx v' : \tau_1 \rightharpoonup \tau_2$. Hence,

$$\Pi^{i}_{\tau_1 \rightharpoonup \tau_2}\, (e) \mapsto^* \Pi^{i}_{\tau_1 \rightharpoonup \tau_2}\, (v') \mapsto^* \lambda x{:}\tau_1.\Pi^{i}_{\tau_2}\, (v'\, (\Pi^{i}_{\tau_1}\, (x)))$$

so by Lemma 3.7

$$\vdash \Pi^{i}_{\tau_1 \rightharpoonup \tau_2}\, (e) \approx \lambda x{:}\tau_1.\Pi^{i}_{\tau_2}\, (v'\, (\Pi^{i}_{\tau_1}\, (x))) : \tau_1 \rightharpoonup \tau_2$$

We now claim that

$$\vdash \lambda x{:}\tau_1.\Pi^i_{\tau_2}\,(v\,(\Pi^i_{\tau_1}\,(x))) \approx \lambda x{:}\tau_1.\Pi^i_{\tau_2}\,(v'\,(\Pi^i_{\tau_1}\,(x))) : \tau_1 \rightharpoonup \tau_2 \tag{25}$$

from which the required follows by transitivity. By Corollary 3.14, to show the claim (25), it suffices to show that

$$\vdash \lambda x{:}\tau_1.\Pi^i_{\tau_2}\,(v\,(\Pi^i_{\tau_1}\,(x)))\,(v_1) \approx \lambda x{:}\tau_1.\Pi^i_{\tau_2}\,(v'\,(\Pi^i_{\tau_1}\,(x)))\,(v_1) : \tau_2 \tag{26}$$

where $v_1 \in \mathrm{Val}_{\tau_1}$ is arbitrary. Clearly, the left hand side in (26) is operationally equivalent to $\Pi^i_{\tau_2}\,(v\,(\Pi^i_{\tau_1}\,(v_1)))$ and the right hand side is operationally equivalent to $\Pi^i_{\tau_2}\,(v'\,(\Pi^i_{\tau_1}\,(v_1)))$, but these two expressions are operationally equivalent because v and v' are operationally equivalent and by composition of evaluation contexts (with the context $\Pi^i_{\tau_2}\,(\_{\tau_1 \rightharpoonup \tau_2}\,(\Pi^i_{\tau_1}\,(v_1)))$). Hence, by transitivity, the desired (26) follows. □

Lemma 3.46 *If* $\vdash e \approx \mathsf{in}\; v : \rho$, *then* $\Pi^0_\rho\, e \Uparrow$.

Lemma 3.47 *If* $\vdash e \approx \mathsf{in}\; v : \rho$, *then*

1. *For all* $i \geq 1$, $\vdash \Pi^i_\rho\, e \approx \mathsf{in}\; (\Pi^{i-1}_{\tau_\rho}\, v) : \rho$.
2. $\vdash \Pi^\infty_\rho\, e \approx \mathsf{in}\; (\Pi^\infty_{\tau_\rho}\, v) : \rho$.

Lemma 3.48 *If* $\vdash e \approx \mathsf{in}\; v : \rho$, *then*

1. *For all* $i \geq 1$, $\vdash \pi^i\, e \approx \mathsf{in}\; (\Pi^{i-1}_{\tau_\rho}\, v) : \rho$.
2. $\vdash \pi^\infty\, e \approx \mathsf{in}\; (\Pi^\infty_{\tau_\rho}\, v) : \rho$.

Lemma 3.49 *For all* τ *and for all* $i \geq 0$, $\vdash \Pi^i_\tau \preceq \lambda x{:}\tau.x : \tau \rightharpoonup \tau$.

Proof By Lemma 3.6 and Corollary 3.14 it suffices to show, for all τ, for all $v \in \mathrm{Val}_\tau$, for all $E\{\_{\tau \rightharpoonup \tau}\, v\}$

$$E\{\Pi^i_\tau\, v\} \preceq^k E\{v\} \tag{27}$$

We show this by induction on (i, τ) ordered lexicographically. We proceed by cases on τ.

Case $\tau = 1$: Follows by Lemma 3.42.
Case $\tau = 0$: Vacously true since $\mathrm{Val}_0 = \emptyset$.
Case $\tau = \rho$: We consider two cases, $i = 0$ and $i > 0$.
SubCase $i = 0$: Follows trivially by Lemma 3.46.

SubCase $i > 0$: By Canonical Forms Lemma(Lemma 2.11), $v = \mathsf{in}\, v'$ for some $v' \in \mathrm{Val}_{\tau_\rho}$. Assume $E\{\Pi^i_\tau\, \mathsf{in}\, v'\} \mapsto^* *$. Then by Lemma 3.47 (with $e = \mathsf{in}\, v'$ and using reflexivity of $\approx$ and noting that $i > 0$ by assumption) we also have that $E\{\mathsf{in}\, (\Pi^{i-1}_{\tau_\rho}\, v')\} \mapsto^* *$. Note that $i - 1 \geq 0$ as $i > 0$ by assumption and that $(i-1, \tau_\rho) \leq (i, \tau)$ in the lexicographical order, so we can apply induction to get $E\{\mathsf{in}\, v'\} \mapsto^* *$, which is the required.

Case $\tau = \tau_1 \times \tau_2$: Follows by Lemma 3.43 and induction on (i, τ_1) and (i, τ_2).

Case $\tau = \tau_1 + \tau_2$: Follows by Lemma 3.44 and induction on (i, τ_1) or (i, τ_2) depending on whether $v = \mathsf{inl}_{\tau_2}\, v'$ or $v = \mathsf{inr}_{\tau_1}\, v'$.

Case $\tau = \tau_1 \rightharpoonup \tau_2$: Follows by Lemma 3.45 and Corollary 3.14, induction on (i, τ_1) and induction on (i, τ_2). □

We are now in a position to show one half of the operational equivalence of π^∞ and the identity function, namely that π^∞ approximates the identity function.

Lemma 3.50 $\vdash \pi^\infty \preceq \lambda x{:}\rho.x : \rho \rightharpoonup \rho$

Proof By Corollary 3.37, it suffices to show

$$\forall i \in N : \quad \vdash \pi^i \preceq \lambda x{:}\rho.x : \rho \rightharpoonup \rho \tag{28}$$

We show this by induction on i.

Basis $(i = 0)$*:* By Lemma 3.6, Corollary 3.14 and Canonical Forms Lemma (Lemma 2.11), it suffices to show, for all $E\{\_{\rho}\, (\mathsf{in}\, v)\} \in \mathrm{ECtx}_1$ and all $v \in \mathrm{Val}_{\tau_\rho}$,

$$E\{\pi^0\, (\mathsf{in}\, v)\} \preceq^k E\{\mathsf{in}\, v\}$$

Recalling that $\pi^0 = \mathsf{fix}\, \pi(x{:}\rho).\pi\, x$ the required follows immediately.

Inductive Step: We assume (28) holds for i and show for $i + 1$. By Lemma 3.6, Corollary 3.14 and Canonical Forms Lemma (Lemma 2.11), it suffices to show, for all $E\{\_{\rho}\, (\mathsf{in}\, v)\} \in \mathrm{ECtx}_1$ and all $v \in \mathrm{Val}_{\tau_\rho}$,

$$E\{\pi^{i+1}\, (\mathsf{in}\, v)\} \preceq^k E\{\mathsf{in}\, v\}$$

To this end, assume

$$E\{\pi^{i+1}\, (\mathsf{in}\, v)\} \mapsto^* * \tag{29}$$

Then by Lemma 3.47 (with $e = \mathsf{in}\, v$ and using reflexivity of $\approx$ and noting that $i + 1 \geq 1$ as $i \geq 0$ by the assumption that $i \in N$) we also have that

$$E\{\mathsf{in}\, (\Pi^i_{\tau_\rho}\, v)\} \mapsto^* * \tag{30}$$

Then by Lemma 3.49, also $E\{\mathsf{in}\ v\} \mapsto^* *$, as required. $\square$

Next we aim to show the other half of the operational equivalence of π^∞ and the identity function, that is, that the identity function operationally approximates π^∞. We shall employ an idea of Mason, Smith, and Talcott [11].

We now proceed to show idempotency of Π^∞_τ and π^∞. The strategy is to show lemmas for Π^i_τ and π^i and then use compactness of evaluation to get the desired results.

Lemma 3.51 *For all $i \geq 0$ and for all τ, $\vdash \Pi^i_\tau \preceq \lambda x{:}\tau.\Pi^\infty_\tau\,(\Pi^\infty_\tau\, x) : \tau \rightharpoonup \tau$.*

Proof By Corollary 3.14 it suffices to show, for all $i \geq 0$, for all $v \in \mathrm{Val}_\tau$, and for all $E\{\_{\tau \rightharpoonup \tau}\, v\} \in \mathrm{ECtx}_1$,

$$E\{\Pi^i_\tau\, v\} \preceq^k E\{(\lambda x{:}\tau.\Pi^\infty_\tau\,(\Pi^\infty_\tau\, x))\, v\}$$

This can shown by induction on (i, τ) ordered lexicographically. $\square$

Lemma 3.52 *For all $i \geq 0$, $\vdash \pi^i \preceq \lambda x{:}\rho.\pi^\infty\,(\pi^\infty\, x) : \rho \rightharpoonup \rho$.*

Proof Follows by Lemma 3.51. $\square$

Lemma 3.53 *For all $i \geq 0$ and for all τ, $\vdash \lambda x{:}\tau.\Pi^i_\tau\,(\Pi^i_\tau\, x) \preceq \Pi^\infty_\tau : \tau \rightharpoonup \tau$.*

Proof By Corollary 3.14 it suffices to show, for all $i \geq 0$, for all $v \in \mathrm{Val}_\tau$, and for all $E\{\_{\tau \rightharpoonup \tau}\, v\} \in \mathrm{ECtx}_1$,

$$E\{(\lambda x{:}\tau.\Pi^i_\tau\,(\Pi^i_\tau\, x))\, v\} \preceq^k E\{\Pi^\infty_\tau\, v\}$$

This can shown by induction on (i, τ) ordered lexicographically. $\square$

Lemma 3.54 *For all $i \geq 0$, $\vdash \lambda x{:}\rho.\pi^i\,(\pi^i\, x) \preceq \pi^\infty : \rho \rightharpoonup \rho$.*

Proof Follows by Lemma 3.53. $\square$

Lemma 3.55 *For all τ, $\vdash \Pi^\infty_\tau \preceq \lambda x{:}\tau.\Pi^\infty_\tau\,(\Pi^\infty_\tau\, x) : \tau \rightharpoonup \tau$.*

Proof By Corollary 3.37, with $C = \_{\tau \rightharpoonup \tau}$, and Lemma 3.51. $\square$

Lemma 3.56 $\vdash \pi^\infty \preceq \lambda x{:}\rho.\pi^\infty\,(\pi^\infty\,x) : \rho \rightharpoonup \rho.$

Proof By Corollary 3.37, with $C = {}_{-\rho \rightharpoonup \rho}$, and Lemma 3.52. $\square$

Lemma 3.57 *For all* τ, $\vdash \lambda x{:}\tau.\Pi^\infty_\tau\,(\Pi^\infty_\tau\,x) \preceq \Pi^\infty_\tau : \tau \rightharpoonup \tau.$

Proof By Corollary 3.37, with $C = \lambda x{:}\rho.{}_{-1}\,({}_{-2}\,x)$ with ${}_{-1}$ and ${}_{-2}$ of type $\tau \rightharpoonup \tau$, and Lemma 3.53. $\square$

Lemma 3.58 $\vdash \lambda x{:}\rho.\pi^\infty\,(\pi^\infty\,x) \preceq \pi^\infty : \rho \rightharpoonup \rho.$

Proof By Corollary 3.37, with $C = \lambda x{:}\rho.{}_{-1}\,({}_{-2}\,x)$ with ${}_{-1}$ and ${}_{-2}$ of type $\rho \rightharpoonup \rho$, and Lemma 3.54. $\square$

Corollary 3.59 *For all* $e \in Exp_\tau$ *and for all* $E\{{}_{-\tau}\} \in ECtx_{\tau'}$, $\vdash E\{\Pi^\infty_\tau\,(\Pi^\infty_\tau\,e)\} \approx E\{\Pi^\infty_\tau\,e\} : \tau'$.

Proof Follows by Lemmas 3.55 and 3.57. $\square$

Corollary 3.60 *For all* $e \in Exp_\rho$ *and for all* $E\{{}_{-\rho}\} \in ECtx_\tau$, $\vdash E\{\pi^\infty\,(\pi^\infty\,e)\} \approx E\{\pi^\infty\,e\} : \tau$.

Proof Follows by Lemmas 3.56 and 3.58. $\square$

We then define a "compilation" relation for expressions that annotates terms with syntactic projections. The relation $\Gamma \vdash e : \tau \Rightarrow |e|$ is defined by induction on $\Gamma \vdash e : \tau$ by the axioms and inference rules in Figure 2. It is easy to see that if $\Gamma \vdash e : \tau$, then $\Gamma \vdash e : \tau \Rightarrow |e|$, for some $|e|$.

Lemma 3.61 *If* $\Gamma \vdash e : \tau \Rightarrow |e|$, *then* $\Gamma \vdash |e| : \tau$.

Proof By induction on $\Gamma \vdash e : \tau \Rightarrow |e|$. $\square$

For any $E\{{}_{-\tau}\} \in \mathrm{ECtx}_{\tau'}$, we define $|E|$ as follows. Clearly, $[z : \tau] \vdash E\{z\} : \tau'$. Thus for some e', $[z : \tau] \vdash E\{z\} : \tau' \Rightarrow e'$. By induction on the derivation there will be one free occurrence of z in e'. We define $|E| \stackrel{\text{def}}{=} [{}_{-\tau}/z]e'$, and by the remarks given here and Lemma 3.61, $|E|\{{}_{-\tau}\} \in \mathrm{ECtx}_{\tau'}$.

Lemma 3.62 *For all* $e \in Exp_\tau(\Gamma)$ *and for all expression substitutions* γ *for* Γ, *if* $\Gamma \vdash e : \tau \Rightarrow |e|$, *then* $\vdash \Pi^\infty_\tau\,(\gamma|e|) \approx \gamma|e| : \tau$.

$$\Gamma \vdash x : \tau \Rightarrow \Pi^{\infty}_{\tau}\, x \quad (\Gamma(x) = \tau) \tag{TR-VAR}$$

$$\Gamma \vdash * : 1 \Rightarrow \Pi^{\infty}_{1}\, * \tag{TR-ONE}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \Rightarrow |e_1| \qquad \Gamma \vdash e_2 : \tau_2 \Rightarrow |e_2|}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2 \Rightarrow \Pi^{\infty}_{\tau_1 \times \tau_2}\,(|e_1|, |e_2|)} \tag{TR-PROD}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \Rightarrow |e|}{\Gamma \vdash \mathsf{fst}\ e : \tau_1 \Rightarrow \mathsf{fst}\ |e|} \tag{TR-FST}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \Rightarrow |e|}{\Gamma \vdash \mathsf{snd}\ e : \tau_2 \Rightarrow \mathsf{snd}\ |e|} \tag{TR-SND}$$

$$\frac{\Gamma \vdash e : \tau_1 \Rightarrow |e|}{\Gamma \vdash \mathsf{inl}_{\tau_2}\ e : \tau_1 + \tau_2 \Rightarrow \Pi^{\infty}_{\tau_1 + \tau_2}\,(\mathsf{inl}_{\tau_2}\ |e|)} \tag{TR-INL}$$

$$\frac{\Gamma \vdash e : \tau_2 \Rightarrow |e|}{\Gamma \vdash \mathsf{inr}_{\tau_1}\ e : \tau_1 + \tau_2 \Rightarrow \Pi^{\infty}_{\tau_1 + \tau_2}\,(\mathsf{inr}_{\tau_1}\ |e|)} \tag{TR-INR}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 + \tau_2 \Rightarrow |e_1| \qquad \Gamma \vdash e_2 : \tau_1 \rightharpoonup \tau \Rightarrow |e_2| \qquad \Gamma \vdash e_3 : \tau_2 \rightharpoonup \tau \Rightarrow |e_3|}{\Gamma \vdash \mathsf{case}(e_1, e_2, e_3) : \tau \Rightarrow \mathsf{case}(|e_1|, |e_2|, |e_3|)} \tag{TR-CASE}$$

$$\frac{\Gamma[f : \tau_1 \rightharpoonup \tau_2][x : \tau_1] \vdash e : \tau_2 \Rightarrow |e|}{\Gamma \vdash \mathsf{fix}\ f(x{:}\tau_1).e : \tau_1 \rightharpoonup \tau_2 \Rightarrow \Pi^{\infty}_{\tau_1 \rightharpoonup \tau_2}\,(\mathsf{fix}\ f(x{:}\tau_1).|e|)} \quad (f, x \notin \mathrm{Dom}(\Gamma)) \tag{TR-FIX}$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \rightharpoonup \tau \Rightarrow |e_1| \qquad \Gamma \vdash e_2 : \tau_2 \Rightarrow |e_2|}{\Gamma \vdash e_1\, e_2 : \tau \Rightarrow |e_1|\, |e_2|} \tag{TR-APP}$$

$$\frac{\Gamma \vdash e : \rho \Rightarrow |e|}{\Gamma \vdash \mathsf{out}\ e : \tau_\rho \Rightarrow \mathsf{out}\ |e|} \tag{TR-OUT}$$

$$\frac{\Gamma \vdash e : \tau_\rho \Rightarrow |e|}{\Gamma \vdash \mathsf{in}\ e : \rho \Rightarrow \Pi^{\infty}_{\rho}\,(\mathsf{in}\ |e|)} \tag{TR-IN}$$

Figure 2: Definition of $\Gamma \vdash e : \tau \Rightarrow |e|$.

Proof By induction on $\Gamma \vdash e : \tau \Rightarrow |e|$.

Case TR-VAR, TR-ONE, TR-PROD, TR-INL, TR-INR, TR-FIX, TR-IN: Use Corollary 3.59.

Case TR-FST: By induction we get that

$$\vdash \gamma|e| \approx \Pi^{\infty}_{\tau_1 \times \tau_2} (\gamma|e|) : \tau_1 \times \tau_2 \tag{31}$$

We are to show $\vdash \mathsf{fst}\ (\gamma|e|) \approx \Pi^{\infty}_{\tau_1} (\mathsf{fst}\ (\gamma|e|)) : \tau_1 \times \tau_2$. If $\gamma|e| \Uparrow$ then it follows by Lemma 2.15. Thus assume that $\gamma|e| \Downarrow$, that is, that there exists $v \in \mathrm{Val}_{\tau_1 \times \tau_2}$ such that $\gamma|e| \mapsto^* v$. By Canonical Forms Lemma (Lemma 2.11), $v = (v_1, v_2)$ for some v_1, v_2. By (31), Lemmas 3.7 and 3.43 and transitivity of $\approx$,

$$\vdash \gamma|e| \approx (\Pi^{\infty}_{\tau_1} v_1, \Pi^{\infty}_{\tau_2} v_2) : \tau_1 \times \tau_2 \tag{32}$$

By Lemmas 3.7, 3.8, 3.9, and (32), we get

$$\vdash \mathsf{fst}\ (\gamma|e|) \approx \Pi^{\infty}_{\tau_1} v_1 : \tau_1 \tag{33}$$

Further, again using Lemmas 3.7 and 3.9,

$$\vdash \mathsf{fst}\ (\gamma|e|) \approx v_1 : \tau_1 \tag{34}$$

so by composition of evaluation contexts, (34) gives

$$\vdash \Pi^{\infty}_{\tau_1} (\mathsf{fst}\ (\gamma|e|)) \approx \Pi^{\infty}_{\tau_1} v_1 : \tau_1 \tag{35}$$

which together with (33) gives the required by transitivity and symmetry of $\approx$.

Case TR-SND: Similar to the case for TR-FST.

Case TR-CASE: We are to show that $\vdash \Pi^{\infty}_{\tau} (\mathsf{case}(\gamma|e_1|, \gamma|e_2|, \gamma|e_3|)) \approx \mathsf{case}(\gamma|e_1|, \gamma|e_2|, \gamma|e_3|) : \tau$. If $\gamma|e| \Uparrow$ then it follows by Lemma 2.15. Thus assume that $\gamma|e| \Downarrow$.

SubCase I: Assume $\gamma|e| \mapsto^* \mathsf{inl}_{\tau_2}\ v_1$. Then by Lemma 3.7, it suffices to show $\vdash \Pi^{\infty}_{\tau} (\gamma|e_2|\ (v_1)) \approx \gamma|e_2|\ (v_1) : \tau$. Assume $\gamma|e_2| \mapsto^* v$ (otherwise the required follows by Lemma 2.15). By induction we have

$$\vdash \gamma|e_2| \approx \Pi^{\infty}_{\tau_1 \rightharpoonup \tau} (\gamma|e_2|) : \tau_1 \rightharpoonup \tau$$

so by Lemma 3.7 and transitivity of $\approx$ we get

$$\vdash v \approx \Pi^{\infty}_{\tau_1 \rightharpoonup \tau} v : \tau_1 \rightharpoonup \tau$$

Thus it suffices to show

$$\vdash \Pi^{\infty}_{\tau}\,((\Pi^{\infty}_{\tau_1 \rightharpoonup \tau}\, v)\, v_1) \approx (\Pi^{\infty}_{\tau_1 \rightharpoonup \tau}\, v)\, v_1 : \tau$$

But

$$(\Pi^{\infty}_{\tau_1 \rightharpoonup \tau}\, v)\, v_1 \mapsto^* \Pi^{\infty}_{\tau}\,(v\,(\Pi^{\infty}_{\tau_1}\, v_1))$$

so by Lemma 3.7 and transitivity of $\approx$ it suffices to show

$$\vdash \Pi^{\infty}_{\tau}\,(\Pi^{\infty}_{\tau}\,(v\,(\Pi^{\infty}_{\tau_1}\, v_1))) \approx \Pi^{\infty}_{\tau}\,(v\,(\Pi^{\infty}_{\tau_1}\, v_1)) : \tau$$

but this follows from Corollary 3.59.

SubCase II: Assume $\gamma|e| \mapsto^* \mathsf{inr}_{\tau_1}\, v_1$. Similar to SubCase I.

Case TR-APP: Follows by induction and Corollary 3.59.

Case TR-OUT: Follows by induction and Corollary 3.59. □

Lemma 3.63 *For all $e \in Exp_{\tau}(\Gamma)$ and for all expression substitutions γ, γ' for Γ, if $\vdash \gamma \preceq \gamma' : \Gamma$ and $\Gamma \vdash e : \tau \Rightarrow |e|$, then $\vdash \gamma|e| \preceq \gamma'(e) : \tau$.*

Proof By induction on $\Gamma \vdash e : \tau \Rightarrow |e|$, using Lemma 3.31 and Lemma 3.50. For rule TR-FIX, by compactness it suffices to show, for all $i \in N$,

$$\vdash \gamma(\mathsf{fix}\ f^i(x{:}\tau_1).|e|) \preceq \gamma'(\mathsf{fix}\ f^i(x{:}\tau_1).(e)) : \tau$$

This is shown by induction on i using the outer induction hypothesis in the inductive step. □

Corollary 3.64 *If $\emptyset \vdash e : \tau \Rightarrow |e|$, then $\vdash |e| \preceq e : \tau$.*

Proof By Lemma 3.63. □

Corollary 3.65 *For all $E\{\_{\tau}\} \in ECtx_{\tau'}$ and for all expression substitutions γ for $\Gamma = [z : \tau]$, if $[z : \tau] \vdash E\{z\} \Rightarrow |E\{z\}|$, then $\vdash \gamma|E\{z\}| \preceq \gamma(E\{z\}) : \tau'$.*

Proof Follows by Lemma 3.63. □

Lemma 3.66 *For all $e \in Exp_{\tau}$ and for all $E\{\_{\tau}\} \in ECtx_{\tau'}$*

1. *If $[x_1 : \tau_1, \ldots, x_k : \tau_k] \vdash e \Rightarrow |e|$ and $\emptyset \vdash e_1 : \tau_1,\ \ldots,\ \emptyset \vdash e_k : \tau_k$, then*

$$\vdash |[e_1, \ldots, e_k/x_1, \ldots, x_k]e| \approx [|e_1|, \ldots, |e_k|/x_1, \ldots, x_k]|e| : \tau$$

2. $\vdash |E\{e\}| \approx |E|\{|e|\} : \tau'$.

Proof

1. By induction on $[x_1 : \tau_1, \ldots, x_k : \tau_k] \vdash e \Rightarrow |e|$.
2.

$$\begin{array}{rcll} |E\{e\}| & = & |[e/x]E\{x\}| & \text{by Lemma 2.3} \\ & = & [|e|/x]|E\{x\}| & \text{by 1} \\ & = & |E|\{|e|\} & \text{by Lemma 2.3} \end{array}$$

where for the last application of Lemma 2.3 note that the lemma indeed is applicable since $|E|$ *is* an evaluation context by the remarks on Page 31.

□

Lemma 3.67 *For all τ and for all $v \in Val_\tau$ the following holds.*

1. $|v| \Downarrow$

2. $\Pi^\infty_\tau\, |v| \Downarrow$

Proof By induction on v. □

Lemma 3.68 *For all $e \in Exp_\tau$, if $\emptyset \vdash e : \tau \Rightarrow |e|$ and $e \mapsto e'$, then $\vdash |e| \approx |e'| : \tau$, where $\emptyset \vdash e' : \tau \Rightarrow |e'|$*

Proof Assume $e \mapsto e'$. Then $e = E\{r\}$ for some E and r. We proceed by cases on the reduction rule applied. We will use Lemmas 3.7 and 3.8 repeatedly without explicit mentioning.

Case R-OUT: Then $r = \mathsf{out}\ (\mathsf{in}\ v)$ for some v. We reason as follows.

$$\begin{array}{rcll} |e| & = & |E\{r\}| & \\ & \approx & |E|\{|r|\} & \text{by Lemma 3.66, item 2} \\ & = & |E|\{|\mathsf{out}\ (\mathsf{in}\ v)|\} & \\ & = & |E|\{\mathsf{out}\ (\Pi^\infty_\rho\ (\mathsf{in}\ |v|))\} & \text{by definition} \\ & \approx & |E|\{\mathsf{out}\ (\Pi^\infty_\rho\ (\mathsf{in}\ v'))\} & \text{by Lemma 3.67, } \exists v' : |v| \mapsto^* v' \\ & \approx & |E|\{\mathsf{out}\ (\mathsf{in}\ (\Pi^\infty_{\tau_\rho}\ v'))\} & \text{by Lemmas 3.31 and 3.47} \\ & \approx & |E|\{\mathsf{out}\ (\mathsf{in}\ (\Pi^\infty_{\tau_\rho}\ |v|))\} & \\ & \approx & |E|\{\mathsf{out}\ (\mathsf{in}\ v'')\} & \text{by Lemma 3.67, } \exists v'' : \Pi^\infty_{\tau_\rho}\ |v| \mapsto^* v'' \\ & \approx & |E|\{v''\} & \text{by R-OUT} \\ & \approx & |E|\{\Pi^\infty_{\tau_\rho}\ |v|\} & \\ & \approx & |E|\{|v|\} & \text{by Lemma 3.62} \\ & \approx & |e'| & \end{array}$$

Case R-BETA: We reason as follows.

$$\begin{array}{rcll}
|e| & \approx & |E|\{|(\mathsf{fix}\ f(x{:}\tau_1).e_1)\ v|\} & \text{by Lemma 3.66, item 2} \\
& \approx & |E|\{(\Pi^{\infty}_{\tau_1 \rightharpoonup \tau_2}\ (\mathsf{fix}\ f(x{:}\tau_1).|e_1|))\ |v|\} & \text{by definition} \\
& \approx & |E|\{(\lambda x{:}\tau_1.\Pi^{\infty}_{\tau_2}\ ((\mathsf{fix}\ f(x{:}\tau_1).|e_1|)\ (\Pi^{\infty}_{\tau_1}\ x)))\ |v|\} & \text{by Lemma 3.45} \\
& \approx & |E|\{(\lambda x{:}\tau_1.\Pi^{\infty}_{\tau_2}\ ((\mathsf{fix}\ f(x{:}\tau_1).|e_1|)\ (\Pi^{\infty}_{\tau_1}\ x)))\ v'\} & \text{by Lemma 3.67, } \exists v' : |v| \mapsto^* v' \\
& \approx & |E|\{\Pi^{\infty}_{\tau_2}\ ((\mathsf{fix}\ f(x{:}\tau_1).|e_1|)\ (\Pi^{\infty}_{\tau_1}\ v'))\} & \text{by R-BETA} \\
& \approx & |E|\{\Pi^{\infty}_{\tau_2}\ ((\mathsf{fix}\ f(x{:}\tau_1).|e_1|)\ (\Pi^{\infty}_{\tau_1}\ |v|))\} & \\
& \approx & |E|\{\Pi^{\infty}_{\tau_2}\ ((\mathsf{fix}\ f(x{:}\tau_1).|e_1|)\ |v|)\} & \text{by Lemma 3.62} \\
& \approx & |E|\{\Pi^{\infty}_{\tau_2}\ ((\mathsf{fix}\ f(x{:}\tau_1).|e_1|)\ v')\} & \\
& \approx & |E|\{\Pi^{\infty}_{\tau_2}\ ([\mathsf{fix}\ f(x{:}\tau_1).|e_1|, v'/f, x]|e_1|)\} & \text{by R-BETA} \\
& \approx & |E|\{\Pi^{\infty}_{\tau_2}\ ([\mathsf{fix}\ f(x{:}\tau_1).|e_1|, |v|/f, x]|e_1|)\} & \text{by Lemma 3.31} \\
& \approx & |E|\{[\mathsf{fix}\ f(x{:}\tau_1).|e_1|, |v|/f, x]|e_1|\} & \text{by Lemma 3.62} \\
& \approx & |E|\{|[\mathsf{fix}\ f(x{:}\tau_1).e_1, v/f, x]e_1|\} & \text{by Lemma 3.66, item 1} \\
& \approx & |e'| & \text{by Lemma 3.66, item 2}
\end{array}$$

Case R-FST: We reason as follows.

$$\begin{array}{rcll}
|e| & \approx & |E|\{|\mathsf{fst}\ ((v_1, v_2))|\} & \text{by Lemma 3.66, item 2} \\
& \approx & |E|\{\mathsf{fst}\ ((|v_1|, |v_2|))\} & \text{by definition} \\
& \approx & |E|\{\mathsf{fst}\ ((v_1', v_2'))\} & \text{by Lemma 3.67, } \exists v_1' : |v_1| \mapsto^* v_1' \text{ and } \exists v_2' : |v_2| \mapsto^* v_2' \\
& \approx & |E|\{v_1'\} & \text{by R-FST} \\
& \approx & |E|\{|v_1|\} & \\
& \approx & |e'| & \text{by Lemma 3.66, item 2}
\end{array}$$

Case R-SND: Similar to the R-FST case.

Case R-CASE-INL: We reason as follows.

$$\begin{array}{rcll}
|e| & \approx & |E|\{|\mathsf{case}(\mathsf{inl}_{\tau_2}\ v, e_1, e_2)|\} & \text{by Lemma 3.66, item 2} \\
& \approx & |E|\{\mathsf{case}(\mathsf{inl}_{\tau_2}\ |v|, |e_1|, |e_2|)\} & \text{by definition} \\
& \approx & |E|\{\mathsf{case}(\mathsf{inl}_{\tau_2}\ v', |e_1|, |e_2|)\} & \text{by Lemma 3.67, } \exists v' : |v| \mapsto^* v' \\
& \approx & |E|\{|e_1|\ v'\} & \text{by R-CASE-INL} \\
& \approx & |E|\{|e_1|\ |v|\} & \\
& \approx & |E|\{|e_1\ v|\} & \text{by definition} \\
& \approx & |e'| & \text{by Lemma 3.66, item 2}
\end{array}$$

Case R-CASE-INR: Similar to the R-CASE-INL case. □

Lemma 3.69 $\vdash \lambda x{:}\rho.x \preceq \pi^{\infty} : \rho \rightharpoonup \rho$

Proof By Corollary 3.14 and Canonical Forms Lemma (Lemma 2.11) it suffices to show, for all $E\{\_{\rho \rightharpoonup \rho}\ (\mathsf{in}\ v)\} \in \mathrm{ECtx}_1$,

$$E\{\lambda x{:}\rho.x\ (\mathsf{in}\ v)\} \preceq^k E\{\pi^{\infty}\ (\mathsf{in}\ v)\}$$

Let $E\{_{-\rho\rightharpoonup\rho}(\mathsf{in}\ v)\} \in \mathrm{ECtx}_1$ be arbitrary. By Lemma 3.6, it then suffices to show,

$$E\{\mathsf{in}\ v\} \preceq^k E\{\pi^\infty\,(\mathsf{in}\ v)\}$$

By Corollary 3.64 it then suffices to show,

$$E\{\mathsf{in}\ v\} \preceq^k E\{\pi^\infty\,|\mathsf{in}\ v|\}$$

Since clearly $\vdash \pi^\infty \approx \Pi^\infty_\rho : \rho \rightharpoonup \rho$, by Lemma 3.62 it then suffices to show,

$$E\{\mathsf{in}\ v\} \preceq^k E\{|\mathsf{in}\ v|\} \tag{36}$$

Suppose that

$$E\{\mathsf{in}\ v\} \preceq^k |E\{\mathsf{in}\ v\}| \tag{37}$$

holds. Assuming this, we can reason as follows

$$\begin{array}{rcll} E\{\mathsf{in}\ v\} \mapsto^* * & \Rightarrow & |E\{\mathsf{in}\ v\}| \mapsto^* * & \text{by assumption (37)} \\ & \Rightarrow & |E|\{|\mathsf{in}\ v|\} \mapsto^* * & \text{by Lemma 3.66, item 2} \\ & \Rightarrow & E\{|\mathsf{in}\ v|\} \mapsto^* * & \text{by Corollary 3.65} \end{array}$$

which gives (36) as required.

Thus we are left with showing (37). Clearly this follows from showing, for all closed expressions $e \in \mathrm{Exp}_1$,

$$e \mapsto^* * \Rightarrow |e| \mapsto^* *$$

We show this by induction on the length m of the computation of $e \mapsto^* *$.

Basis $(m = 0)$*:* Then $e = *$, whence $|e| = \Pi^\infty_1 * \mapsto^* *$, as required.

Inductive Step: Assume $e \mapsto e' \mapsto^m *$. Then by induction we get that $|e'| \mapsto^* *$. By Lemma 3.68, also $|e| \mapsto^* *$, as required. □

We are now in a position to establish the following theorem, which we refer to as the syntactic minimal invariant property by analogy to the domain-theoretic work of Pitts [17].

Theorem 3.70 (Syntactic Minimal Invariance) $\vdash \pi^\infty \approx \lambda x{:}\rho.x : \rho \rightharpoonup \rho$

Proof By Lemmas 3.50, 3.69, and 3.3. □

3.3 Summary

In this section we have defined a notion of experimental approximation and experimental equivalence between terms and established some basic equivalences of terms. Further, we have seen that the finite unrollings of a given fix-term forms a chain with respect to the approximation pre-order and that the fix-term itself is the least upper bound of this chain. This has been crucial to establish the syntactical minimal invariant property for the recursive type ρ, that is, that the projection term π^∞ associated with the recursive type ρ is operationally equivalent to the identity term $\lambda x{:}\rho.x$.

In the following we shall show how to construct relations over equivalence classes of terms (with respect to the operational equivalence). The properties established in this section are crucial to this construction, in particular, the syntactial minimal invariant property plays a central rôle in adapting Pitts' method [17] to our operational setting.

4 Relations

In this and the following section we shall show how to construct a relational interpretation of types over an operational semantics. We shall end up by showing "The Fundamental Theorem of Logical Relations" which states that the relational interpretation of types is sound in the sense that well-typed terms are related to themselves by the relation associated to their type. The constructed relations can be seen to provide a notion of equality of terms, which we shall refer to as "logical equivalence". In Section 6 we define this notion of equivalence and show that it coincides with contextual equivalence. Moreover, we derive a useful coinduction principle for establishing logical equivalence and thus contextual equivalence. This section also provides the necessary understanding for constructing a relational interpretation, which we can use to show the correctness of cps transformation in Section 7.

In this section we define a universe of relations over equivalence classes of closed expressions, with respect to operational equivalence. Further, we define a notion of admissibility for relations. This corresponds to the notion of admissibility (also known as inclusiveness or completeness) used in domain theory, and is also here used as a condition on relations, which, loosely speaking, allows one to show that a fix-term is in a relation by showing that its approximants are in the relation. Next we show that admissible relations equipped with the obvious ordering form a complete lattice, define relational constructors corresponding to the type constructors of the language, and show that these constructors preserve admissibility.

Throughout this section we will let $n \in N$ be an arbitrary but fixed natural number, that is, we will consider n-ary relations for a fixed, but arbitrary $n \in N$. We will use the same abbreviations for terms involving fix and for contexts as in Section 3.1. For any set A and natural number m we write A^m for the m-ary cartesian product of A. For any set A and any equivalence relation $\equiv$ on A, we write $A/\!\equiv$ for the set of equivalence classes of A with respect to $\equiv$. To simplify notation we denote each equivalence class by one of its representatives. Moreover, we will simply use $\approx$ for the operational equivalence relation at type τ (i.e., $(e, e') \in \approx \iff \vdash e \approx e' : \tau$) when τ is clear from context.

Definition 4.1 *For all τ, we define a universe of n-ary relations Rel_τ as follows.*

$$Rel_\tau \stackrel{\text{def}}{=} \mathcal{P}\left((Exp_\tau /\!\approx)^n\right)$$

We use R to range over Rel_τ.

Definition 4.2 *A relation $R \in Rel_\tau$ is* admissible *if and only if it satisfies both of the following two conditions.*

Strictness: $(e_1, \ldots, e_n) \in R$ *if and only if* $((\forall i \in 1..n : e_i \Uparrow) \vee (\forall i \in 1..n : \exists v_i : e_i \mapsto^* v_i \wedge (v_1, \ldots, v_n) \in R))$

Completeness: *For all $i \in 1..n$ and for all $C_i\{\vec{\mathsf{p}}\} \in Ctx_\tau$ with all parameters in $\vec{\mathsf{p}}$ of type $\tau_1 \rightharpoonup \tau_2$ and for all $F^i_\omega = \mathsf{fix}\, f(x{:}\tau_1).e_i \in Exp_{\tau_1 \rightharpoonup \tau_2}$, and for all $I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|})$,*

$$\left(\forall \vec{m} \in I : (C_1\{F^1_{\vec{m}}\}, \ldots, C_n\{F^n_{\vec{m}}\}) \in R\right) \Rightarrow$$

$$\left((C_1\{F^1_\omega\}, \ldots, C_n\{F^n_\omega\}) \in R\right)$$

Recall that $C\{\vec{\mathsf{p}}\}$ means that all of the parameters of C are *included* in $\vec{\mathsf{p}}$, that is, in the completeness condition the contexts C_i are not required to all have the same number of parameters.

The completeness condition on relations is motivated as follows. For simplicity, let us just consider unary relations ($n = 1$). We wish to impose a completeness property that allows us to conclude that $C\{F_\omega\} \in R$ based on whether some collection of finite unrollings of $C\{F_\omega\}$ are in R. Clearly, it is not sufficient to establish that $C\{F_i\} \in R$ for some $i \geq 0$, since $C\{F_i\}$ may fail to terminate (and hence lie in R by the strictness condition on relations), whereas $C\{F_\omega\}$ may terminate with some value. This suggests that it may be sufficient to establish that $C\{F_i\} \in R$ for some i such that

$C\{F_i\}$ terminates. But such a weak notion of completeness would not be closed under the formation of function spaces between relations. Knowing that $C\{F_i\}$ terminates and that $C\{F_i\} \in R_1 \rightharpoonup R_2$ does not entail that there exists i' such that $C\{F_{i'}\}(e)$ terminates and lies in R_2. Consequently we must assume that for every i there is a larger i' such that $C\{F_i\} \in R$ so that in the case of $R = R_1 \rightharpoonup R_2$ we may pick a large enough i' to ensure that an application $C\{F_{i'}\}(e)$ terminates and hence lies in R_2. The completeness condition we have stated here ensures that this is the case.

Definition 4.3 *For all τ, we define a universe of admissible n-ary relations $Radm_\tau$ as follows.*

$$Radm_\tau \stackrel{\text{def}}{=} \{ R \in Rel_\tau \mid R \textit{ is admissible} \}$$

We also use R to range over $Radm_\tau$.

We now define a series of relational constructors corresponding to the syntactic type constructors. For each of these constructors it is easy to verify that the definition does not depend on the choice of representative of an operational equivalence class.

Definition 4.4

$$R_0 \stackrel{\text{def}}{=} \{ (e_1, \ldots, e_n) \in (Exp_0 /\!\approx)^n \mid \forall i \in 1..n : e_i \Uparrow \}$$

Definition 4.5

$$R_1 \stackrel{\text{def}}{=} \{ (e_1, \ldots, e_n) \in (Exp_1 /\!\approx)^n \mid (\forall i \in 1..n : e_i \Uparrow) \vee (\forall i \in 1..n : e_i \mapsto^* *) \}$$

Definition 4.6 *For all $R_1 \in Rel_{\tau_1}$ and $R_2 \in Rel_{\tau_2}$,*

$$\begin{array}{rcl} R_1 \times R_2 & \stackrel{\text{def}}{=} & \{ (e_1, \ldots, e_n) \in (Exp_{\tau_1 \times \tau_2} /\!\approx)^n \mid \\ & & (\forall i \in 1..n : e_i \Uparrow) \vee \\ & & (\forall i \in 1..n : \exists v_i, v_i' : \ \vdash e_i \approx (v_i, v_i') : \tau_1 \times \tau_2 \\ & & \qquad\qquad \wedge (v_1, \ldots, v_n) \in R_1 \wedge (v_1', \ldots, v_n') \in R_2) \} \end{array}$$

Definition 4.7 *For all $R_1 \in Rel_{\tau_1}$ and $R_2 \in Rel_{\tau_2}$,*

$$\begin{array}{rcl} R_1 + R_2 & \stackrel{\text{def}}{=} & \{ (e_1, \ldots, e_n) \in (Exp_{\tau_1 + \tau_2} /\!\approx)^n \mid \\ & & (\forall i \in 1..n : e_i \Uparrow) \vee \\ & & (\forall i \in 1..n : \exists v_i : \ \vdash e_i \approx \mathsf{inl}_{\tau_2}\ v_i : \tau_1 + \tau_2 \wedge (v_1, \ldots, v_n) \in R_1) \\ & & (\forall i \in 1..n : \exists v_i : \ \vdash e_i \approx \mathsf{inr}_{\tau_1}\ v_i : \tau_1 + \tau_2 \wedge (v_1, \ldots, v_n) \in R_2) \} \end{array}$$

Definition 4.8 *For all $R_1 \in Rel_{\tau_1}$ and $R_2 \in Rel_{\tau_2}$,*

$$\begin{aligned} R_1 \rightharpoonup R_2 \overset{\text{def}}{=} \{ (e_1, \ldots, e_n) \in (Exp_{\tau_1 \rightharpoonup \tau_2}/{\approx})^n \mid \\ (\forall i \in 1..n : e_i \Uparrow) \vee \\ (\forall i \in 1..n : \exists v_i : \vdash e_i \approx v_i : \tau_1 \rightharpoonup \tau_2 \wedge ((e'_1, \ldots, e'_n) \in R_1 \Rightarrow \\ (v_1\, e'_1, \ldots, v_n\, e'_n) \in R_2)) \} \end{aligned}$$

Lemma 4.9 *For all τ, $(Radm_\tau, \subseteq)$ is a complete lattice.*

Proof By a standard lattice-theory theorem (see, e.g., [2, Theorem 2.16(ii)]) it suffices to show that the greatest lower bound, $\bigwedge S$, exists for every subset of $Radm_\tau$. Thus let S be an arbitrary subset of $Radm_\tau$. Define $\bigwedge S \overset{\text{def}}{=} \bigcap S$. We then have to show

1. $\bigwedge S \in Radm_\tau$
2. $\bigwedge S$ is the greatest lower bound of S

Item 2 is obvious by the definitions. To prove item 1 we have to show that the two conditions in the definition of admissibility are satisfied. They both follow easily using the fact that each $R \in S$ is admissible. $\square$

We now proceed to show that the relational constructors preserve admissibility. To this end we shall employ the following lemma about the $\Downarrow^F$ relation, which was defined in Section 3.1.

Lemma 4.10 *For all $i \in 1..n$ and for all contexts $C_i\{\vec{\mathsf{p}}\}$ and all value contexts $V_i\{\vec{\mathsf{p}_i}\}$ satisfying $C_i\{\vec{\mathsf{p}}\} \Downarrow^F V_i\{\vec{\mathsf{p}_i}\}$, there exists a $\vec{\mathsf{p}}'$ such that for all $i \in 1..n$, $C_i\{\vec{\mathsf{p}}\} \Downarrow^F V_i\{\vec{\mathsf{p}}'\}$ and furthermore, for all $I \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}|})$, letting*

$$I_i \overset{\text{def}}{=} \{ \vec{m}\vec{m}' \mid \vec{m} \in I \wedge C_i\{F^i_{\vec{m}}\} \mapsto^* V_i\{F^i_{\vec{m}'}\} \} \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|})$$

then

$$I' \overset{\text{def}}{=} \bigcap_{i=1}^{n} I_i$$

is a cofinal subset of $N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|}$.

Proof Since $\Downarrow^F$ is preserved under renaming of parameters we can assume without loss of generality that all parameters p_{ij} are distinct. Let $\vec{\mathsf{p}}' = \vec{\mathsf{p}_1} \cdots \vec{\mathsf{p}_n}$. The result follows by Lemma 3.20 and simple properties of cofinal sets (it is the fact that each V_i involve a distinct subset of the parameters of $\vec{\mathsf{p}}'$ that ensures that the intersection defining I' indeed is a cofinal set). ◻

We will also make use of the following lemma to show admissibility of the relational constructors.

Lemma 4.11 *For all $i \in 1..n$, all $C_i\{\vec{\mathsf{p}}\}$, and for all $R_1 \in Radm_{\tau_1}$ and $R_2 \in Radm_{\tau_2}$ if the following conditions are all satisfied*

1. *R is either R_0, R_1, $R_1 \times R_2$, $R_1 + R_2$, or $R_1 \rightharpoonup R_2$*
2. *$\forall \vec{m} \in I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|}) : (C_1\{F^1_{\vec{m}}\}, \ldots, C_n\{F^n_{\vec{m}}\}) \in R$*
3. *each R is strict*

then

$$(\forall i \in 1..n : C_i\{F^i_{\vec{\omega}}\} \Downarrow) \vee (\forall i \in 1..n : C_i\{F^i_{\vec{\omega}}\} \Uparrow)$$

Proof By contradiction using Lemma 3.22. ◻

Lemma 4.12 *For all $R_1 \in Radm_{\tau_1}$ and all $R_2 \in Radm_{\tau_2}$, $R_1 \times R_2 \in Radm_{\tau_1 \times \tau_2}$.*

Proof We are to show that the two conditions of admissibility hold.

Strictness Follows by Lemmas 3.5, 3.7. and 3.8.

Completeness Let $I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|})$. Assume

$$\forall \vec{m} \in I : (C_1\{F^1_{\vec{m}}\}, \ldots, C_n\{F^n_{\vec{m}}\}) \in R_1 \times R_2 \tag{38}$$

By Lemma 4.11 (note that we have already argued that the strictness condition of admissibility is satisfied) there are two cases to consider.

Case I: $\forall i \in 1..n : C_i\{F^i_{\vec{\omega}}\} \Uparrow$. Then the desired follows by definition of $R_1 \times R_2$.

Case II: $\forall i \in 1..n : C_i\{F^i_{\vec{\omega}}\} \Downarrow$. Then $\forall i \in 1..n : \exists v_i : C_i\{F^i_{\vec{\omega}}\} \mapsto^* v_i$. By Lemma 3.22, for all $i \in 1..n$ there exists a $V_i\{\vec{\mathsf{p}_i}\}$ such that $v_i = V_i\{F^i_{\vec{\omega}}\}$ and $C_i\{\vec{\mathsf{p}}\} \Downarrow^F V_i\{\vec{\mathsf{p}_i}\}$. Thus by Lemma 4.10, there exists a $\vec{\mathsf{p}}'$ such that for all $i \in 1..n$, $C_i\{\vec{\mathsf{p}}\} \Downarrow^F V_i\{\vec{\mathsf{p}}'\}$ and

$$I_i \stackrel{\mathrm{def}}{=} \{\, \vec{m}\vec{m}' \mid \vec{m} \in I \wedge C_i\{F^i_{\vec{m}}\} \mapsto^* V_i\{F^i_{\vec{m}'}\} \,\} \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|})$$

and

$$I' \stackrel{\text{def}}{=} \bigcap_{i=1}^{n} I_i \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|})$$

Let

$$I'' \stackrel{\text{def}}{=} \{\, \vec{m}' \mid \vec{m} \in I \wedge \vec{m}\vec{m}' \in I' \,\}$$

Clearly, $I'' \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}'|})$. By (38), Lemma 3.7 and definition of I'', we have,

$$\forall \vec{m} \in I'' : (V_1\{F^1_{\vec{m}}\}, \ldots, V_n\{F^n_{\vec{m}}\}) \in R_1 \times R_2 \tag{39}$$

By Canonical Forms Lemma, for all $i \in 1..n$, there exist V_{i1}, V_{i2} such that $V_i = (V_{i1}, V_{i2})$, and by (39) and definition of $R_1 \times R_2$ we then have

$$\forall \vec{m} \in I'' : (V_{11}\{F^1_{\vec{m}}\}, \ldots, V_{n1}\{F^n_{\vec{m}}\}) \in R_1 \tag{40}$$

and

$$\forall \vec{m} \in I'' : (V_{12}\{F^1_{\vec{m}}\}, \ldots, V_{n2}\{F^n_{\vec{m}}\}) \in R_2 \tag{41}$$

By admissibility of R_1 and (40) we then get

$$(V_{11}\{F^1_{\vec{\omega}}\}, \ldots, V_{n1}\{F^n_{\vec{\omega}}\}) \in R_1 \tag{42}$$

and by admisibility of R_2 and (41) we get

$$(V_{12}\{F^1_{\vec{\omega}}\}, \ldots, V_{n2}\{F^n_{\vec{\omega}}\}) \in R_2 \tag{43}$$

Hence, by definition of $R_1 \times R_2$ we then have

$$(V_1\{F^1_{\vec{\omega}}\}, \ldots, V_n\{F^n_{\vec{\omega}}\}) \in R_1 \times R_2 \tag{44}$$

which together with Lemma 3.7 (and recalling that the relations are over equivalence classes w.r.t. operational equivalence) gives that

$$(C_1\{F^1_{\vec{\omega}}\}, \ldots, C_n\{F^n_{\vec{\omega}}\}) \in R_1 \times R_2$$

as required.

□

Lemma 4.13 *For all* $R_1 \in Radm_{\tau_1}$ *and all* $R_2 \in Radm_{\tau_2}$, $R_1 + R_2 \in Radm_{\tau_1+\tau_2}$.

Proof We are to show that the two conditions of admissibility hold.

Strictness Follows by Lemmas 3.5, 3.7. and 3.8.

Completeness Let $I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|})$. Assume

$$\forall \vec{m} \in I : (C_1\{F^1_{\vec{m}}\}, \ldots, C_n\{F^n_{\vec{m}}\}) \in R_1 + R_2 \tag{45}$$

By Lemma 4.11 (note that we have already argued that the strictness condition of admissibility is satisfied) there are two cases to consider.

Case I: $\forall i \in 1..n : C_i\{F^i_{\vec{\omega}}\} \Uparrow$. Then the desired follows by definition of $R_1 + R_2$.

Case II: $\forall i \in 1..n : C_i\{F^i_{\vec{\omega}}\} \Downarrow$. Then $\forall i \in 1..n : \exists v_i : C_i\{F^i_{\vec{\omega}}\} \mapsto^* v_i$. By Lemma 3.22, for all $i \in 1..n$ there exists a $V_i\{\vec{\mathsf{p}}_i\}$ such that $v_i = V_i\{F^i_{\vec{\omega}}\}$ and $C_i\{\vec{\mathsf{p}}\} \Downarrow^F V_i\{\vec{\mathsf{p}}_i\}$. Thus by Lemma 4.10, there exists a $\vec{\mathsf{p}}'$ such that for all $i \in 1..n$, $C_i\{\vec{\mathsf{p}}\} \Downarrow^F V_i\{\vec{\mathsf{p}}'\}$ and

$$I_i \stackrel{\mathrm{def}}{=} \{\, \vec{m}\vec{m}' \mid \vec{m} \in I \wedge C_i\{F^i_{\vec{m}}\} \mapsto^* V_i\{F^i_{\vec{m}'}\}\,\} \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|}) \tag{46}$$

and

$$I' \stackrel{\mathrm{def}}{=} \bigcap_{i=1}^{n} I_i \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|})$$

Let

$$I'' \stackrel{\mathrm{def}}{=} \{\, \vec{m}' \mid \vec{m} \in I \wedge \vec{m}\vec{m}' \in I'\,\}$$

Clearly, $I'' \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}'|})$. By (45), Lemma 3.7 and definition of I'', we have,

$$\forall \vec{m} \in I'' : (V_1\{F^1_{\vec{m}}\}, \ldots, V_n\{F^n_{\vec{m}}\}) \in R_1 + R_2 \tag{47}$$

By Canonical Forms Lemma,

$$\forall i \in 1..n : ((\exists V_{i1} : V_i = \mathsf{inl}_{\tau_2}\ V_{i1}) \vee (\exists V_{i2} : V_i = \mathsf{inr}_{\tau_1}\ V_{i2}))$$

Claim:

$$(\forall i \in 1..n : \exists V_{i1} : V_i = \mathsf{inl}_{\tau_2}\ V_{i1}) \vee (\forall i \in 1..n : \exists V_{i2} : V_i = \mathsf{inl}_{\tau_1}\ V_{i2})$$

Proof of Claim: By contradiction (of the assumption (45)), using Lemma 3.7, and (46). *(End of Proof of Claim)*

Thus there are two subcases to consider.

SubCase I: $\forall i \in 1..n : \exists V_{i1} : V_i = \mathsf{inl}_{\tau_2}\ V_{i1}$. Now proceed as in the proof of Lemma 4.12, using admissibility of R_1.

SubCase II: $\forall i \in 1..n : \exists V_{i2} : V_i = \mathsf{inl}_{\tau_1}\ V_{i2}$. Now proceed as in the proof of Lemma 4.12, using admissibility of R_2.

$\square$

Lemma 4.14 *For all $R_1 \in Rel_{\tau_1}$ and all $R_2 \in Radm_{\tau_2}$, $R_1 \rightharpoonup R_2 \in Radm_{\tau_1 \rightharpoonup \tau_2}$.*

Proof We are to show that the two conditions of admissibility hold.

Strictness Follows by Lemmas 3.5, 3.7. and 3.8.

Completeness Let $I \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}|})$. Assume

$$\forall \vec{m} \in I : (C_1\{F^1_{\vec{m}}\}, \ldots, C_n\{F^n_{\vec{m}}\}) \in R_1 \rightharpoonup R_2 \tag{48}$$

By Lemma 4.11 (note that we have already argued that the strictness condition of admissibility is satisfied) there are two cases to consider.

Case I: $\forall i \in 1..n : C_i\{F^i_{\vec{\omega}}\} \Uparrow$. Then the desired follows by definition of $R_1 \rightharpoonup R_2$.

Case II: $\forall i \in 1..n : C_i\{F^i_{\vec{\omega}}\} \Downarrow$. Then $\forall i \in 1..n : \exists v_i : C_i\{F^i_{\vec{\omega}}\} \mapsto^* v_i$. By Lemma 3.22, for all $i \in 1..n$ there exists a $V_i\{\vec{\mathsf{p}_i}\}$ such that $v_i = V_i\{F^i_{\vec{\omega}}\}$ and $C_i\{\vec{\mathsf{p}}\} \Downarrow^F V_i\{\vec{\mathsf{p}_i}\}$. Thus by Lemma 4.10, there exists a $\vec{\mathsf{p}}'$ such that for all $i \in 1..n$, $C_i\{\vec{\mathsf{p}}\} \Downarrow^F V_i\{\vec{\mathsf{p}}'\}$ and

$$I_i \stackrel{\text{def}}{=} \{\, \vec{m}\vec{m}' \mid \vec{m} \in I \wedge C_i\{F^i_{\vec{m}}\} \mapsto^* V_i\{F^i_{\vec{m}'}\}\,\} \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|}) \tag{49}$$

and

$$I' \stackrel{\text{def}}{=} \bigcap_{i=1}^{n} I_i \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|})$$

Let

$$I'' \stackrel{\text{def}}{=} \{\, \vec{m}' \mid \vec{m} \in I \wedge \vec{m}\vec{m}' \in I'\,\}$$

Clearly, $I'' \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}'|})$. By (48), Lemma 3.7 and definition of I'', we have,

$$\forall \vec{m} \in I'' : (V_1\{F^1_{\vec{m}}\}, \ldots, V_n\{F^n_{\vec{m}}\}) \in R_1 \rightharpoonup R_2 \tag{50}$$

Hence by definition of $R_1 \rightharpoonup R_2$

$$\forall \vec{m} \in I'' : \forall (e'_1, \ldots, e'_n) \in R_1 : (V_1\{F^1_{\vec{m}}\}\, e'_1, \ldots, V_n\{F^n_{\vec{m}}\}\, e'_n) \in R_2 \tag{51}$$

Let $(e'_1, \ldots, e'_n) \in R_1$ be arbitrary. Then by (51) we have

$$\forall \vec{m} \in I'' : (V_1\{F^1_{\vec{m}}\}\, e'_1, \ldots, V_n\{F^n_{\vec{m}}\}\, e'_n) \in R_2 \tag{52}$$

whence by admissibility of R_2, also

$$(V_1\{F^1_{\vec{\omega}}\}\, e'_1, \ldots, V_n\{F^n_{\vec{\omega}}\}\, e'_n) \in R_2 \tag{53}$$

Since $(e'_1, \ldots, e'_n)$ was arbitrary and using Lemma 3.7 we have that

$$(C_1\{F^1_{\vec{\omega}}\}, \ldots, C_n\{F^n_{\vec{\omega}}\}) \in R_1 \rightharpoonup R_2$$

as required.

□

Lemma 4.15 $R_1 \in Radm_1$.

Proof We are to show that the two conditions of admissibility hold.

Strictness Follows by Lemmas 3.5, 3.7. and 3.8.

Completeness Let $I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|})$. Assume

$$\forall \vec{m} \in I : (C_1\{F^1_{\vec{m}}\}, \ldots, C_n\{F^n_{\vec{m}}\}) \in R_1 \tag{54}$$

By Lemma 4.11 (note that we have already argued that the strictness condition of admissibility is satisfied) there are two cases to consider.

Case I: $\forall i \in 1..n : C_i\{F^i_{\vec{\omega}}\} \Uparrow$. Then the desired follows by definition of R_N.

Case II: $\forall i \in 1..n : C_i\{F^i_{\vec{\omega}}\} \Downarrow$. Then $\forall i \in 1..n : C_i\{F^i_{\vec{\omega}}\} \mapsto^* *$. By Lemma 3.22, for all $i \in 1..n$ there exists a $V_i\{\vec{\mathsf{p}}_i\}$ such that $* = V_i\{F^i_{\vec{\omega}}\}$ and $C_i\{\vec{\mathsf{p}}\} \Downarrow^F V_i\{\vec{\mathsf{p}}_i\}$. Thus by Lemma 4.10, there exists a $\vec{\mathsf{p}}'$ such that for all $i \in 1..n$, $C_i\{\vec{\mathsf{p}}\} \Downarrow^F V_i\{\vec{\mathsf{p}}'\}$ and

$$I_i \stackrel{\mathrm{def}}{=} \{\, \vec{m}\vec{m}' \mid \vec{m} \in I \wedge C_i\{F^i_{\vec{m}}\} \mapsto^* V_i\{F^i_{\vec{m}'}\}\,\} \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|}) \tag{55}$$

and

$$I' \stackrel{\mathrm{def}}{=} \bigcap_{i=1}^{n} I_i \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|})$$

Let

$$I'' \stackrel{\mathrm{def}}{=} \{\, \vec{m}' \mid \vec{m} \in I \wedge \vec{m}\vec{m}' \in I' \,\}$$

Clearly, $I'' \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}'|})$. Clearly, $V_i = *$. Since I'' is cofinal, in particular it is non-empty, so by (54) we have $(*, \ldots, *) \in R_1$. Whence, by Lemma 3.7 we have that

$$(C_1\{F^1_{\vec{\omega}}\}, \ldots, C_n\{F^n_{\vec{\omega}}\}) \in R_1$$

as required.

□

Lemma 4.16 $R_0 \in Radm_0$.

Proof Immediate by the definition of R_0 and the fact that, for all $e \in \mathrm{Exp}_0$, $e \Uparrow$; the latter follows from progress and the fact that there are no values of type 0 (formally, by Theorem 2.12 and Lemma 2.11). □

5 Relational Interpretation

In this section we give a relational interpretation of the types of $\mathcal{L}$, that is, an assignment of admissible relations to each type. To interpret the different type constructors we, of course, make use of the corresponding relational constructors defined in the previous section. Our construction follows along the lines of Pitts [17].

Definition 5.1 *For all τ, define $[\![\tau]\!] : Radm_\rho \to Radm_\tau$ by induction on τ as follows.*

$$\begin{array}{lcl}
[\![0]\!]R & = & R_0 \\
[\![1]\!]R & = & R_1 \\
[\![\rho]\!]R & = & R \\
[\![\tau_1 \times \tau_2]\!]R & = & [\![\tau_1]\!]R \times [\![\tau_2]\!]R \\
[\![\tau_1 + \tau_2]\!]R & = & [\![\tau_1]\!]R + [\![\tau_2]\!]R \\
[\![\tau_1 \rightharpoonup \tau_2]\!]R & = & [\![\tau_1]\!]R \rightharpoonup [\![\tau_2]\!]R
\end{array}$$

Note that the operation $[\![\tau]\!]$ is well-defined by induction on τ and Lemmas 4.9–4.16.

Definition 5.2 *Define $\Phi : Radm_\rho \to Radm_\rho$ by*

$$\begin{array}{rcl}
\Phi(R) & \stackrel{\text{def}}{=} & \{\, (e_1, \ldots, e_n) \in (Exp_\rho/\!\approx)^n \mid \\
& & \quad (\forall i \in 1..n : e_i \Uparrow) \vee (\forall i \in 1..n : \exists v_i :\ \vdash e_i \approx \mathsf{in}\ v_i : \rho\ \wedge \\
& & \qquad\qquad (v_1, \ldots, v_n) \in [\![\tau_\rho]\!]R) \,\}
\end{array}$$

Lemma 5.3 Φ *is well-defined.*

Proof First note that the definition does not depend on the chosen equivalence class representatives (by Lemma 3.5 and transitivity of $\approx$). Let $R \in Radm_\rho$. We are to show that $\Phi(R)$ is admissible. Use the fact that $[\![\tau_\rho]\!]R$ is admissible and proceed as in Lemma 4.12. □

Lemma 5.4 *($Radm^{op} \times Radm$) ordered componentwise is a complete lattice.*

Proof Follows by Lemma 4.9 $\square$

Definition 5.5 *For all τ, define $[\![\tau]\!]' : (Rel_\rho^{op} \times Radm_\rho) \to Radm_\tau$ by induction on τ as follows.*

$$\begin{array}{lcl}
[\![0]\!]'(R^-, R^+) & = & R_0 \\
[\![1]\!]'(R^-, R^+) & = & R_1 \\
[\![\rho]\!]'(R^-, R^+) & = & R^+ \\
[\![\tau_1 \times \tau_2]\!]'(R^-, R^+) & = & [\![\tau_1]\!]'(R^-, R^+) \times [\![\tau_2]\!]'(R^-, R^+) \\
[\![\tau_1 + \tau_2]\!]'(R^-, R^+) & = & [\![\tau_1]\!]'(R^-, R^+) + [\![\tau_2]\!]'(R^-, R^+) \\
[\![\tau_1 \rightharpoonup \tau_2]\!]'(R^-, R^+) & = & [\![\tau_1]\!]'(R^+, R^-) \rightharpoonup [\![\tau_2]\!]'(R^-, R^+)
\end{array}$$

Note that the operation $[\![\tau]\!]'$ is well-defined by induction on τ and Lemmas 4.9–4.16. Moreover, note that the first argument to $[\![\tau]\!]'$ is not required to be admissible; this will be useful in the following section.

Definition 5.6 *Define $\Psi : (Radm_\rho^{op} \times Radm_\rho) \to Radm_\rho$ by*

$$\begin{array}{lll}
\Psi(R^-, R^+) & \stackrel{\text{def}}{=} & \{ (e_1, \ldots, e_n) \in (Exp_\rho /\!\approx)^n \mid \\
& & (\forall i \in 1..n : e_i \Uparrow) \vee (\forall i \in 1..n : \exists v_i : \vdash e_i \approx \mathsf{in}\ v_i : \rho \wedge \\
& & \qquad (v_1, \ldots, v_n) \in [\![\tau_\rho]\!]'(R^-, R^+)) \}
\end{array}$$

Lemma 5.7 *Ψ is well-defined.*

Proof As in the proof of 5.3. $\square$

Definition 5.8 *Define $\Psi^\S : (Radm_\rho^{op} \times Radm_\rho) \to (Radm_\rho^{op} \times Radm_\rho)$ as follows.*

$$\Psi^\S(R^-, R^+) = (\Psi(R^+, R^-), \Psi(R^-, R^+))$$

Lemma 5.9 *$\Psi^\S$ is monotone.*

Proof By induction on τ using monotonicity properties of the relational constructors in the obvious way. $\square$

Definition 5.10 *By Lemma 5.9 and 5.4 and Tarski's fixed point theorem, $\Psi^\S$ has a least fixed point* $\mathrm{lfp}(\Psi^\S)$*. Define $(\Delta^-, \Delta^+) \stackrel{\text{def}}{=} \mathrm{lfp}(\Psi^\S)$.*

Lemma 5.11 (Δ^-, Δ^+) *satisfies the following properties*

1. $\Delta^-, \Delta^+ \in Radm_\rho$
2. $\Delta^- = \Psi(\Delta^+, \Delta^-)$
3. $\Delta^+ = \Psi(\Delta^-, \Delta^+)$
4. *for all* $(R^-, R^+) \in (Radm_\rho^{op} \times Radm_\rho)$, *if* $\Psi^{\S}(R^-, R^+) \sqsubseteq (R^-, R^+)$ *then* $R^- \subseteq \Delta^-$ *and* $R^+ \supseteq \Delta^+$
5. $\Delta^+ \subseteq \Delta^-$

Proof Items 1–3 are obvious. Item 4 follows by the least fixed point property. Item 5 follows by letting $R^- = \Delta^+$ and $R^+ = \Delta^-$ in 4. $\square$

To simplify notation, we write $e : R \subset R'$ for

$$\forall (e_1, \ldots, e_n) \in R : (e\, e_1, \ldots, e\, e_n) \in R'.$$

Note that this notation does not depend on the chosen equivalence class representative, so the notation is indeed well-defined.

Lemma 5.12 *For all* $i \in N$ *and for all* τ,

$$\Pi^i_\tau : [\![\tau]\!]'(\Delta^+, \Delta^-) \subset [\![\tau]\!]'(\Delta^-, \Delta^+)$$

Proof By induction on (i, τ) ordered lexicographically. We proceed by cases on τ.

Case $\tau = 0$: Follows immediately by $[\![0]\!]'(\Delta^+, \Delta^-) = [\![0]\!]'(\Delta^-, \Delta^+) = R_0$ and $\Pi^i_0 = \lambda x{:}0.x$, for all i, and Lemma 3.6.

Case $\tau = 1$: As the previous case.

Case $\tau = \rho$: Then $[\![\tau]\!]'(\Delta^+, \Delta^-) = \Delta^-$ and $[\![\tau]\!]'(\Delta^-, \Delta^+) = \Delta^+$. Assume $e_1, \ldots, e_n \in \Delta^-$. We are to show that $(\Pi^i_\rho\, e_1, \ldots, \Pi^i_\rho\, e_n) \in \Delta^+$. By admissibility of Δ^-, in particular by the strictness condition of admissibility, there are two cases to consider.

SubCase $e_k \Uparrow$, for all $1 \leq k \leq n$: Then also $\Pi^i_\rho\, e_k \Uparrow$, for all $1 \leq k \leq n$, so by admissibility of Δ^+, the required follows.

SubCase $e_k \Downarrow$, for all $1 \leq k \leq n$: Then, as $\Delta^- = \Psi(\Delta^+, \Delta^-)$, $(e_1, \ldots, e_n) = (\mathsf{in}\ v_1, \ldots, \mathsf{in}\ v_n)$ for some $(v_1, \ldots, v_n) \in [\![\tau_\rho]\!](\Delta^+, \Delta^-)$ (recall that we are working over equivalence classes). There are two subcases.

SubSubCase $i = 0$: Then $\Pi^i_\rho\, e_k \Uparrow$, for all $1 \leq k \leq n$, by Lemma 3.46, so by admissibility of Δ^+, the required follows.

SubSubCase $i > 0$: Then by Lemma 3.47 (aplicable as $i \geq 1$), $\vdash \Pi^i_\rho\, e \approx \mathsf{in}\,(\Pi^{i-1}_{\tau_\rho}\, v_k) : \rho$. By induction (note that $(i-1, \tau_\rho) < (i, \rho)$ in the lexicographic order), we get that $(\Pi^{i-1}_{\tau_\rho} v_1, \ldots, \Pi^{i-1}_{\tau_\rho} v_n) \in [\![\tau_\rho]\!](\Delta^-, \Delta^+)$. By admissibility there are two cases to consider.

SubSubSubCase $\Pi^{i-1}_{\tau_\rho}\, v_k \Uparrow$, for all $1 \leq k \leq n$: Then also $\Pi^i_\rho\, e_k \Uparrow$, for all $1 \leq k \leq n$, so by admissibility of $[\![\rho]\!](\Delta^-, \Delta^+)$, the required follows.

SubSubSubCase $\Pi^{i-1}_{\tau_\rho}\, v_k \Downarrow$, for all $1 \leq k \leq n$: Then $\Pi^{i-1}_{\tau_\rho}\, v_k = v'_k$, for all $1 \leq k \leq n$ such that $(v'_1, \ldots, v'_n) \in [\![\tau_\rho]\!](\Delta^-, \Delta^+)$, whence by Lemma 3.11, $\vdash \Pi^i_\rho\, e_k \approx \mathsf{in}\; v'_k : \rho$, for all $1 \leq k \leq n$, so by definition of Ψ, $(\Pi^i_\rho\, e_1, \ldots \Pi^i_\rho\, e_n) \in \Psi(\Delta^-, \Delta^+) = \Delta^+ = [\![\rho]\!](\Delta^+, \Delta^-)$, as required.

Case $\tau = \tau_1 \times \tau_2$: Then $[\![\tau]\!](\Delta^+, \Delta^-) = [\![\tau_1]\!](\Delta^+, \Delta^-) \times [\![\tau_2]\!](\Delta^+, \Delta^-)$ and $[\![\tau]\!](\Delta^-, \Delta^+) = [\![\tau_1]\!](\Delta^-, \Delta^+) \times [\![\tau_2]\!](\Delta^-, \Delta^+)$. Assume $(e_1, \ldots, e_n) \in [\![\tau]\!](\Delta^+, \Delta^-)$. We are to show that $(\Pi^i_\tau\, e_1, \ldots, \Pi^i_\tau\, e_n) \in [\![\tau]\!](\Delta^-, \Delta^+)$. By admissibility there are two cases to consider.

SubCase $e_k \Uparrow$, for all $1 \leq k \leq n$: Easy.

SubCase $e_k \Downarrow$, for all $1 \leq k \leq n$: Then by definition of $[\![\tau_1]\!](\Delta^+, \Delta^-) \times [\![\tau_2]\!](\Delta^+, \Delta^-)$, $e_k = (v'_k, v''_k)$, for all $1 \leq k \leq n$, $(v'_1, \ldots, v'_n) \in [\![\tau_1]\!](\Delta^+, \Delta^-)$, and $(v''_1, \ldots, v''_n) \in [\![\tau_2]\!](\Delta^+, \Delta^-)$. By Lemma 3.43, $\vdash \Pi^i_\tau\, e_k \approx (\Pi^i_{\tau_1}\, v'_k, ,)\Pi^i_{\tau_1}\, v''_k : \tau_1 \times \tau_2$, for all $1 \leq k \leq n$. By induction on (i, τ_1), $(v'_1, \ldots, v'_n) \in [\![\tau_1]\!](\Delta^-, \Delta^+)$. By induction on (i, τ_2), $(v''_1, \ldots, v''_n) \in [\![\tau_2]\!](\Delta^-, \Delta^+)$. By admissibility of $[\![\tau_1]\!](\Delta^-, \Delta^+)$ and $[\![\tau_2]\!](\Delta^-, \Delta^+)$, there are three subcases to consider.

SubSubCase $\Pi^i_{\tau_1}\, v'_k \Uparrow$, for all $1 \leq k \leq n$: Easy using Lemma 3.43.

SubSubCase $\Pi^i_{\tau_2}\, v''_k \Uparrow$, for all $1 \leq k \leq n$: Easy using Lemma 3.43.

SubSubCase $\vdash \Pi^i_{\tau_1}\, v'_k \approx v_{k'} : \tau_1$ for some $(v_{1'}, \ldots, v_{n'}) \in [\![\tau_1]\!](\Delta^-, \Delta^+)$ and $\vdash \Pi^i_{\tau_2}\, v''_k \approx v_{k''} : \tau_2$ for some $(v_{1''}, \ldots, v_{n''}) \in [\![\tau_2]\!](\Delta^-, \Delta^+)$: By Lemma 3.9, $\vdash \Pi^i_\tau\, e_k \approx (v_{k'}, v_{k''}) : \tau_1 \times \tau_2$, so by defintion of $[\![\tau_1]\!](\Delta^-, \Delta^+) \times [\![\tau_2]\!](\Delta^-, \Delta^+)$, the required follows.

Case $\tau = \tau_1 + \tau_2$: Similar to the case for $\tau = \tau_1 \times \tau_2$, using Lemmas 3.44 and 3.10.

Case $\tau = \tau_1 \rightharpoonup \tau_2$: Then $[\![\tau]\!](\Delta^+, \Delta^-) = [\![\tau_1]\!](\Delta^-, \Delta^+) \rightharpoonup [\![\tau_2]\!](\Delta^+, \Delta^-)$ and $[\![\tau]\!](\Delta^-, \Delta^+) = [\![\tau_1]\!](\Delta^+, \Delta^-) \rightharpoonup [\![\tau_2]\!](\Delta^-, \Delta^+)$. Assume $(e_1, \ldots, e_n) \in [\![\tau]\!](\Delta^+, \Delta^-)$. We are to show that $(\Pi^i_\tau\, e_1, \ldots, \Pi^i_\tau\, e_n) \in [\![\tau]\!](\Delta^-, \Delta^+)$. By admissibility there are two cases to consider.

SubCase $e_k \Uparrow$, for all $1 \leq k \leq n$: Easy.

SubCase $e_k \Downarrow$, for all $1 \leq k \leq n$: Then $(e_1, \ldots, e_n) = (v_1, \ldots, v_n)$ for some $(v_1, \ldots, v_n) \in [\![\tau]\!](\Delta^+, \Delta^-)$. By definition of $\rightharpoonup$ we thus have

$$(e'_1, \ldots, e'_n) \in [\![\tau_1]\!](\Delta^-, \Delta^+) \Rightarrow (v_1\, e'_1, \ldots, v_n\, e'_n) \in [\![\tau_2]\!](\Delta^+, \Delta^-) \qquad (56)$$

By Lemma 3.45, for all $1 \leq k \leq n$,

$$\vdash \Pi^i_\tau\, e_k \approx \lambda x{:}\tau_1.\Pi^i_{\tau_2}\,(v_k\,(\Pi^i_{\tau_1}\,x)) : \tau_1 \rightharpoonup \tau_2$$

Assume $(e'_1, \ldots, e'_n) \in [\![\tau_1]\!](\Delta^+, \Delta^-)$. By definition of $\rightharpoonup$ it then suffices to show that

$$(\lambda x{:}\tau_1.\Pi^i_{\tau_2}\,(v_1\,(\Pi^i_{\tau_1}\,x))\,(e'_1), \ldots, \lambda x{:}\tau_1.\Pi^i_{\tau_2}\,(v_n\,(\Pi^i_{\tau_1}\,x))\,(e'_n)) \in [\![\tau_2]\!](\Delta^-, \Delta^+)$$

By admissibility there are two subcases.

SubSubCase $e'_k \Uparrow$, for all $1 \leq k \leq n$: Easy.

SubSubCase $e'_k \Downarrow$, for all $1 \leq k \leq n$: Then $(e'_1, \ldots, e'_n) = (v'_1, \ldots, v'_n)$ for some $(v'_1, \ldots, v'_n) \in [\![\tau_1]\!](\Delta^+, \Delta^-)$. Then, for all $1 \leq k \leq n$,

$$\vdash \lambda x{:}\tau_1.\Pi^i_{\tau_2}\,(v_k\,(\Pi^i_{\tau_1}\,x))\,(e'_k) \approx \Pi^i_{\tau_2}\,(v_k\,(\Pi^i_{\tau_1}\,v'_k)) : \tau_2$$

By induction on (i, τ_1), $(\Pi^i_{\tau_1}\,v'_1, \ldots, \Pi^i_{\tau_1}\,v'_n) \in [\![\tau_1]\!](\Delta^-, \Delta^+)$. Hence, by (56),

$$(v_1\,(\Pi^i_{\tau_1}\,v'_1), \ldots, v_n\,(\Pi^i_{\tau_1}\,v'_n)) \in [\![\tau_2]\!](\Delta^+, \Delta^-)$$

By admissibility there are two cases to consider.

SubSubCase $v_k\,\Pi^i_{\tau_1}\,v'_k \Uparrow$, for all $1 \leq k \leq n$: Easy.

SubSubCase $v_k\,\Pi^i_{\tau_1}\,v'_k \Downarrow$, for all $1 \leq k \leq n$: Then $(v_1\,\Pi^i_{\tau_1}\,v'_1, \ldots, v_n\,\Pi^i_{\tau_1}\,v'_n) = (v''_1, \ldots, v''_n)$ for some $(v''_1, \ldots, v''_n) \in [\![\tau_2]\!](\Delta^+, \Delta^-)$. Then, for all $1 \leq k \leq n$, $\vdash \Pi^i_{\tau_2}\,(v_k\,(\Pi^i_{\tau_1}\,v'_k)) \approx \Pi^i_{\tau_2}\,v''_k : \tau_2$ and by induction on (i, τ_2), $(\Pi^i_{\tau_2}\,v''_1, \ldots, \Pi^i_{\tau_2}\,v''_n) \in [\![\tau_2]\!](\Delta^-, \Delta^+)$. Hence by admissibility of $[\![\tau_2]\!](\Delta^-, \Delta^+)$ and transitivity of $\approx$, the required follows. □

Lemma 5.13 *For all $i \in N$, $\pi^i : \Delta^- \subset \Delta^+$.*

Proof By induction on i.

Basis ($i = 0$): Assume $(e_1, \ldots, e_n) \in \Delta^-$. By Lemma 3.46 and since $\vdash \pi^0 \approx \Pi^0_\rho : \rho \rightharpoonup \rho$, $\pi^0\,e_k \Uparrow$, for all $1 \leq k \leq n$. Hence, by admissibility of Δ^+, $(\pi^0\,e_1, \ldots, \pi^0\,e_n) \in \Delta^+$, as required.

Inductive Step: We assume it holds for i and show for $i+1$. Assume $(e_1, \ldots, e_n) \in \Delta^-$. By admissibility of Δ^- there are two cases to consider.

SubCase $e_k \Uparrow$, for all $1 \leq k \leq n$: Easy.

SubCase $(e_1, \ldots, e_n) = (\mathsf{in}\,v_1, \ldots, \mathsf{in}\,v_n)$ for some $(v_1, \ldots, v_n) \in [\![\tau_\rho]\!](\Delta^+, \Delta^-)$: By Lemma 3.47 (applicable as $i + 1 \geq 1$), $\vdash \Pi^{i+1}_\rho\,e_k \approx \mathsf{in}\,(\Pi^i_{\tau_\rho}\,v_k) : \rho$, for all $1 \leq k \leq n$. By Lemma 5.12, $(\Pi^i_{\tau_\rho}\,v_1, \ldots, \Pi^i_{\tau_\rho}\,v_n) \in [\![\tau_\rho]\!](\Delta^-, \Delta^+)$. By admissibility of $[\![\tau_\rho]\!](\Delta^-, \Delta^+)$, there are two subcases to consider.

SubSubCase $\Pi^i_{\tau_\rho}\, v_k \Uparrow$, for all $1 \leq k \leq n$: Easy using Lemma 3.5.

SubSubCase $(\Pi^i_{\tau_\rho}\, v_1, \ldots, \Pi^i_{\tau_\rho}\, v_n) = (v'_1, \ldots, v'_n)$ for some $(v'_1, \ldots, v'_n) \in [\![\tau_\rho]\!](\Delta^-, \Delta^+)$: Then by transitivity and Lemma 3.7, $\vdash \Pi^{i+1}_\rho\, e_k \approx \mathsf{in}\; v'_k : \rho$, for all $1 \leq k \leq n$, so by definition of Ψ $(\Pi^{i+1}_\rho\, e_1, \ldots, \Pi^{i+1}_\rho\, e_n) \in \Psi(\Delta^-, \Delta^+) = \Delta^+$, as required. □

Lemma 5.14

$$\pi^\infty : \Delta^- \subset \Delta^+$$

Proof Let $(e_1, \ldots, e_n) \in \Delta^-$. We are to show that $(\pi^\infty\, e_1, \ldots, \pi^\infty\, e_n) \in \Delta^+$. By admissibility of Δ^+ (Lemma 5.11, item 1), with $I = N$ in the definition of admissibility, it suffices to show $\forall i \in N : (\pi^i\, e_1, \ldots, \pi^i\, e_n) \in \Delta^+$. But this follows from Lemma 5.13. □

Lemma 5.15

$$\Delta^- \subseteq \Delta^+$$

Proof By Lemma 5.14, Theorem 3.70 and the fact that admissible relations are over equivalence classes w.r.t. operational equivalence. □

Lemma 5.16

$$\Delta^- = \Delta^+$$

Proof By Lemmas 5.11 and 5.15. □

Definition 5.17

$$\Delta \stackrel{\text{def}}{=} \Delta^+$$

Definition 5.18 *For all* τ *define* $R_\tau \stackrel{\text{def}}{=} [\![\tau]\!]\Delta^+$.

This completes the construction of relations R_τ for all τ.

We now aim to show "The Fundamental Theorem of Logical Relations" which states that the relational interpretation of types is sound in the sense that well-typed terms are related to themselves by the relation associated to their type. To this end we first extend the interpretation of types as relations to type environments.

Definition 5.19 *For all type environments* Γ*,*

$$R_\Gamma \stackrel{\text{def}}{=} \{ (\gamma_1, \ldots, \gamma_n) \mid (\forall i \in 1..n : \gamma_i \textit{ is an expression substitution for } \Gamma) \wedge (\forall x \in \mathrm{Dom}(\Gamma) : (\gamma_1(x), \ldots, \gamma_n(x)) \in R_{\Gamma(x)}) \}$$

Theorem 5.20 *If* $\Gamma \vdash e : \tau$ *and* $(\gamma_1, \ldots, \gamma_n) \in R_\Gamma$*, then* $(\gamma_1(e), \ldots, \gamma_n(e)) \in R_\tau$*.*

Proof (Sketch) By induction on $\Gamma \vdash e : \tau$. In the case for T-FIX, by admissibility of the relations R_τ, for all τ, it suffices to show, for all $i \in N$,

$$(\mathsf{fix}\ f^i(x{:}\tau_1).\gamma_1(e), \ldots, \mathsf{fix}\ f^i(x{:}\tau_1).\gamma_n(e)) \in R_{\tau_1 \rightharpoonup \tau_2}$$

but this is easy to show by an inner induction on i using the outer induction hypothesis. $\square$

6 Logical Equivalence

In this section we shall be concerned with binary relations (i.e., $n = 2$) as constructed in the previous section. The relations can be used to define a notion of *logical equivalence* as follows.

Definition 6.1 (Logical Equivalence) *For all* $e, e' \in Exp_\tau$ *we define* $\vdash e\ R_\tau\ e'$ *if and only if* $(e, e') \in R_\tau$*.*

(Recall that e and e' denote the equivalence classes, wrt. operational equivalence, of e and e' respectively in the expression $(e, e') \in R_\tau$.)

Theorem 6.2 *If* $\vdash e \approx e' : \tau$ *then* $\vdash e\ R_\tau\ e'$*.*

Proof By Theorem 5.20. $\square$

Theorem 6.3 *If* $\vdash e\ R_\tau\ e'$ *then* $\vdash e \approx e' : \tau$*.*

Proof Suppose $\vdash e\ R_\tau\ e'$. Let $E\{\_{\tau}\} \in \mathrm{ECtx}_1$ be arbitrary. Further let $\Gamma = \{x \mapsto \tau\}$ and let $e_0 = E\{x\}$, $\gamma = \{x \mapsto e\}$, and $\gamma' = \{x \mapsto e'\}$. Then we have that $\Gamma \vdash e : 1$ and $(\gamma, \gamma') \in R_\Gamma$. Thus by Theorem 5.20, we get that $(\gamma(e_0), \gamma'(e_0)) \in R_1$. Thus $(E\{e\}, E\{e'\}) \in R_1$, so by definition of R_1, we have that $E\{e\} \approx^k E\{e'\}$. Hence as E was arbitrary, we have $\vdash e \approx e' : \tau$, as required. $\square$

Definition 6.4 (Open Logical Equivalence) *For all e and e', if $\Gamma \vdash e : \tau$ and $\Gamma \vdash e' : \tau$, then we define $\Gamma \vdash e\ R_\tau\ e'$ if and only if for all value substitutions γ and γ' for Γ satisfying $(\gamma, \gamma') \in R_\Gamma$, $\vdash \gamma(e)\ R_\tau\ \gamma'(e')$.*

Theorem 6.5 *$\Gamma \vdash e \approx e' : \tau$ if and only if $\Gamma \vdash e\ R_\tau\ e'$.*

Proof Suppose $\Gamma \vdash e\ R_\tau\ e'$ and let γ and γ' be value substitutions satisfying $(\gamma, \gamma') \in R_\Gamma$. Then $\forall x \in \mathrm{Dom}(\Gamma) :\ \vdash \gamma(x)\ R_{\Gamma(x)}\ \gamma'(x)$. Hence by Theorem 6.3, $\forall x \in \mathrm{Dom}(\Gamma) :\ \vdash \gamma(x) \approx \gamma'(x) : \Gamma(x)$. Thus from our assumption we get that $\vdash \gamma(e) \approx \gamma'(e') : \tau$ so by Theorem 6.2, $\vdash \gamma(e)\ R_\tau\ \gamma'(e')$, as required.

For the other direction, suppose that $\Gamma \vdash e\ R_\tau\ e'$. Let γ and γ' be value substitutions such that $\forall x \in \mathrm{Dom}(\Gamma) :\ \vdash \gamma(x) \approx \gamma'(x) : \Gamma(x)$. Then by Theorem 6.2, we have that $\forall x \in \mathrm{Dom}(\Gamma) :\ \vdash \gamma(x)\ R_{\Gamma(x)}\ \gamma'(x)$. Thus from our assumption we get that $\vdash \gamma(e)\ R_\tau\ \gamma'(e')$ so Theorem 6.3, $\vdash \gamma(e) \approx \gamma'(e') : \tau$, as required. $\square$

In summary, what we have so far is that contextual equivalence is equivalent to open experimental equivalence which is again equivalent by to open logical equivalence. In symbols

$$\begin{array}{rcll} \vdash e \approx^c e' : \tau & \iff & \Gamma \vdash e \approx e' : \tau & \text{Corollary 3.35} \\ & \iff & \Gamma \vdash e\ R_\tau\ e' & \text{Theorem 6.5.} \end{array}$$

Hence we may use logical equivalence to prove experimental and contextual equivalence. This is especially useful, as we shall now show, since we can derive a useful *coinduction principle* for establishing logical equivalence. One can also derive an *induction principle* but we shall not go into that here. These principle are derived in a manner analogously to the way in which Pitts [17] derives such principles. For reasons of space, we shall be less formal in our presentation of these reasoning principles than we are elsewhere.

Theorem 6.6 *For all $R^- \in Rel_\rho$ and for all $R^+ \in Radm_\rho$, the following inference rule is valid:*

$$\frac{\mathsf{out}\ : R^- \subset [\![\tau_\rho]\!]'(R^+, R^-) \qquad \mathsf{in}\ : [\![\tau_\rho]\!]'(R^-, R^+) \subset R^+}{R^- \subseteq \Delta \subseteq R^+}$$

Remark 6.7 *Note that R^- is* not *required to be admissible. (If R^- was required to be admissible then the theorem would essentially just be a restatement of Lemma 5.11, item 4.)*

Proof The idea of the proof is to show that, under the given assumptions, $\pi^\infty : R^- \subset \Delta$ and $\pi^\infty : \Delta \subset R^+$ and then use the syntactical minimal invariance to get the conclusion. Since Δ (as shown earlier) and R^+ (by assumption) are both admissible, we can show this by showing it for the finite unrollings of π^∞, as in the proof of Lemma 5.14. For the finite unrollings of π^∞, one proceeds as in the proofs of Lemmas 5.12 and 5.13. □

We now show how to specialize Theorem 6.6 to a coinduction principle and give some examples of how to use it. More examples of the kind found in [16] may also be treated this way.

Theorem 6.8 (Coinduction Principle) *For all $R \in Rel_\rho$, if* $\mathsf{in} : [\![\tau_\rho]\!]'(R, \Delta) \subset \Delta$*, then the following inference rule is valid:*

$$\frac{\mathsf{out}\ : R \subset [\![\tau_\rho]\!]'(\Delta, R)}{R \subseteq \Delta}$$

Remark 6.9 *Note that R is* not *required to be admissible.*

Proof By Theorem6.6, letting $R^- = R$ and $R^+ = \Delta$ and using that $[\![\tau_\rho]\!]'(R, \Delta) = \Delta$. □

Example For the purpose of this example, we shall assume that we have another ground type N and that $\tau_\rho = 1 + N \times \rho$, such that ρ is intuitively the type of lists of natural numbers. Moreover, assume R_N is the obvious equality relation on the type N (essentially defined analogously to R_1). Then $[\![\tau_\rho]\!]'(R, \Delta) = R_1 + R_N \times \Delta$, for any R, and thus, by definition of Δ, $\mathsf{in} : [\![\tau_\rho]\!]'(R, \Delta) = R_1 + R_N \times \Delta \subset \Delta$. Hence, for any $R \in Rel_\rho$, we have that the following inference rule is valid:

$$\frac{\mathsf{out}\ : R \subset R_1 + R_N \times R}{R \subseteq \Delta}$$

Unwinding the definitions, this rule says that *if*, whenever $e\ R\ e'$ then either

1. $\mathsf{out}\ e \Uparrow \wedge \mathsf{out}\ e' \Uparrow$; or
2. $\mathsf{out}\ e \mapsto^* \mathsf{inl}_{N\times\rho} * \wedge\ \mathsf{out}\ e' \mapsto^* \mathsf{inl}_{N\times\rho} *$; or
3. $\mathsf{out}\ e \mapsto^* \mathsf{inr}_1\ (n, v) \wedge \mathsf{out}\ e' \mapsto^* \mathsf{inr}_1\ (n, v') \wedge v\ R\ v'$;

then $e\ R\ e' \Rightarrow e\ \Delta\ e'$.

Let us now further assume that the list function *map* is defined as usual:

$$\begin{array}{lcl} map & = & \mathsf{fix}\ map(f{:}N \rightharpoonup N).\lambda x{:}\rho. \\ & & \quad \mathsf{case}(\mathsf{out}\ x, \lambda y{:}\tau_\rho.\mathsf{in}\ (\mathsf{inl}_{N\times\rho}\ *), \lambda y{:}\tau_\rho.\mathsf{in}\ (\mathsf{inr}_1\ (f\ (\mathsf{fst}\ y), map\ f\ (\mathsf{snd}\ y)))) \end{array}$$

and that *succ* is the successor function for the type of natural numbers and that $\circ$ is the functional-composition term. We want to show that $map\ succ\ (map\ succ\ e)$ is experimentally equivalent to $map\ (succ \circ succ)\ e$, for all $e : \rho$. By Theorem 6.3 it suffices to show that they are logically equivalent. To show that they are logically equivalent, we can apply our coinduction principle. To this end we let

$$R = \{(map\ succ\ (map\ succ\ e), map\ (succ \circ succ)e)\ \mid \vdash e : \rho\}.$$

One can now show that whenever $e\ R\ e'$, then the three items above are satisfied. Hence we can conclude that $e\ R\ e'$ implies that $e\ \Delta\ e'$ so recalling that $R_\rho = \Delta$, we have that $map\ succ\ (map\ succ\ e)$ is indeed logically equivalent to $map\ (succ \circ succ)e$, for all e such that $\vdash e : \rho$. $\square$

Example In this example, we shall again assume that we have a type of natural numbers N. We shall consider streams of natural numbers. Streams are implemented by means of functions, as is often the case in languages with call-by-value semantics. Thus we shall consider the case where $\tau_\rho = 1 \rightharpoonup N \times \rho$. Then one can show that $\mathsf{in}\ : [\![\tau_\rho]\!]'(R, \Delta) = R_1 \rightharpoonup R_N \times \Delta \subset \Delta$. Hence, for any $R \in Rel_\rho$, we have that the following inference rule is valid:

$$\frac{\mathsf{out}\ : R \subset 1 \rightharpoonup R_N \times R}{R \subseteq \Delta}$$

Unwinding the definitions, this rule says that *if*, whenever $e\ R\ e'$ then either

1. $\mathsf{out}\ e * \Uparrow \wedge \mathsf{out}\ e' * \Uparrow$; or

2. $\mathsf{out}\ e * \mapsto^* (n, v) \wedge \mathsf{out}\ e' * \mapsto^* (n, v') \wedge v\ R\ v'$;

then $e\ R\ e' \Rightarrow e\ \Delta\ e'$. Pitts [16] also derives a coinduction principle for infinite streams in his theory of program equivalence based on bisimulation. Pitts' coinduction principle corresponds closely to the one we have obtained here by specializing the recursive type to the type of streams.

Consider the following terms:

$$\begin{array}{lcl} ones & = & \mathsf{fix}\ ones(x{:}1).(1, \mathsf{in}\ (\lambda x{:}1.ones\ *)) \\ twos & = & \mathsf{fix}\ twos(x{:}1).(2, \mathsf{in}\ (\lambda x{:}1.twos\ *)) \\ succstr & = & \mathsf{fix}\ succstr(s{:}\rho).\lambda x{:}1.(\lambda p{:}N \times \rho.(succ\ \mathsf{fst}\ p, \mathsf{in}\ (succstr\ (\mathsf{snd}\ p))))\ (s\ *) \end{array}$$

Intuitively, *ones* is the streams of all ones, *twos* is the stream of all twos, and *succstr* is the successor operation on streams which applies the successor function to every element in the stream. Thus we would expect that *succstr ones* is operationally equivalent to *twos*. We can show this using coinduction, by considering the relation

$$R = \{(twos, succstr\ ones)\},$$

because supposing that $e\ R\ e'$, one can see that item 2 above is satisfied. Thus we conclude that $R \subseteq \Delta$ and thus that *succstr ones* is logically equivalent (and hence operationally equivalent) to *twos*. □

7 Correctness of CPS Transformation

We define the cps transformation as a relation between a "source" and a "target" language. The source language, $\mathcal{L}^{\rho}$, is just the language $\mathcal{L}$ defined earlier. The target language, $\mathcal{L}^{\rho^*}$, is the variant of $\mathcal{L}$ obtained by replacing the single recursive type ρ by another recursive type ρ^* obtained from ρ by a transformation on types similar to that given by Meyer and Wand [12].

We let Type^{ρ} denote the set of type expressions of $\mathcal{L}^{\rho}$, that is $\mathrm{Type}^{\rho} = \mathrm{Type}$. The set of target type expressions, denoted Type^{ρ^*}, is defined exactly as Type, but with ρ^* for ρ.

Below we define two type translations from Type^{ρ} to Type^{ρ^*}, one for computations, $\overline{\tau}$, and one for values, τ^* and extend the one for values to type environments. Note that the case $(\rho)^* = \rho^*$ is not recursive; it reads: "the value type translation of the source type ρ is the target type ρ^*."

Computations	$\overline{\tau}$	$=$	$(\tau^* \to 1) \to 1$
Values	0^*	$=$	0
	1^*	$=$	1
	$(\rho)^*$	$=$	ρ^*
	$(\tau_1 \times \tau_2)^*$	$=$	${\tau_1}^* \times {\tau_2}^*$
	$(\tau_1 + \tau_2)^*$	$=$	${\tau_1}^* + {\tau_2}^*$
	$(\tau_1 \rightharpoonup \tau_2)^*$	$=$	${\tau_1}^* \rightharpoonup \overline{\tau_2}$
Type Environments	$\Gamma^*(x)$	$=$	$(\Gamma(x))^* \quad (x \in \mathrm{Dom}(\Gamma))$

In the target language $\mathcal{L}^{\rho^*}$ we take the recursive type ρ^* to be isomorphic to ${\tau_\rho}^*$.

$$\Gamma \vdash x : \tau \rightsquigarrow_v x \quad (\Gamma(x) = \tau) \tag{CPS-VAR}$$

$$\Gamma \vdash * : 1 \rightsquigarrow_v * \tag{CPS-ONE}$$

$$\frac{\Gamma[f : \tau_1 \rightharpoonup \tau_2][x : \tau_1] \vdash e : \tau_2 \rightsquigarrow_c e'}{\Gamma \vdash \mathsf{fix}\, f(x{:}\tau_1).e : \tau_1 \rightharpoonup \tau_2 \rightsquigarrow_v \mathsf{fix}\, f(x{:}{\tau_1}^*).e'} \quad (f, x \notin \mathrm{Dom}(\Gamma)) \tag{CPS-FIX}$$

$$\frac{\Gamma \vdash v : \tau \rightsquigarrow_v v'}{\Gamma \vdash v : \tau \rightsquigarrow_c \lambda k{:}\tau^* \rightharpoonup 1.k\, v'} \tag{CPS-VAL}$$

Figure 3: CPS Transformation — Part I

We shall use the same notation for both the source and target language, but we must take care to remember to which language we are referring. Of course, all the results obtained in previous sections for $\mathcal{L}$ hold analogously for both the source and target language (for the source it is obvious as it is equal to $\mathcal{L}$, for the target, just replace ρ with ρ^* and τ_ρ with ${\tau_\rho}^*$ everywhere) and we will freely refer to these results to reason about both the source and the target language. When we need to distinguish between sets of expressions of the source and the target language, we shall use the notation developed for $\mathcal{L}$ but use a superscript ρ for the source language and a superscript ρ^* for the target language. For example, Exp_τ^ρ denotes the set of closed expressions of type τ of the source language, whereas $\mathrm{Exp}_\tau^{\rho^*}$ denotes the set of closed expressions of type τ of the target language. Moreover, we will abuse notation and write $e \approx^k e'$, for $e \in \mathrm{Exp}_1^\rho$ and $e' \in \mathrm{Exp}_1^{\rho^*}$, to mean that e evaluates to $*$ in $\mathcal{L}^\rho$ if and only if e' evaluates to $*$ in $\mathcal{L}^{\rho^*}$.

The translation relations $\Gamma \vdash v : \tau \rightsquigarrow_v v'$ for values and $\Gamma \vdash e : \tau \rightsquigarrow_c e'$ for computations are inductively defined by the rules in Figures 3 and 4.

Lemma 7.1

1. *$\Gamma \vdash e : \tau \rightsquigarrow_c e'$ for some e' iff $\Gamma \vdash e : \tau$.*
2. *If $\Gamma \vdash v : \tau \rightsquigarrow_v v'$, then $\Gamma^* \vdash v' : \tau^*$.*
3. *If $\Gamma \vdash e : \tau \rightsquigarrow_c e'$, then $\Gamma^* \vdash e' : \overline{\tau}$.*

We extend the notion of experimental equivalence to evaluation contexts as follows.

$$\frac{\Gamma \vdash e_1 : \tau_1 \rightsquigarrow_c e_1' \qquad \Gamma \vdash e_2 : \tau_2 \rightsquigarrow_c e_2'}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2 \rightsquigarrow_c \lambda k{:}(\tau_1 \times \tau_2)^* \rightharpoonup 1.e_1'\,(\lambda x_1{:}{\tau_1}^*.e_2'\,(\lambda x_2{:}{\tau_2}^*.k\,(x_1, x_2)))} \quad (\textsc{cps-prod})$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \rightsquigarrow_c e'}{\Gamma \vdash \mathsf{fst}\; e : \tau_1 \rightsquigarrow_c \lambda k{:}{\tau_1}^* \rightharpoonup 1.e'\,(\lambda x{:}{\tau_1}^* \times {\tau_2}^*.k\,(\mathsf{fst}\; x))} \quad (\textsc{cps-fst})$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \rightsquigarrow_c e'}{\Gamma \vdash \mathsf{snd}\; e : \tau_2 \rightsquigarrow_c \lambda k{:}{\tau_2}^* \rightharpoonup 1.e'\,(\lambda x{:}{\tau_1}^* \times {\tau_2}^*.k\,(\mathsf{snd}\; x))} \quad (\textsc{cps-snd})$$

$$\frac{\Gamma \vdash e : \tau_1 \rightsquigarrow_c e'}{\Gamma \vdash \mathsf{inl}_{\tau_2}\; e : \tau_1 + \tau_2 \rightsquigarrow_c \lambda k{:}(\tau_1 + \tau_2)^* \rightharpoonup 1.e'\,(\lambda x{:}{\tau_1}^*.k\,(\mathsf{inl}_{{\tau_2}^*}\; x))} \quad (\textsc{cps-inl})$$

$$\frac{\Gamma \vdash e : \tau_2 \rightsquigarrow_c e'}{\Gamma \vdash \mathsf{inr}_{\tau_1}\; e : \tau_1 + \tau_2 \rightsquigarrow_c \lambda k{:}(\tau_1 + \tau_2)^* \rightharpoonup 1.e'\,(\lambda x{:}{\tau_2}^*.k\,(\mathsf{inr}_{{\tau_1}^*}\; x))} \quad (\textsc{cps-inr})$$

$$\frac{\Gamma \vdash e_1 : \tau_1 + \tau_2 \rightsquigarrow_c e_1' \qquad \Gamma \vdash e_2 : \tau_1 \rightharpoonup \tau \rightsquigarrow_c e_2' \qquad \Gamma \vdash e_3 : \tau_2 \rightharpoonup \tau \rightsquigarrow_c e_3'}{\Gamma \vdash \mathsf{case}(e_1, e_2, e_3) : \tau \rightsquigarrow_c \lambda k{:}\tau^* \rightharpoonup 1.e_1'\,(\lambda x{:}{\tau_1}^* + {\tau_2}^*.\mathsf{case}(x, e_2'\,x\,k, e_3'\,x\,k))} \quad (\textsc{cps-case})$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \rightharpoonup \tau \rightsquigarrow_c e_1' \qquad \Gamma \vdash e_2 : \tau_2 \rightsquigarrow_c e_2'}{\Gamma \vdash e_1\, e_2 : \tau \rightsquigarrow_c \lambda k{:}\tau^* \rightharpoonup 1.e_1'\,(\lambda x_1{:}(\tau_2 \rightharpoonup \tau)^*.e_2'\,(\lambda x_2{:}{\tau_2}^*.x_1\, x_2\, k))} \quad (\textsc{cps-app})$$

$$\frac{\Gamma \vdash e : \rho \rightsquigarrow_c e'}{\Gamma \vdash \mathsf{out}\; e : \tau_\rho \rightsquigarrow_c \lambda k{:}{\tau_\rho}^* \rightharpoonup 1.e'\,(\lambda x{:}\rho^*.k\,(\mathsf{out}\; x))} \quad (\textsc{cps-out})$$

$$\frac{\Gamma \vdash e : \tau_\rho \rightsquigarrow_c e'}{\Gamma \vdash \mathsf{in}\; e : \rho \rightsquigarrow_c \lambda k{:}\rho^* \rightharpoonup 1.e'\,(\lambda x{:}{\tau_\rho}^*.k\,(\mathsf{in}\; x))} \quad (\textsc{cps-in})$$

Figure 4: CPS Transformation — Part II

Definition 7.2 *For all $E\{\_\tau\}, E'\{\_\tau\} \in ECtx_{\tau'}$, we define*

$$\vdash E\{\_\tau\} \approx E'\{\_\tau\} : \tau' \iff (\forall e, e' \in Exp_\tau : \vdash e \approx e' : \tau \Rightarrow \vdash E\{e\} \approx E'\{e'\} : \tau').$$

As in Section 4 we denote equivalence classes by one of their representatives.

Theorem 7.3 *There exists a* Type^ρ*-indexed family of relations*

$$\begin{array}{rcl} \Delta^c_\tau & \subseteq & Exp^\rho_\tau /\!\approx \times\, Exp^{\rho^*}_{\overline{\tau}} /\!\approx \\ \Delta^v_\tau & \subseteq & Val^\rho_\tau /\!\approx \times\, Val^{\rho^*}_{\tau^*} /\!\approx \\ \Delta^k_\tau & \subseteq & ECtx^\rho_\tau /\!\approx \times\, Val^{\rho^*}_{\tau^* \rightharpoonup 1} /\!\approx \end{array}$$

satisfying

$$\begin{array}{rcl} e\ \Delta^c_\tau\ e' & \iff & E\{\_\tau\}\ \Delta^k_\tau\ v' \Rightarrow E\{e\} \approx^k e'\, v' \\[1ex] v\ \Delta^v_1\ v' & \iff & v = *, v' = * \\ v\ \Delta^v_0\ v' & & \mathit{never} \\ v\ \Delta^v_\rho\ v' & \iff & \vdash v \approx \mathsf{in}\ v_1 : \rho,\ \vdash v' \approx \mathsf{in}\ v_1' : \rho^*,\ v_1\ \Delta^v_{\tau_\rho}\ v_1' \\ v\ \Delta^v_{\tau_1\times\tau_2}\ v' & \iff & \vdash v \approx (v_1, v_2) : \tau_1 \times \tau_2,\ \vdash v' \approx (v_1', v_2') : {\tau_1}^* \times {\tau_2}^*, \\ & & v_1\ \Delta^v_{\tau_1}\ v_1',\ v_2\ \Delta^v_{\tau_2}\ v_2' \\ v\ \Delta^v_{\tau_1+\tau_2}\ v' & \iff & \big(\vdash v \approx \mathsf{inl}_{\tau_2}\ v_1 : \tau_1 + \tau_2,\ \vdash v' \approx \mathsf{inl}_{\tau_2^*}\ v_1' : {\tau_1}^* + {\tau_2}^*,\ v_1\ \Delta^v_{\tau_1}\ v_1'\big) \\ & \vee & \big(\vdash v \approx \mathsf{inr}_{\tau_1}\ v_1 : \tau_1 + \tau_2,\ \vdash v' \approx \mathsf{inr}_{\tau_1^*}\ v_1' : {\tau_1}^* + {\tau_2}^*,\ v_1\ \Delta^v_{\tau_2}\ v_1'\big) \\ v\ \Delta^v_{\tau_1\rightharpoonup\tau_2}\ v' & \iff & v_1\ \Delta^v_{\tau_1}\ v_1' \Rightarrow v\, v_1\ \Delta^c_{\tau_2}\ v'\, v_1' \\[1ex] E\{\_\tau\}\ \Delta^k_\tau\ v' & \iff & v_1\ \Delta^v_\tau\ v_1' \Rightarrow E\{v_1\} \approx^k v'\, v_1', \end{array}$$

and

$$\big(\forall i \in N : \mathsf{fix}\ f^i(x{:}\tau_1).e\ \Delta^v_{\tau_1\rightharpoonup\tau_2}\ \mathsf{fix}\ f^i(x{:}{\tau_1}^*).e'\big) \Rightarrow \mathsf{fix}\ f(x{:}\tau_1).e\ \Delta^v_{\tau_1\rightharpoonup\tau_2}\ \mathsf{fix}\ f(x{:}{\tau_1}^*).e'.$$

(Note that the conditions satisfied by the relations are all independent of the choice of equivalence class representative and are thus well-defined conditions.)

The proof of this theorem will be postponed until Section 7.1. Now we shall first see how to use the relations that exists by the theorem to prove the correctness of the cps transformation.

Definition 7.4 *Let Δ^c_τ, Δ^v_τ, and Δ^k_τ be relations as in Theorem 7.3. We then define a source type environment indexed family of relations, Δ^v_Γ, relating source value substitutions for Γ modulo experimental equivalence*[1] *to target value substitutions for Γ^* modulo experimental equivalence as follows:*

$$\gamma \; \Delta^v_\Gamma \; \gamma' \iff \forall x \in \mathrm{Dom}(\Gamma) : \gamma(x) \; \Delta^v_{\Gamma(x)} \; \gamma'(x).$$

Theorem 7.5

1. *If $\Gamma \vdash v : \tau \leadsto_v v'$ and $\gamma \; \Delta^v_\Gamma \; \gamma'$, then $\gamma(v) \; \Delta^v_\tau \; \gamma'(v')$.*
2. *If $\Gamma \vdash e : \tau \leadsto_c e'$ and $\gamma \; \Delta^v_\Gamma \; \gamma'$, then $\gamma(e) \; \Delta^c_\tau \; \gamma'(e')$.*

Proof By simultaneous induction on $\Gamma \vdash v : \tau \leadsto_v v'$ and $\Gamma \vdash e : \tau \leadsto_c e'$. $\Box$

Corollary 7.6 (Correctness of cps transformation) *If $\vdash e : 1 \leadsto_c e'$, then $e' \approx^k e\,(\lambda x{:}1.x)$.*

7.1 Construction of Relations for CPS Correctness

In this section we prove Theorem 7.3. This amounts to constructing relations satisfying the conditions in Theorem 7.3. The idea is to proceed as in Sections 4 and 5 but, of course, with a different universe of relations and with different relational constructors.

We define a source type indexed family of universes of relations as follows.

Definition 7.7 *For all source types τ, we define a universe of relations*

$$Rel_\tau \stackrel{\mathrm{def}}{=} \mathcal{P}\left((Exp^\rho_\tau /\!\approx) \times (Exp^{\rho^*}_{\tau^*} /\!\approx) \right).$$

We use R to range over Rel_τ.

Notation 7.8 *When $I \in \mathcal{P}_{\mathrm{cof}}(N^{k+l})$ we write "$\vec{m}\vec{m}'$" for "$(i_1, \ldots, i_k, i_{k+1}, \ldots, i_{k+l}) \in I$ and $\vec{m} = (i_1, \ldots, i_k)$ and $\vec{m}' = (i_{k+1}, \ldots, i_{k+l})$."*

As in Section 4, we shall also use a notion admissibility.

Definition 7.9 *A relation $R \in Rel_\tau$ is* admissible *if and only if it satisfies both of the following conditions.*

[1] Recall the definition of experimental equivalence for substitutions, Definition 3.26.

Strictness: $(e, e') \in R$ *iff* $(e \Uparrow \wedge e' \Uparrow) \vee (\exists v, v' : e \mapsto^* v \wedge e' \mapsto^* v' \wedge (v, v) \in R)$.

Completeness: *For all* $C\{\vec{\mathsf{p}}\} \in Ctx^{\rho}_{\tau}$ *with all parameters in* $\vec{\mathsf{p}}$ *of type* $\tau_1 \rightharpoonup \tau_2$*, for all* $C'\{\vec{\mathsf{q}}\} \in Ctx^{\rho^*}_{\tau^*}$ *with all parameters in* $\vec{\mathsf{q}}$ *of type* $(\tau_1 \rightharpoonup \tau_2)^*$*, for all* $F_\omega = \mathsf{fix}\ f(x{:}\tau_1).e \in Exp^{\rho}_{\tau_1 \rightharpoonup \tau_2}$*, for all* $F'_\omega = \mathsf{fix}\ f(x{:}\tau_1{}^*).e' \in Exp^{\rho^*}_{(\tau_1 \rightharpoonup \tau_2)^*}$*,*

$$\left(\forall \vec{m}\vec{m}' \in I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{q}}|}) : (C\{F_{\vec{m}}\}, C'\{F'_{\vec{m}'}\}) \in R\right) \Rightarrow$$

$$\big((C\{F_\omega\}, C'\{F'_\omega\}) \in R)\big)\,.$$

Definition 7.10 *For all source types* τ*, we define a universe of admissible relations* $Radm_\tau$ *as follows.*

$$Radm_\tau \stackrel{\mathrm{def}}{=} \{\, R \in Rel_\tau \mid R \text{ is admissible} \,\}$$

We also use R *to range over* $Radm_\tau$*.*

We now define a series of relational type constructors, just as in Section 4. In each case, one has to check that the definitions we give are independent of the chosen equivalence class representative; this is straightforward in all cases (it is just like in Section 4).

Definition 7.11

$$R_0 \stackrel{\mathrm{def}}{=} \{\, (e, e') \in (Exp^{\rho}_0/{\approx}) \times (Exp^{\rho^*}_0/{\approx}) \mid e \Uparrow \wedge\, e' \Uparrow \}$$

Definition 7.12

$$R_1 \stackrel{\mathrm{def}}{=} \{\, (e, e') \in (Exp^{\rho}_1/{\approx}) \times (Exp^{\rho^*}_1/{\approx}) \mid (e \Uparrow \wedge e' \Uparrow) \vee (e \mapsto^* * \wedge\, e' \mapsto^* *) \,\}$$

Definition 7.13 *For all* $R_1 \in Rel_{\tau_1}$ *and* $R_2 \in Rel_{\tau_2}$*,*

$$\begin{array}{rcl} R_1 \times R_2 & \stackrel{\mathrm{def}}{=} & \{\, (e, e') \in (Exp^{\rho}_{\tau_1 \times \tau_2}/{\approx}) \times (Exp^{\rho^*}_{\tau_1 \times \tau_2}/{\approx}) \mid \\ & & \quad (e \Uparrow \wedge e' \Uparrow) \vee \\ & & \quad (\exists v_1, v_2, v'_1, v'_2 : \ \vdash e \approx (v_1, v_2) : \tau_1 \times \tau_2 \wedge \\ & & \quad\ \vdash e' \approx (v'_1, v'_2) : \tau_1{}^* \times \tau_2{}^* \wedge \\ & & \quad\ (v_1, v'_1) \in R_1 \wedge (v_2, v'_2) \in R_2) \,\} \end{array}$$

Definition 7.14 *For all* $R_1 \in Rel_{\tau_1}$ *and* $R_2 \in Rel_{\tau_2}$,

$$\begin{array}{rcl} R_1 + R_2 & \stackrel{\text{def}}{=} & \{\, (e, e') \in (Exp^{\rho}_{\tau_1+\tau_2} /\!\approx) \times (Exp^{\rho^*}_{\overline{\tau_1+\tau_2}} /\!\approx) \mid \\ & & (e \Uparrow \wedge e' \Uparrow) \vee \\ & & (\exists v, v' : \vdash e \approx \mathsf{inl}_{\tau_2}\ v : \tau_1 + \tau_2 \wedge \vdash e' \approx \mathsf{inl}_{\tau_2^*}\ v' : {\tau_1}^* + {\tau_2}^* \wedge \\ & & (v, v') \in R_1) \vee \\ & & (\exists v, v' : \vdash e \approx \mathsf{inr}_{\tau_1}\ v : \tau_1 + \tau_2 \wedge \vdash e' \approx \mathsf{inr}_{\tau_1^*}\ v' : {\tau_1}^* + {\tau_2}^* \wedge \\ & & (v, v') \in R_2) \,\} \end{array}$$

The following relational constructors will be used in the definition of the relational constructor for function types.

Definition 7.15 *For all* $R \in Rel_{\tau}$,

$$\begin{array}{rcl} \Delta^k_{\tau}\,(R) & \stackrel{\text{def}}{=} & \{\, (E\{_{-\tau}\}, v') \in (ECtx^{\rho}_{\tau} /\!\approx) \times (Val^{\rho^*}_{\tau^* \rightharpoonup 1} /\!\approx) \mid \\ & & \forall (e, e') \in R : E\{e\} \approx^k v'\, e' \,\} \end{array}$$

Definition 7.16 *For all* $R \in Rel_{\tau}$,

$$\begin{array}{rcl} \Delta^c_{\tau}\,(R) & \stackrel{\text{def}}{=} & \{\, (e, e') \in (Exp^{\rho}_{\tau} /\!\approx) \times (Exp^{\rho^*}_{\overline{\tau}} /\!\approx) \mid \\ & & (e \Uparrow \wedge e' \Uparrow) \vee \\ & & (\exists v, v' : \vdash e \approx v : \tau \wedge \vdash e' \approx v' : \overline{\tau} \wedge \\ & & \qquad \forall (E_0\{_{-\tau}\}, v'_0) \in \Delta^k_{\tau}\,(R) : E_0\{v\} \approx^k v'\, v'_0) \,\} \end{array}$$

Definition 7.17 *For all* $R_1 \in Rel_{\tau_1}$ *and* $R_2 \in Rel_{\tau_2}$,

$$\begin{array}{rcl} R_1 \rightharpoonup R_2 & \stackrel{\text{def}}{=} & \{\, (e, e') \in (Exp^{\rho}_{\tau_1 \rightharpoonup \tau_2} /\!\approx) \times (Exp^{\rho^*}_{{\tau_1}^* \rightharpoonup \overline{\tau_2}} /\!\approx) \mid \\ & & (e \Uparrow \wedge e' \Uparrow) \vee \\ & & (\exists v, v' : \vdash e \approx v : \tau_1 \rightharpoonup \tau_2 \wedge \vdash e' \approx v' : {\tau_1}^* \rightharpoonup \overline{\tau_2} \wedge \\ & & \qquad \forall (e_1, e'_1) \in R_1 : (v\, e_1, v'\, e'_1) \in \Delta^c_{\tau_2}\,(R_2) \,\} \end{array}$$

Note that $R_1 \rightharpoonup R_2$ is antimonotone in R_1 and monotone in R_2.

By proofs exactly analogous to the proofs in Section 4 of the corresponding results, one can now show that $(Radm_{\tau}, \subseteq)$ is a complete lattice, for all source types τ; a lemma corresponding to Lemma 4.11 holds; R_0 and R_1 are admissible; and $\times$ and $+$ both preserve admissibility. We now show that $\rightharpoonup$ preserve admissibility:

Lemma 7.18 *For all* $R_1 \in Rel_{\tau_1}$ *and all* $R_2 \in Radm_{\tau_2}$, $R_1 \rightharpoonup R_2 \in Radm_{\tau_1 \rightharpoonup \tau_2}$.

Proof The strictness condition is straightforward(as in the proof of Lemma 4.14). For completeness, assume

$$\forall \vec{m}\vec{m}' \in I \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{q}}|}) : (C\{F_{\vec{m}}\}, C'\{F'_{\vec{m}'}\}) \in R. \tag{57}$$

By the lemma corresponding to Lemma 4.11 there are two cases to consider.

Case I: $C\{F_\omega\} \Uparrow \wedge C'\{F'_\omega\} \Uparrow$ Easy.

Case II: $C\{F_\omega\} \mapsto^* v$ and $C'\{F'_\omega\} \mapsto^* v'$. By two applications of Lemma 3.22, there exist $V\{\vec{\mathsf{p}_1}\}$ and $V'\{\vec{\mathsf{q}_1}\}$ such that

$$\begin{array}{rclcrcl} v & = & V\{F_\omega\} & \quad & C\{\vec{\mathsf{p}}\} & \Downarrow^F & V\{\vec{\mathsf{p}_1}\} \\ v' & = & V'\{F'_\omega\} & \quad & C'\{\vec{\mathsf{q}}\} & \Downarrow^{F'} & V'\{\vec{\mathsf{q}_1}\} \end{array}$$

so

$$I'_1 \stackrel{\text{def}}{=} \{\, \vec{m}\vec{m}' \mid \vec{m}\vec{n} \in I \wedge C\{F_{\vec{m}}\} \mapsto^* V\{F_{\vec{m}'}\}\,\} \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}_1}|})$$

and

$$I'_2 \stackrel{\text{def}}{=} \{\, \vec{n}\vec{n}' \mid \vec{m}\vec{n} \in I \wedge C'\{F_{\vec{n}}\} \mapsto^* V'\{F'_{\vec{n}'}\}\,\} \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{q}}|+|\vec{\mathsf{q}_1}|}).$$

Thus

$$I'' \stackrel{\text{def}}{=} \{\, \vec{m}'\vec{n}' \mid \vec{m}\vec{m}' \in I'_1 \wedge \vec{n}\vec{n}' \in I'_2 \wedge \vec{m}\vec{n} \in I\,\}$$

is cofinal, i.e., $I'' \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}_1}|+|\vec{\mathsf{q}_1}|})$. By (57), Lemma 3.7, and definition of I'',

$$\forall \vec{m}'\vec{n}' \in I'' : (V\{F_{\vec{m}'}\}, V'\{F'_{\vec{n}'}\}) \in R_1 \rightharpoonup R_2.$$

Hence, by definition of $\rightharpoonup$,

$$\forall \vec{m}'\vec{n}' \in I'' : \forall (e, e') \in R_1 : (V\{F_{\vec{m}'}\}\, e, V'\{F'_{\vec{n}'}\}\, e') \in \Delta^c_{\tau_2}(R_2).$$

Let $\vec{m}'\vec{n}' \in I''$ and $(e, e') \in R_1$ be arbitrary. By definition of $\Delta^c_{\tau_2}(R_2)$ we then have that

$$\forall (E_0\{-_{\tau_2}\}, v'_0) \in \Delta^k_{\tau_2}(R_2) : E_0\{V\{F_{\vec{m}'}\}\, e\} \approx^k V'\{F'_{\vec{n}'}\}\, e'\, v'_0. \tag{58}$$

We are to show that

$$\forall (E_0\{-_{\tau_2}\}, v'_0) \in \Delta^k_{\tau_2}(R_2) : E_0\{V\{F_\omega\}\, e\} \approx^k V'\{F'_\omega\}\, e'\, v'_0. \tag{59}$$

Let $(E_0\{-_{\tau_2}\}, v'_0) \in \Delta^k_{\tau_2}(R_2)$ be arbitrary. Suppose $E_0\{V\{F_\omega\}\, e\} \mapsto^* *$. Let $C_{11}\{\vec{\mathsf{p}_1}\} = E_0\{V\{\vec{\mathsf{p}_1}\}\, e\}$. Then by Lemma 3.22,

$$C_{11}\{\vec{\mathsf{p}_1}\} \Downarrow^F *.$$

Hence
$$I_{11} \stackrel{\text{def}}{=} \{\, \vec{m}'\vec{n}' \mid \vec{m}'\vec{n}' \in I \wedge C_{11}\{F_{\vec{m}'}\} \mapsto^* * \,\}$$
is cofinal, thus non-empty. So there exists $\vec{m}'\vec{n}' \in I$ such that $C_{11}\{F_{\vec{m}'}\} \mapsto^* *$, i.e. $E_0\{V\{F_{\vec{m}'}\}\, e\} \mapsto^* *$. Hence, by (57), $V'\{F'_{\vec{n}'}\}\, e'\, v'_0 \mapsto^* *$, from which $V'\{F'_\omega\}\, e'\, v'_0 \mapsto^* *$ follows by Lemma 3.23. The other direction is similar, completing the proof of (59). Thus we conclude that $(C\{F_\omega\}, C'\{F'_\omega\}) \in R_1 \rightharpoonup R_2$, as required, since (e, e') and $(E_0\{-_{\tau_2}\}, v'_0)$ were arbitrary and using Lemma 3.7. □

For all source types $\tau \in \mathrm{Type}^\rho$ we define an interpretation $[\![\tau]\!]'$ exactly as in Definition 5.5.

Definition 7.19 *Define* $\Psi : (Radm_\rho^{op} \times Radm_\rho) \to Radm_\rho$ *by*

$$\begin{array}{rcl}\Psi(R^-, R^+) & \stackrel{\text{def}}{=} & \{\, (e, e') \in (Exp^\rho_\rho/\!\approx) \times (Exp^{\rho^*}_{\rho^*}/\!\approx) \mid \\ & & (e \Uparrow \wedge e' \Uparrow) \vee \\ & & (\exists v, v' : \;\vdash e \approx \mathsf{in}\; v : \rho \wedge \;\vdash e' \approx \mathsf{in}\; v' : \rho^* \wedge (v, v') \in [\![\tau_\rho]\!]'(R^-, R^+))\,\}\end{array}$$

Just like in Section 5 it is now easy to show that Ψ is well-defined.

We define $\Psi^\S : (Radm_\rho^{op} \times Radm_\rho) \to (Radm_\rho^{op} \times Radm_\rho)$ and as in Section 5 we get that $\Psi^\S$ is well-defined and monotone, so that we can define (Δ^-, Δ^+) as the least fixed point of $\Psi^\S$. Moreover, Lemma 5.11 holds also now.

We write $(e, e') : R \subset R'$ for $\forall (e_1, e'_1) \in R : (e\, e_1, e'\, e'_1) \in R'$.

Lemma 7.20 *For all* $i \in N$, *for all* $\tau \in \mathrm{Type}^\rho$,

$$(\Pi^{\rho,i}_\tau, \Pi^{\rho^*,i}_{\tau^*}) : [\![\tau]\!]'(\Delta^+, \Delta^-) \subset [\![\tau]\!]'(\Delta^-, \Delta^+).$$

Proof By induction on (i, τ), ordered lexicographically. All the cases are as in the proof of Lemma 5.12, except the case for $\tau = \tau_1 \rightharpoonup \tau_2$, which we now consider. Then

$$[\![\tau]\!]'(\Delta^+, \Delta^-) = [\![\tau_1]\!]'(\Delta^-, \Delta^+) \rightharpoonup [\![\tau_2]\!]'(\Delta^+, \Delta^-)$$

and

$$[\![\tau]\!]'(\Delta^-, \Delta^+) = [\![\tau_1]\!]'(\Delta^+, \Delta^-) \rightharpoonup [\![\tau_2]\!]'(\Delta^-, \Delta^+).$$

Assume

$$(e, e') \in [\![\tau]\!]'(\Delta^+, \Delta^-).$$

We are to show that

$$(\Pi^{\rho,i}_{\tau_1 \rightharpoonup \tau_2}\, e, \Pi^{\rho^*,i}_{\tau^* \rightharpoonup \overline{\tau_2}}\, e') \in [\![\tau]\!]'(\Delta^-, \Delta^+).$$

By admissibility there are two cases to consider.

SubCase $e \Uparrow \wedge e' \Uparrow$: Easy.

SubCase $\vdash e \approx v : \tau \wedge \vdash e' \approx v' : \tau^*$ for some $(v, v') \in [\![\tau]\!]'(\Delta^+, \Delta^-)$: By definition of $\rightharpoonup$, we thus have

$$(e_1, e_1') \in [\![\tau_1]\!]'(\Delta^-, \Delta^+) \Rightarrow (v\, e_1, v'\, e_1') \in \Delta^c_{\tau_2}\, ([\![\tau_2]\!]'(\Delta^+, \Delta^-)) \tag{60}$$

By two applications of Lemma 3.47 we get that

$$\vdash \Pi^{\rho,i}_{\tau_1 \rightharpoonup \tau_2}\, e \approx \lambda x{:}\tau_1.\Pi^{\rho,i}_{\tau_2}\, (v\, (\Pi^{\rho,i}_{\tau_1}\, x)) : \tau_1 \rightharpoonup \tau_2$$

and

$$\vdash \Pi^{\rho^*,i}_{{\tau_1}^* \rightharpoonup \overline{\tau_2}}\, e' \approx \lambda x{:}{\tau_1}^*.\Pi^{\rho^*,i}_{\overline{\tau_2}}\, (v'\, (\Pi^{\rho^*,i}_{{\tau_1}^*}\, x)) : {\tau_1}^* \rightharpoonup \overline{\tau_2}.$$

Assume

$$(e_1, e_1') \in [\![\tau_1]\!]'(\Delta^+, \Delta^-).$$

It then suffices to show that

$$\left(\lambda x{:}\tau_1.\Pi^{\rho,i}_{\tau_2}\, (v\, (\Pi^{\rho,i}_{\tau_1}\, x))\, e_1, \lambda x{:}{\tau_1}^*.\Pi^{\rho^*,i}_{\overline{\tau_2}}\, (v'\, (\Pi^{\rho^*,i}_{{\tau_1}^*}\, x))\, e_1'\right) \in \Delta^c_{\tau_2}\, ([\![\tau_2]\!]'(\Delta^-, \Delta^+)).$$

By admissibility there are two subcases to consider.

SubSubCase $e_1 \Uparrow \wedge e_1' \Uparrow$: Easy.

SubSubCase $\vdash e_1 \approx v_1 : \tau_1 \wedge \vdash e_1' \approx v_1' : {\tau_1}^*$ for some $(v_1, v_1') \in [\![\tau_1]\!]'(\Delta^+, \Delta^-)$: Then

$$\vdash \lambda x{:}\tau_1.\Pi^{\rho,i}_{\tau_2}\, (v\, (\Pi^{\rho,i}_{\tau_1}\, x))\, e_1 \approx \Pi^{\rho,i}_{\tau_2}\, (v\, (\Pi^{\rho,i}_{\tau_1}\, v_1)) : \tau_2$$

and

$$\vdash \lambda x{:}{\tau_1}^*.\Pi^{\rho^*,i}_{\overline{\tau_2}}\, (v'\, (\Pi^{\rho^*,i}_{{\tau_1}^*}\, x))\, e_1' \approx \Pi^{\rho^*,i}_{\overline{\tau_2}}\, (v'\, (\Pi^{\rho^*,i}_{{\tau_1}^*}\, v_1')) : \overline{\tau_2}.$$

By induction,

$$\left(\Pi^{\rho,i}_{\tau_1}\, v_1, \Pi^{\rho^*,i}_{{\tau_1}^*}\, v_1'\right) \in [\![\tau_1]\!]'(\Delta^-, \Delta^+).$$

Hence, by (60),

$$\left(v\, \Pi^{\rho,i}_{\tau_1}\, v_1, v'\, \Pi^{\rho^*,i}_{{\tau_1}^*}\, v_1'\right) \in \Delta^c_{\tau_2}\, ([\![\tau_2]\!]'(\Delta^+, \Delta^-)).$$

By admissibility there are two cases to consider.

SubSubSubCase $v\, \Pi^{\rho,i}_{\tau_1}\, v_1 \Uparrow \wedge v'\, \Pi^{\rho^*,i}_{{\tau_1}^*}\, v_1' \Uparrow$: Easy

SubSubSubCase $\vdash v\,\Pi^{\rho,i}_{\tau_1}\,v_1 \approx v_2 : \tau_2 \wedge \vdash v'\,\Pi^{\rho^*,i}_{\tau_1{}^*}\,v'_1 \approx v'_2 : \overline{\tau_2}$ for some $(v_2, v'_2) \in\!\Delta^c_{\tau_2}\ (\llbracket\tau_2\rrbracket'(\Delta^+, \Delta^-))$: Then it suffices to show that

$$(\Pi^{\rho,i}_{\tau_2}\,v_2, \Pi^{\rho^*,i}_{\overline{\tau_2}}\,v'_2) \in\!\Delta^c_{\tau_2}\ (\llbracket\tau_2\rrbracket'(\Delta^-, \Delta^+)).$$

To this end, assume

$$(E_{10}\{\_{\tau_2}\}, v_{10}) \in\!\Delta^k_{\tau_2}\ (\llbracket\tau_2\rrbracket'(\Delta^-, \Delta^+)). \tag{61}$$

We are to show that

$$E_{10}\{\Pi^{\rho,i}_{\tau_2}\,v_2\} \approx^k \Pi^{\rho^*,i}_{\overline{\tau_2}}\,v'_2\,v_{10}.$$

Since

$$\vdash \Pi^{\rho^*,i}_{\overline{\tau_2}}\,v'_2\,v_{10} \approx v'_2\,(\lambda x'{:}\tau_2{}^*.v_{10}\,(\Pi^{\rho^*,i}_{\tau_2{}^*}\,x')) : \overline{\tau_2}$$

it suffices to show

$$E_{10}\{\Pi^{\rho,i}_{\tau_2}\,v_2\} \approx^k v'_2\,(\lambda x'{:}\tau_2{}^*.v_{10}\,(\Pi^{\rho^*,i}_{\tau_2{}^*}\,x')).$$

Hence it suffices to show that

$$\left(E_{10}\{\Pi^{\rho,i}_{\tau_2}\,\_{\tau_2}\}, \lambda x'{:}\tau_2{}^*.v_{10}\,(\Pi^{\rho^*,i}_{\tau_2{}^*}\,x')\right) \in\!\Delta^k_{\tau_2}\ (\llbracket\tau_2\rrbracket'(\Delta^+, \Delta^-)).$$

(because then the above follows since $(v_2, v'_2) \in\!\Delta^c_{\tau_2}\ (\llbracket\tau_2\rrbracket'(\Delta^+, \Delta^-))$). To this end, assume

$$(e_{11}, e'_{11}) \in \llbracket\tau_2\rrbracket'(\Delta^+, \Delta^-). \tag{62}$$

We are to show that

$$E_{10}\{\Pi^{\rho,i}_{\tau_2}\,e_{11}\} \approx^k (\lambda x'{:}\tau_2{}^*.v_{10}\,(\Pi^{\rho^*,i}_{\tau_2{}^*}\,x'))\,e'_{11}.$$

Since

$$\vdash (\lambda x'{:}\tau_2{}^*.v_{10}\,(\Pi^{\rho^*,i}_{\tau_2{}^*}\,x'))\,e'_{11} \approx v_{10}\,(\Pi^{\rho^*,i}_{\tau_2{}^*}\,e'_{11}) : 1$$

it suffices to show

$$E_{10}\{\Pi^{\rho,i}_{\tau_2}\,e_{11}\} \approx^k v_{10}\,(\Pi^{\rho^*,i}_{\tau_2{}^*}\,e'_{11}). \tag{63}$$

But by induction on (62),

$$\left(\Pi^{\rho,i}_{\tau_2}\,e_{11}, \Pi^{\rho^*,i}_{\tau_2{}^*}\,e'_{11}\right) \in \llbracket\tau_2\rrbracket'(\Delta^-, \Delta^+),$$

so by assumption (61), the required (63) follows. $\square$

Lemma 7.21 *For all $i \in N$, $(\pi^{\rho,i}, \pi^{\rho^*,i}) : \Delta^- \subset \Delta^+$.*

Proof As the proof of Lemma 5.13. $\square$

Lemma 7.22

$$(\pi^{\rho,\infty}, \pi^{\rho^*,\infty}) : \Delta^- \subset \Delta^+$$

Proof As the proof of Lemma 5.14. $\square$

As in Section 5, it now follows that $\Delta^- = \Delta^+$ and we can define $\Delta \stackrel{\text{def}}{=} \Delta^+$.

Definition 7.23 *For all source types $\tau \in \text{Type}^\rho$, we define*

$$\Delta^e_\tau \stackrel{\text{def}}{=} [\![\tau]\!]'(\Delta, \Delta).$$

Definition 7.24 *For all source types $\tau \in \text{Type}^\rho$, we define*

$$\begin{array}{lcl}
\Delta^v_\tau & \stackrel{\text{def}}{=} & \{\, (e, e') \in \Delta^e_\tau \mid\; e \Downarrow \wedge e' \Downarrow \} \\
\Delta^k_\tau & \stackrel{\text{def}}{=} & \{\, (E\{_{\tau}\}, v') \in (ECtx^\rho_\tau/{\approx}) \times (Val^{\rho^*}_{\tau^* \rightharpoonup 1}/{\approx}) \mid (v_1, v_1') \in \Delta^v_\tau \Rightarrow E\{v\} \approx^k v'\, v \,\} \\
\Delta^c_\tau & \stackrel{\text{def}}{=} & \{\, (e, e') \in (Exp^\rho_\tau/{\approx}) \times (Exp^{\rho^*}_{\overline{\tau}}/{\approx}) \mid (E\{_{\tau}\}, v') \in \Delta^k_\tau \Rightarrow E\{e\} \approx^k e'\, v' \,\}
\end{array}$$

Lemma 7.25 *The above defined relations satisfy the conditions in Theorem 7.3.*

Proof All the conditions, except the one for $\Delta^v_{\tau_1 \rightharpoonup \tau_2}$ and the completeness condition, are obvious from the above definitions. By definition of $\Delta^v_{\tau_1 \rightharpoonup \tau_2}$, we have that

$$v \; \Delta^v_{\tau_1 \rightharpoonup \tau_2} \; v' \iff \left(e_1 \; \Delta^e_{\tau_1} \; e_1' \Rightarrow v\, e_1 \; \Delta^c_{\tau_2} \; v'\, e_1'\right),$$

but it is easy to check, using the definition of $\Delta^c_{\tau_2}$, that

$$(e_1 \; \Delta^e_{\tau_1} \; e_1' \Rightarrow v\, e_1 \; \Delta^c_{\tau_2} \; v'\, e_1') \iff (v_1 \; \Delta^v_{\tau_1} \; v_1' \Rightarrow v\, v_1 \; \Delta^c_{\tau_2} \; v'\, v_1')$$

which gives the required. The completeness condition for $\Delta^v_{\tau_1 \rightharpoonup \tau_2}$ follows by admissibility of $\Delta^e_{\tau_1 \rightharpoonup \tau_2}$ (using $I = \{\, (i, i) \mid i \in N \,\}$ a the cofinal set) and the facts that $\mathsf{fix}\, f(x{:}\tau_1).e \Downarrow$ and $\mathsf{fix}\, f(x{:}{\tau_1}^*).e' \Downarrow$. $\square$

This completes the construction of relations for CPS correctness.

8 Related Work

The construction of relations over recursive types hinges on a syntactic version of the minimal invariant property of the solution of a domain equation. The critical ingredient in the construction is Pitts's observation [17] that the existence of a relational interpretation can be reduced to minimal invariance, combined with the observation that this criterion can be stated and proved at a purely operational level. The proof of syntactic minimal invariance is a generalization of methods used by Mason, Smith, and Talcott [11] to a typed language with a recursive type. In addition to the applications given here this generalization sheds light on the need for "run-time type checks" in Mason, Smith, and Talcott's work — they arise here as compositions of recursive unrolling and case analysis on a disjoint union type, confirming Scott's observation that "untyped" really means "unityped".

The two applications of relational interpretations suggested here — analyzing contextual equivalence and proving correctness of the cps transformation — have been studied elsewhere using different methods. Pitts has emphasized the importance of a characterization of contextual equivalence for a language with streams as a bisimulation relation constructed as the maximal fixed point of a monotone operator on relations [16]. To apply this framework to specific examples Pitts relies on a lemma characterizing contextual equivalence of values of stream type. In our setting this lemma arises as a simple consequence of the definition of logical equivalence relation for a recursive type, as outlined in Section 6. Several authors have considered the correctness of the cps transformation. Reynolds [20] gives a proof for an untyped functional language by working over a domain model given by an inverse limit construction. Meyer & Wand [12] give a somewhat different proof for the simply typed λ-calculus (without a recursive type). The proof given in Section 7 generalizes both of these to a typed language with a recursive type without appealing to a denotational semantics.

9 Conclusion

We have presented a method for constructing relational interpretations of recursive types in an operational setting. The key result is the syntactic minimal invariant property up to a suitable notion of operational equivalence. With this in hand we may define relational interpretations of types over operational equivalence classes of closed terms. Using this construction we give a relational characterization of experimental and contextual equivalence and derive a coinduction principle for establishing contextual equivalence. Taking the recursive type to be the type of infinite streams, the coinduction principle specializes to a principle corresponding to the one used by Pitts [16] in his theory of program equivalence based on bisimulation. Using our construction we further give a relational proof of correctness of cps conversion, generalizing Reynolds' proof to the typed setting.

The proof of correctness for the cps transformation that we give here does not appear to extend easily to a language with control operators such as `call/cc` [1, 10]. The reason is that we rely on a "uniformity" property of the evaluation relation which states that evaluation steps are parametric in the evaluation context — if $E\{e\} \mapsto E\{e'\}$, then $E'\{e\} \mapsto E'\{e'\}$ — that fails in the presence of `call/cc`. It is also unclear whether our proof can be extended to a language with mutable storage. One possible approach may be to consider a store-passing transformation in which the store is represented by a value of a recursive type, and then to apply the methods considered here to complete the proof of correspondence between the original program and its cps transformation.

The treatment of cps conversion given here invites generalization to an arbitrary syntactically-definable monad for the language. Filinski's dissertation [3] is a first step towards a general theory of representation of computational effects. Filinski's work suggests that one could give a fairly general correctness proof along the lines suggested here for a wide variety of definable effects.

Acknowledgements

We are grateful to John Mitchell and Andy Pitts for many useful discussions and for their suggestions on this paper. We also thank Martín Abadi, Furio Honsell and Søren Lassen for their helpful comments. The work described here was carried out in part at the Isaac Newton Institute for Mathematical Sciences of Cambridge University in the autumn of 1995. We are grateful to the Newton Institute and the organizers of the program on Semantics of Computation for their support.

Robert Harper is supported by the National Science Foundation under Grant No. CCR-95-2674. Lars Birkedal is supported in part by the Danish National Research Council and in part by the National Science Foundation under Grant No. CCR-9409997.

References

[1] William Clinger and Jonathan Rees. Revised^4 report on the algorithmic language Scheme. *LISP Pointers*, IV(3):1–55, July-Sep. 1991.

[2] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990.

[3] Andrzej Filinski. *Controlling Effects*. CMU–CS–96–119, School of Computer Science, Carnegie Mellon University, May 1996.

[4] Michael J. Fischer. Lambda-calculus schemata. *LISP and Symbolic Computation*, 6(3/4):259–288, November 1993.

[5] Peter Freyd. Algebraically complete categories. In A. Carboni, M. C. Pedicchio, and G. Rosolini, editors, *Category Theory. Proceedings, Como 1990*, volume 1488 of *Lecture Notes in Mathematics*, pages 95–104. Springer-Verlag, 1990.

[6] Peter Freyd. Recursive types reduced to inductive types. In *Proceedings of the fifth IEEE Conference on Logic in Computer Science*, pages 498–507, 1990.

[7] Peter Freyd. Remarks on algebraically compact categories. In M. P. Fourman, P.T. Johnstone, and A. M. Pitts, editors, *Applications of Categories in Computer Science. Proceedings of the LMS Symposium, Durham 1991*, volume 177 of *London Mathematical Society Lecture Note Series*, pages 95–106. Cambridge University Press, 1991.

[8] Jean-Yves Girard. *Interprétation Fonctionnelle et Élimination des Coupures dans l'Arithmétique d'Ordre Supérieure*. PhD thesis, Université Paris VII, 1972.

[9] Carl A. Gunter. *Semantics of Programming Languages. Structures and Techniques*. MIT Press, 1992.

[10] Robert Harper, Bruce Duba, and David MacQueen. Typing first-class continuations in ML. *Journal of Functional Programming*, 3(4):465–484, October 1993.

[11] Ian A. Mason, Scott F. Smith, and Carolyn L. Talcott. From operational semantics to domain theory. *Information and Computation*, 1995. To Appear.

[12] Albert R. Meyer and Mitchell Wand. Continuation semantics in typed lambda calculi (summary). In Rohit Parikh, editor, *Logics of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 219–224. Springer-Verlag, 1985.

[13] Robin Milner, Mads Tofte, and Robert Harper. *The Definition of Standard ML*. MIT Press, 1990.

[14] John C. Mitchell. Type systems for programming languages. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B, Formal Models and Semantics*, chapter 8, pages 366–458. Elsevier Science Publishers B.V., 1990.

[15] John C. Mitchell. *Foundations for Programming Languages.* Foundations of Computing. MIT Press, 1996.

[16] Andrew M. Pitts. Operationally-based theories of program equivalence. In *In Proc. of Summer School on Semantics and Logics of Computation. ESPRIT CLiCS-II.* University of Cambridge. Isaac Newton Institute for Mathematical Sciences., September 1995.

[17] Andrew M. Pitts. Relational properties of domains. *Information and Computation*, 127(2):66–90, June 1996.

[18] Gordon Plotkin. Call-by-name, call-by-value, and the lambda calculus. *Theoretical Computer Science*, 1:125–159, 1975.

[19] Gordon Plotkin. Domains. Department of Computer Science. University of Edinburgh, 1983.

[20] John C. Reynolds. On the relation between direct and continuation semantics. In J. Loeckx, editor, *Proceedings of the Second Colloquium on Automata, Languages and Programming, Saarbrücken*, volume 174 of *Lecture Notes in Computer Science*, pages 141–156. Springer-Verlag, 1974.