Relational Interpretations of Recursive Types in an Operational Setting[*]

Lars Birkedal and Robert Harper
birkedal@cs.cmu.edu and rwh@cs.cmu.edu
School of Computer Science
Carnegie Mellon University

December 29, 1998

[*]A summary of some of the results in this paper appeared in TACS '97.

Proposed running head:
Recursive Types in an Operational Setting

Mailing address of corresponding author:
Lars Birkedal
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA

Abstract

Relational interpretations of type systems are useful for establishing properties of programming languages. For languages with recursive types it is difficult to establish the existence of a relational interpretation. The usual approach is to pass to a domain-theoretic model of the language, and exploiting the structure of the model to derive relations properties of it. 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 prove *syntactic minimal invariance* for an extension of PCF with a recursive type, a syntactic analogue of the minimal invariance property used by Freyd and Pitts to characterize the domain interpretation of a recursive type. As Pitts has shown in the setting of domains, syntactic minimal invariance suffices to establish the existence of relational interpretations. 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 experimental 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.

1 Introduction

It is an important problem to develop techniques for building compilers that help to ensure that the generated code behaves as expected. A natural approach is to view compilation as a form of program transformation between a source and a target language, each with a well-defined syntax and semantics. The problem is then to prove that the source and target program have the same observable behavior at execution time.

Most studies of compiler transformations focus on type-free languages, for which types play no role at run time. Compiler transformations are given as syntax-directed translations between untyped intermediate languages. Recent work, however, stresses the use of types during compilation and at run time to enhance the reliability of the compiler itself and to improve the quality of generated code (Shao *et al.* , 1998; Tarditi *et al.* , 1996; Morrisett, 1995). To take advantage of types, compiler transformations are generalized to type-directed translations between typed intermediate languages. Type-directed translations are guided by, and preserve, the type of the program to ensure that types are propagated from one stage to the next.

We will consider two forms of type-based compiler transformation. *Local*, or *peephole*, transformations are those that replace one code fragment by another that is *contextually equivalent* to it, which means that which means that the second exhibits the same behavior as the first in all contexts. Determining whether two fragments are contextually equivalent is difficult since it requires consideration of all possible contexts in which these fragments might be used. *Global* transformations are those that operate on complete programs, rather than program fragments. An example of a global transformation is the *continuation-passing (cps) transformation*, which makes explicit the control state of a program (Fischer, 1993; Plotkin, 1975). The correctness of this transformation states that a complete program and its cps transformation yield the same observable behavior; when viewed as a transformation on program fragments, it does not preserve contextual equivalence.

We will use the method of *logical relations* (Statman, 1985) in our study of compiler transformations for typed languages. Roughly speaking, logical relations are a means of specifying an invariant relation between two programs that ensures that they engender the same observable behavior when used in a complete program, even though they may differ substantially on intermediate results. This is achieved by associating with each type a relation that is preserved (in a sense to be made precise below) by the primitive operations of that type. We will demonstrate the use of logical relations to characterize experimental equivalence and to give a proof of correctness of the cps transformation. As we will see, the key to applying the method of logical relations is to establish the existence of a family of relations satisfying the required properties.

It is straightforward to construct logical relations for languages with simple type systems, including those with product, sum, and function types, because in these systems we may make use of definition by induction on the structure of type expressions. However, practical programming languages (such as Standard ML (Milner *et al.* , 1997)) have richer type systems for which the it is more difficult to establish the existence of relational interpretations. As a case in point we will consider a call-by-value variant of Plotkin's PCF (Plotkin, 1977) extended with a single (unrestricted) recursive type. For such a language the conditions required of a logical relation are "circular", precluding their definition by induction on types. (Other language features, such as impredicative polymorphism (Girard, 1972; Reynolds, 1974b), or computational effects, such as mutable references, present further difficulties for the relational approach. We do not consider these complications here.)

The usual method for handling recursive types is to pass to a denotational semantics of the language (Plotkin, 1983). In such a semantics the recursive type is interpreted as the inverse limit of a system of domains (Scott, 1982). Relations over the domain model may be constructed by an analogous inverse limit construction (see, for example, Reynolds's proof of correctness for the continuation transform (Reynolds, 1974a)). A weakness of this approach is that it is still necessary to prove that the denotational semantics is *computationally adequate* (Plotkin, 1983) in order to transfer properties of the model to properties of the execution behavior of the program. But the usual proof of adequacy relies on a logical relations argument, raising a further question of existence of a relational interpretation of types (Plotkin, 1983; Pitts, 1996)!

A natural question is whether it is possible to avoid the passage to a domain model, instead working entirely with the operational semantics of the language. We answer this in the affirmative by transferring key properties of the domain interpretation into the operational setting. Specifically, we exploit recent results of Pitts on relational properties of domains (Pitts, 1996) and the methods of Mason, Smith, and Talcott for deriving equivalences of expressions (Mason *et al.* , 1995). Pitts demonstrated that relational properties of a domain model of a recursive type can be obtained using only a universal property of the model, called *minimal invariance*, that states that a recursive function canonically associated with the recursive type is the identity function on that type. Mason, Smith, and Talcott developed methods for establishing equivalences of expressions in an untyped language that we adapt and extend to the case of a typed language with a single recursive type. This extension sheds light on the need for "run-time type checks" in their formalism; here they arise naturally from the isomorphism between a recursive type and its unrolling, and the primitive case analysis construct of disjoint union types.

The starting point for our work is the observation that the minimal invariance condition isolated by Pitts is expressible entirely in the syntax of the language itself. More precisely, the canonical recursive function associated with the recursive type is definable as an expression of PCF extended with that recursive type. Adapting techniques introduced by Mason, Smith, and Talcott (Mason *et al.* , 1995), we prove that this function is operationally equivalent to the identity, a property that we call *syntactic minimal invariance.* Following Pitts, we then show that syntactic minimal invariance is sufficient for the construction of logical relations, which we use to characterize experimental equivalence and to prove correct the continuation transformation. The characterization of experimental equivalence provides induction and co-induction methods for proving equivalence of expression of recursive type. The proof of correctness for the cps transformation extends Reynolds's proof (Reynolds, 1974a) to a typed language with an arbitrary recursive type, and avoids the passage to a denotational semantics.

The remainder of this paper is organized as follows. In Section 2 we introduce the language $\mathcal{L}$, a call-by-value variant of Plotkin's PCF enriched with a single recursive type. 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 (Mason *et al.* , 1995). In Section 4 we define a universe of admissible relations over experimental 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 to define the relational interpretation of types. In Section 6 we use this construction to characterize experimental equivalence. In Section 7 we use a similar construction to give a proof of correctness of the cps transformation. Finally, in Section 8 we discuss related work, and in Section 9 we conclude and suggest directions for further research.

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:

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

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 $f \mathbin{\backslash\!\backslash} A$ 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 $1 \leq i \leq n$.

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 $(\mathsf{fix}\ f(x{:}1){:}0.f\,x)\ *$.

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

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

Further define

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

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

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

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

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

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

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

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

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

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

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

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

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

Figure 1: Typing Rules

Likewise, we define sets for values as follows.

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

and

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

Substitution of an expression e' for free occurrences of x in e is written $[e'/x]e$. The parallel substitution 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. Suppose $\Gamma[x : \tau] \vdash e : \tau'$. We then write $\lambda x{:}\tau.e$ as an abbreviation for $\mathsf{fix}\ f(x{:}\tau){:}\tau'.e$, where f is some variable satisfying $f \notin \mathrm{FV}(e)$; when there is no risk of confusion we shall use this abbreviation without explicating the context Γ.

It can easily be shown that the following "strengthening lemma" holds for typing: if $\Gamma[x : \tau] \vdash e : \tau'$ and $x \notin \mathrm{FV}(e)$, then $\Gamma \vdash e : \tau'$. Also, the usual substitution lemma holds: if $\Gamma[x : \tau] \vdash e' : \tau'$ and $\Gamma \vdash e : \tau$, then $\Gamma \vdash [e/x]e' : \tau'$.

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

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

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. $[C/\mathsf{p}_\tau]C'$ denotes the context obtained from context C' by replacing all occurrences of p_τ in C' with C. This may involve capture of variables.

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

Proof By induction on C'. □

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 $\_{\tau}$. We write $C\{\_{\tau}\}$ to indicate that C is a context containing no parameters other than $\_{\tau}$ (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

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 $\_\tau$). 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

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

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. □

Redices are generated by the following grammar.

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

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

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

and

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

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

Proof By induction on e. □

The reduction rules for redices are as follows.

$$\begin{array}{lcll} (\mathsf{fix}\ f(x{:}\tau){:}\tau'.e)\ v & \rightsquigarrow & [\mathsf{fix}\ f(x{:}\tau){:}\tau'.e, v/f, x]e & (\text{R-BETA}) \\ \mathsf{fst}\ (v_1, v_2) & \rightsquigarrow & v_1 & (\text{R-FST}) \\ \mathsf{snd}\ (v_1, v_2) & \rightsquigarrow & v_2 & (\text{R-SND}) \\ \mathsf{out}\ (\mathsf{in}\ v) & \rightsquigarrow & v & (\text{R-OUT}) \\ \mathsf{case}(\mathsf{inl}_\tau\ v, e_1, e_2) & \rightsquigarrow & e_1\, v & (\text{R-CASE-INL}) \\ \mathsf{case}(\mathsf{inr}_\tau\ v, e_1, e_2) & \rightsquigarrow & e_2\, v & (\text{R-CASE-INR}) \end{array}$$

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 $r \rightsquigarrow e_1$.

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. □

Lemma 2.7 *1. For all τ and all $v \in \mathit{Val}_\tau$: $v \Downarrow$.*

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. □

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 $v = \mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.e$ 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 $\vdash e : \tau$. □

Corollary 2.13 *If $\vdash e : \tau$, then either $e \Uparrow$ or $e \Downarrow$.*

Lemma 2.14 (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 2.15 *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 2.16 *If $e \in Exp_{\tau}$ and $e \Uparrow$, then $\forall E\{\_{\tau}\} \in ECtx_{\tau'} : E\{e\} \Uparrow$.*

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. Then the usual list function *map* can be defined as follows:

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

where *succ* is the successor function for the type of natural numbers and $\circ$ is the functional-composition term. □

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 state 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

1. *Experimental equivalence is closed under evaluation contexts, i.e., if $\vdash e \approx e' : \tau$ and $E\{\_{\tau}\} \in ECtx_{\tau'}$, then $\vdash E\{e\} \approx E\{e'\} : \tau'$.*
2. *Experimental equivalence on closed values is closed under arbitrary contexts, i.e., if $\vdash v \approx v' : \tau$ and $C\{\_{\tau}\} \in Ctx_{\tau'}$, then $\vdash C\{v\} \approx C\{v'\} : \tau'$.*

Lemma 3.9 *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 3.10

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 3.11

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 3.12

1. *If* $\vdash e \approx \mathsf{in}\ e' : \rho$*, then* $e \Downarrow$ *iff* $e' \Downarrow$*.*

2. *If* $\vdash e \approx \mathsf{in}\ e' : \rho$*, then* $e' \Uparrow$ *iff* $e \Uparrow$*.*

3. $\vdash \mathsf{in}\ e \approx \mathsf{in}\ e' : \rho$ *iff* $\vdash e \approx e' : \tau_\rho$*.*

Using the above lemmas, it is easy to show the following corollary. The corollary expresses that to show two values of type $\tau_1 \times \tau_2$, $\tau_1 + \tau_2$, or ρ experimentally equivalent we do not have to consider all possible evaluation contexts (as in the definition of experimental equivalence); a more restricted set suffices.

Corollary 3.13

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

$$\forall E\{\_{\tau_1}\} \in ECtx_1 : E\{\mathsf{fst}\ v\} \preceq^k E\{\mathsf{fst}\ v'\}$$

and

$$\forall E\{\_{\tau_2}\} \in ECtx_1 : E\{\mathsf{snd}\ v\} \preceq^k E\{\mathsf{snd}\ v'\}$$

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

$$\forall \tau \in \mathrm{Type} : \forall E\{\_{\tau}\} \in ECtx_1 : \forall e_1 \in Exp_{\tau_1 \rightharpoonup \tau} : \forall e_2 \in Exp_{\tau_2 \rightharpoonup \tau} : \\ E\{\mathsf{case}(v, e_1, e_2)\} \preceq^k E\{\mathsf{case}(v', e_1, e_2)\}$$

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

$$\forall E\{\_{\tau_\rho}\} \in ECtx_1 : E\{\mathsf{out}\ v\} \preceq^k E\{\mathsf{out}\ v'\}$$

We now embark on showing that also for function types $\tau_1 \rightharpoonup \tau_2$ it suffices to consider a restricted set of evaluation contexts. To this end, we first prove the following lemma.

Lemma 3.14 *For all $v, v' \in Val_{\tau_1 \rightharpoonup \tau_2}$, if for all $E\{\_{\tau_2}\} \in ECtx_1$ and for all $v_1 \in Val_{\tau_1}$, $E\{v\, v_1\} \preceq^k E\{v'\, v_1\}$, then for all $\tau' \in \mathrm{Type}$, for all $z \in \mathrm{Var}$, for all $e \in Exp_{\tau'}(z : \tau_1 \rightharpoonup \tau_2)$, for all $E\{\_^{\prime}_{\tau}\} \in ECtx_1(z : \tau_1 \rightharpoonup \tau_2)$,*

$$([v/z](E\{e\}) \mapsto^n *) \Rightarrow \big([v'/z](E\{e\}) \mapsto^* *\big)\,.$$

Proof By induction on n.

Basis $(n = 0)$*:* Then $\tau' = 1$, $E = \_{1}$, and $e = *$ and then also $[v'/z](E\{e\}) = *$ and hence $[v'/z](E\{e\}) \mapsto^* *$, as required.

Inductive Step: We assume that the lemma holds for $n \geq 0$ and show for $n+1$. 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. $E\{e\} = E'\{z\, v_1\}$ for some $E'\{\_{\tau_2}\} \in \mathrm{ECtx}_1([z : \tau_1 \rightharpoonup \tau_2])$ and some $v_1 \in \mathrm{Val}_{\tau_1}([z : \tau_1 \rightharpoonup \tau_2])$
2. $E\{e\} = E'\{\mathsf{fix}\ f(x{:}\tau_1'){:}\tau_2'.e_0\ v_1'\}$ for some $E'\{\_{\tau_2'}\} \in \mathrm{ECtx}_1([z : \tau_1 \rightharpoonup \tau_2])$, some $v_1' \in \mathrm{Val}_{\tau_1'}([z : \tau_1 \rightharpoonup \tau_2])$, and some $\mathsf{fix}\ f(x{:}\tau_1'){:}\tau_2'.e_0 \in \mathrm{Val}_{\tau_1' \rightharpoonup \tau_2'}([z : \tau_1 \rightharpoonup \tau_2])$

SubCase 1: Then v is of the form $\mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.e_0$. Thus

$$\begin{array}{rcl} [v/z](E\{e\}) & = & [v/z](E'\{z\, v_1\}) \\ & = & [v/z](E'\{(\mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.e_0)\, v_1\}) \\ & \mapsto & [v/z](E'\{[v, v_1/f, x]e_0\}) \\ & \mapsto^n & * \end{array}$$

Thus we can apply induction to get that

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

from which the required easily follows, since

$$[v'/z](E\{e\}) \mapsto [v'/z](E'\{[v, v_1/f, x]e_0\}).$$

SubCase 2: Then

$$\begin{array}{rcl}
[v/z](E\{e\}) & = & [v/z](E'\{\mathsf{fix}\, f(x{:}\tau_1'){:}\tau_2'.e_0\; v_1'\}) \\
 & \mapsto & [v/z](E'\{[\mathsf{fix}\, f(x{:}\tau_1'){:}\tau_2'.e_0, v_1'/f, x]e_0\}) \\
 & \mapsto^n & *
\end{array}$$

Now use induction and proceed as in the previous case.

Case R-OUT, R-CASE-INL, R-CASE-INR, R-FST, or R-SND: The proof for each of these cases are all easy applications of the inductive hypothesis. □

Corollary 3.15 *To show* $\vdash v \preceq v' : \tau_1 \rightharpoonup \tau_2$ *it suffices to show*

$$\forall E\{_{-\tau_2}\} \in ECtx_1 : \forall v_1 \in Val_{\tau_1} : E\{v\, v_1\} \preceq^k E\{v'\, v_1\}$$

Proof Let $E\{_{-\tau_1 \rightharpoonup \tau_2}\}$ be arbitrary and suppose $E\{v\} \mapsto^n *$. Let $e = z$, $\tau' = \tau_1 \rightharpoonup \tau_2$, $E\{_{-\tau'}\} = E\{_{-\tau_1 \rightharpoonup \tau_2}\}$ in the previous lemma and conclude that $[v'/z](E\{z\}) \mapsto^* *$. But $E\{v'\} = [v'/z](E\{z\})$, so we have the required. □

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 continuous 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 (Pitts, 1995, 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 $F = \mathsf{fix}\, f(x{:}\tau_1){:}\tau_2.e$ satisfying $F \in \mathrm{Exp}_{\tau_1 \rightharpoonup \tau_2}$, and use the following abbreviations:

$$\begin{array}{rcl}
F_0 & \stackrel{\text{def}}{=} & \mathsf{fix}\, f^0(x{:}\tau_1){:}\tau_2.e \stackrel{\text{def}}{=} \mathsf{fix}\, f(x{:}\tau_1){:}\tau_2.f\, x \\
F_{n+1} & \stackrel{\text{def}}{=} & \mathsf{fix}\, f^{n+1}(x{:}\tau_1){:}\tau_2.e \stackrel{\text{def}}{=} \lambda x{:}\tau_1.[F_n/f]e \\
F_\omega & \stackrel{\text{def}}{=} & F
\end{array}$$

Note that the F_i's are just abbreviations of expressions already in the language. Another approach is to introduce new labelled expressions and new notions of reduction for labelled expressions as, e.g., done by Gunter (Gunter, 1992).

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 3.16 *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 3.17 *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

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

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

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

Remark 3.18 *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 3.19 *We abbreviate $V\{F_{\vec{m}}\}$ and $V\{F_{\vec{\omega}}\}$ analogously to $C\{F_{\vec{m}}\}$ and $C\{F_{\vec{\omega}}\}$.*

Definition 3.20 *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 $I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|})$*

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

The intuition of this definition is the following: Suppose you have m unrollings of F to compute with, that is, if you try to make more than m recursive calls of F, then you will diverge. Now, if m unrollings are enough to result in a value in which there are m' unrollings of F left, then, if you have $k \geq m$ unrollings and you perform the same computation, you will end up with a value with more than m' unrollings left.

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 3.21 *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. □

Lemma 3.22

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 3.21,

$$C\{\vec{\mathsf{p}}\vec{\mathsf{p}}'\} \Downarrow^F V''\{\vec{\mathsf{p}}''\} \qquad (1)$$

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 (1) is a cofinal subset of $N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|+|\vec{\mathsf{p}}''|}$, as required. $\square$

Lemma 3.23 (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 3.22, 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 3.22, 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 3.22.

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

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

$$\begin{array}{rcl} C\{F_{\vec{\omega}}\} & = & E\{(\mathsf{fix}\,f(x{:}\tau_1){:}\tau_2.e)\,v'\} \\ & \mapsto & E\{[\mathsf{fix}\,f(x{:}\tau_1){:}\tau_2.e, v'/f, x]e\} \\ & \mapsto^n & v \end{array}$$

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

$$E\{[\mathsf{fix}\, f(x{:}\tau_1){:}\tau_2.e, v'/f, x]e\} = E'\{[p, V'/f, x]e\}\{F_{\vec{\omega}}\}$$

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

$$C'\{\vec{\mathsf{p}}p\} \Downarrow^F V\{\vec{\mathsf{p}}'\} \tag{2}$$

We aim to show that

$$C\{\vec{\mathsf{p}}\} \Downarrow^F V\{\vec{\mathsf{p}}'\} \tag{3}$$

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 (2) 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 (3) as desired.

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

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

and $E\{[\mathsf{fix}\, f'(x'{:}\tau'){:}\tau''.e', v'/f, x]e'\}$ 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

$$C_1\{\vec{\mathsf{p}}\vec{\mathsf{p}_1}\} \Downarrow^F V\{\vec{\mathsf{p}}'\} \tag{4}$$

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 (4),

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

is cofinal and thus it is easy to see that also

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

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 3.24 *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. □

We now generalize the experimental pre-order to open expressions in the following way.

Definition 3.25 *An* expression substitution γ for a type environment Γ *is a finite map from variables to closed expressions satisfying the following two conditions.*

1. $\text{Dom}(\gamma) = \text{Dom}(\Gamma)$.

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

Definition 3.26 *A* value substitution γ for Γ *is an expression substitution for Γ satisfying $\forall x \in \text{Dom}(\Gamma) : \gamma(x) \Downarrow$.*

Definition 3.27 *Let γ and γ' be expression substitutions for Γ. Then γ approximates γ', written $\vdash \gamma \preceq \gamma' : \Gamma$, if and only if $\forall x \in \text{Dom}(\Gamma) : \vdash \gamma(x) \preceq \gamma'(x) : \Gamma(x)$. Likewise, we write $\vdash \gamma \approx \gamma' : \Gamma$, if and only if $\forall x \in \text{Dom}(\Gamma) : \vdash \gamma(x) \approx \gamma'(x) : \Gamma(x)$.*

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

Definition 3.28 (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 $\Gamma \vdash e' \preceq e : \tau$.*

Lemma 3.29 *If* $[f : \tau_1 \rightharpoonup \tau_2, x : \tau_1] \vdash e \preceq e' : \tau_2$ *then* $\vdash \mathsf{fix}\, f(x{:}\tau_1){:}\tau_2.e \preceq \mathsf{fix}\, f(x{:}\tau_1){:}\tau_2.e' : \tau_1 \rightharpoonup \tau_2$.

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

$$\vdash \mathsf{fix}\, f^i(x{:}\tau_1){:}\tau_2.e \preceq \mathsf{fix}\, f^i(x{:}\tau_1){:}\tau_2.e' : \tau_1 \rightharpoonup \tau_2 \qquad (5)$$

By Corollary 3.15, it suffices to show,

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

So assume, $E\{(\mathsf{fix}\, f(x{:}\tau_1){:}\tau_2.e)\,(v_1)\} \mapsto^* *$. Let $C\{\_{\tau_1 \rightharpoonup \tau_2}\} = E\{\_{\tau_1 \rightharpoonup \tau_2}\, v_1\}$. Then, by Lemma 3.23, there exists a $V\{\vec{\mathsf{p}}'\}$ such that $V\{F_{\vec{\omega}}\} = *$ and

$$C\{\vec{\mathsf{p}}\} \Downarrow^F V\{\vec{\mathsf{p}}'\} \qquad (6)$$

Let $I = N$, clearly a cofinal set. Then by (6),

$$I' \overset{\text{def}}{=} \{\, i \in I \mid C\{\mathsf{fix}\, f^i(x{:}\tau_1){:}\tau_2.e\} \mapsto^* * \,\}$$

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 $C\{\mathsf{fix}\, f^i(x{:}\tau_1){:}\tau_2.e\} \mapsto^* *$. Thus, by definition of I and C, there exists an $i \in N$ such that $E\{(\mathsf{fix}\, f^i(x{:}\tau_1){:}\tau_2.e)\,(v_1)\} \mapsto^* *$. Hence, by (5), we also have $E\{(\mathsf{fix}\, f^i(x{:}\tau_1){:}\tau_2.e')\,(v_1)\} \mapsto^* *$. Then by Lemma 3.24, we get $E\{(\mathsf{fix}\, f(x{:}\tau_1){:}\tau_2.e')\,(v_1)\} \mapsto^* *$, as required. □

Lemma 3.30 *If* $\Gamma, \Gamma' \vdash e \preceq e' : \tau$, $C\{\_\tau\} \in \mathit{Ctx}_{\tau'}(\Gamma)$, $\Gamma \vdash C\{e\} : \tau'$, *and* $\Gamma \vdash C\{e'\} : \tau'$, *then* $\Gamma, \Gamma' \vdash C\{e\} \preceq C\{e'\} : \tau'$.

Proof By induction on C. In the case for $C = \mathsf{fix}\ f(x{:}\tau){:}\tau'.C'$, use Lemma 3.29; all the other cases follow easily (either directly by the assumptions or by induction and using Lemmas 3.7–3.12 and composition of evaluation contexts). □

An alternative definition of open experimental approximation would be to say that $\Gamma \vdash e \preceq e' : \tau$ if and only if, for all value substitutions γ for Γ, $\vdash \gamma(e) \preceq \gamma(e') : \tau$. This notion is referred to as CIU experimental approximation and was used by Mason, Talcott and Smith (Mason *et al.*, 1995). We write $\Gamma \vdash e \preceq^{\mathrm{CIU}} e' : \tau$ when e CIU approximates e' in context Γ. In fact, CIU approximation agrees with open experimental approximation:

Lemma 3.31 $\Gamma \vdash e \preceq e' : \tau \iff \Gamma \vdash e \preceq^{CIU} e' : \tau$.

Proof The left-to-right implication is obvious. For the other direction, suppose $\Gamma = [x_1 : \tau_1, \ldots, x_n : \tau_n]$ and that γ and γ' are value substitutions satisfying $\vdash \gamma \preceq \gamma' : \Gamma$. Then by Lemma 3.7,

$$\vdash \gamma(e) \preceq (\lambda x_1{:}\tau_1.. \cdots \lambda x_n{:}\tau_n.e)\, \gamma(x_1) \cdots \gamma(x_n) : \tau$$

and thus, since each $\gamma(x_i)$ occurs in an evaluation context and $\vdash \gamma \preceq \gamma' : \Gamma$,

$$\vdash \gamma(e) \preceq (\lambda x_1{:}\tau_1.. \cdots \lambda x_n{:}\tau_n.e)\, \gamma'(x_1) \cdots \gamma'(x_n) : \tau$$

By Lemma 3.7 again and transitivity, we get

$$\vdash \gamma(e) \preceq \gamma'(e) : \tau$$

so by the assumption $\Gamma \vdash e \preceq^{\mathrm{CIU}} e' : \tau$ and transitivity,

$$\vdash \gamma(e) \preceq \gamma'(e') : \tau$$

as required. □

Remark 3.32 *One can now show, either directly or using the above lemma as in (Mason* et al. *, 1995), that open experimental equivalence coincides with the usual notion of* contextual equivalence*. We shall not give the details here, since they are fairly standard and since they are not used in the remainder of the paper. The point of this fact, however, is that we can use experimental equivalence to reason about "peephole" optimizations, as argued in the introduction.*

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 3.33 (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 3.30. □

Lemma 3.34 *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 3.33 and transitivity of $\preceq$. $\square$

Corollary 3.35 (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 3.34. $\square$

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

Proof
1 $\Rightarrow$ 3: Let $I \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}|})$ be arbitrary. The required follows by Lemmas 3.34 and 3.24 and transitivity. (as in 1 $\Rightarrow$ 2 above).
3 $\Rightarrow$ 2: Obvious.
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 3.23, there exists a $V\{\vec{\mathsf{p}}'\}$ such that $V\{F_{\vec{\omega}}\} = *$ and

$$E\{C\}\{\vec{\mathsf{p}}\} \Downarrow^F V\{\vec{\mathsf{p}}'\} \tag{7}$$

Clearly, $V = *$. By the assumption that $I \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}|})$ and (7) we get that

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

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. $\square$

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 $C\{\vec{\mathsf{p}}\} = \__{\tau_1 \to \tau_2}$ in Corollary 3.37. Then the corollary together with Lemma 3.24 intuitively says that F_ω is the least upper bound of the chain of its finite unrollings: By Lemma 3.24,

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

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 3.33,

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

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 (Mason *et al.* , 1995) 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, (Mason *et al.* , 1995, 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 (Mason *et al.* , 1995).

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.

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

Note that π is possibly free in these so defined terms. Further, define

$$\pi_{\omega} \overset{\text{def}}{=} \mathsf{fix}\ \pi(x{:}\rho){:}\rho.\mathsf{in}\ (\Pi_{\tau_\rho}\, (\mathsf{out}\ x)) : \rho \rightharpoonup \rho$$

and define π_i ($i \geq 0$) to be the finite unrollings of π_ω, as in the previous section. Observe that π_i and also π_ω are values.

The π_ω term corresponds to the least fixed point *fix*(δ) of the continuous function $\delta(e) = iF(e,e)i^{-1}$ in (Pitts, 1996, Definition 3.2). We shall show that π_ω is experimentally equivalent to the identity function (more precisely, the term $\lambda x{:}\rho.x$); this corresponds to the minimal invariant property in (Pitts, 1996, Definition 3.2).

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

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

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

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

$$\Pi_i^{\tau} \overset{\text{def}}{=} [\pi_i/\pi]\Pi^{\tau} : \tau \rightharpoonup \tau$$

Finally, for all τ, we define

$$\Pi_\omega^\tau \overset{\text{def}}{=} [\pi_\omega/\pi]\Pi^\tau : \tau \rightharpoonup \tau$$

It is easy to show that the above definitions do indeed define terms, i.e., for all τ, $[\pi : \rho \rightharpoonup \rho] \vdash \Pi^\tau : \tau \rightharpoonup \tau$ and $\vdash \Pi_\omega^\tau : \tau \rightharpoonup \tau$; $\vdash \pi_\omega : \rho \rightharpoonup \rho$; and for all τ, for all $i \geq 0$, $\vdash \Pi_i^\tau : \tau \rightharpoonup \tau$; and for all $i \geq 0$, $\vdash \pi_i : \rho \rightharpoonup \rho$.

We aim to show that π_ω 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 3.38 *If* $\vdash e \approx * : 1$ *then*

1. *For all* $i \geq 0$, $\vdash \Pi_i^1\, e \approx * : 1$.
2. $\vdash \Pi_\omega^1\, e \approx * : 1$

Lemma 3.39 *If* $\vdash e \approx (v_1, v_2) : \tau_1 \times \tau_2$, *then*

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

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 3.10 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 3.10, $\vdash \Pi_i^{\tau_1 \times \tau_2}\, (e) \approx (\Pi_i^{\tau_1}\, v_1, \Pi_i^{\tau_2}\, v_2) : \tau_1 \times \tau_2$, as required. $\square$

Lemma 3.40

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) $\vdash \Pi_\omega^{\tau_1+\tau_2}\, e \approx \mathsf{inl}_{\tau_2}\,(\Pi_\omega^{\tau_1}\, v) : \tau_1 + \tau_2$.

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) $\vdash \Pi_\omega^{\tau_1+\tau_2}\, e \approx \mathsf{inr}_{\tau_1}\,(\Pi_\omega^{\tau_2}\, v) : \tau_1 + \tau_2$.

Lemma 3.41 *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. $\vdash \Pi_\omega^{\tau_1 \rightharpoonup \tau_2}\, e \approx \lambda x{:}\tau_1.\Pi_\omega^{\tau_2}\,(v\,(\Pi_\omega^{\tau_1}\, x)) : \tau_1 \rightharpoonup \tau_2$.

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 3.9, 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

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

But by Lemma 3.8, we have

$$\vdash \lambda x{:}\tau_1.\Pi_i^{\tau_2}\,(v\,(\Pi_i^{\tau_1}\,(x))) \approx \lambda x{:}\tau_1.\Pi_i^{\tau_2}\,(v'\,(\Pi_i^{\tau_1}\,(x))) : \tau_1 \rightharpoonup \tau_2$$

from which the required follows by transitivity. $\Box$

Lemma 3.42 *If* $\vdash e \approx \mathsf{in}\, v : \rho$, *then* $\Pi_0^{\rho}\, e \Uparrow$.

Lemma 3.43 *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. $\vdash \Pi_\omega^{\rho}\, e \approx \mathsf{in}\,(\Pi_\omega^{\tau_\rho}\, v) : \rho$.

Lemma 3.44 *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. $\vdash \pi_\omega\, e \approx \mathsf{in}\,(\Pi_\omega^{\tau_\rho}\, v) : \rho$.

Lemma 3.45 *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 3.15 it suffices to show, for all τ, for all $v \in \mathrm{Val}_\tau$, for all $E\{\_{\tau \rightharpoonup \tau}\, v\}$

$$E\{\Pi_i^\tau\, v\} \preceq^k E\{v\} \tag{8}$$

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

Case $\tau = 1$: Follows by Lemma 3.38.

Case $\tau = 0$: Vacuously 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 3.42.

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 3.43 (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 3.39 and induction on (i, τ_1) and (i, τ_2).

Case $\tau = \tau_1 + \tau_2$: Follows by Lemma 3.40 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 3.41 and Corollary 3.15, induction on (i, τ_1) and induction on (i, τ_2). $\square$

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

Lemma 3.46 $\vdash \pi_\omega \preceq \lambda x{:}\rho.x : \rho \rightharpoonup \rho$

Proof By Corollary 3.37, it suffices to show

$$\forall i \in N : \quad \vdash \pi_i \preceq \lambda x{:}\rho.x : \rho \rightharpoonup \rho \tag{9}$$

We show this by induction on i.

Basis $(i = 0)$*:* By Lemma 3.6, Corollary 3.13 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 $\pi_0 = \mathsf{fix}\ \pi(x{:}\rho){:}\rho.\pi\ x$ the required follows immediately.

Inductive Step: We assume (9) holds for i and show for $i+1$. By Lemma 3.6, Corollary 3.13 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

$$E\{\pi_{i+1}\,(\mathsf{in}\ v)\} \mapsto^* * \tag{10}$$

Then by Lemma 3.43 (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

$$E\{\mathsf{in}\ (\Pi_i^{\tau_\rho}\, v)\} \mapsto^* * \tag{11}$$

Then by Lemma 3.45, also $E\{\mathsf{in}\ v\} \mapsto^* *$, as required. □

Next we aim to show the other half of the operational equivalence of π_ω and the identity function, that is, that the identity function operationally approximates π_ω. We shall employ an idea of Mason, Smith, and Talcott (Mason *et al.* , 1995).

We now proceed to show idempotency of Π_ω^τ and π_ω. 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 3.47 *For all $i \geq 0$ and for all τ, $\vdash \Pi_i^\tau \preceq \lambda x{:}\tau.\Pi_\omega^\tau\,(\Pi_\omega^\tau\, x) : \tau \rightharpoonup \tau$.*

Proof By Corollary 3.15 it suffices to show, for all $i \geq 0$, for all $v \in \mathrm{Val}_\tau$, and for all $E\{\_{\tau \rightharpoonup \tau}\, v\} \in \mathrm{ECtx}_1$,

$$E\{\Pi_i^\tau\, v\} \preceq^k E\{(\lambda x{:}\tau.\Pi_\omega^\tau\,(\Pi_\omega^\tau\, x))\, v\}$$

This can shown by induction on (i, τ) ordered lexicographically. □

Lemma 3.48 *For all $i \geq 0$, $\vdash \pi_i \preceq \lambda x{:}\rho.\pi_\omega\,(\pi_\omega\, x) : \rho \rightharpoonup \rho$.*

Proof Follows by Lemma 3.47. □

Lemma 3.49 *For all $i \geq 0$ and for all τ, $\vdash \lambda x{:}\tau.\Pi_i^\tau\,(\Pi_i^\tau\, x) \preceq \Pi_\omega^\tau : \tau \rightharpoonup \tau$.*

Proof By Corollary 3.15 it suffices to show, for all $i \geq 0$, for all $v \in \mathrm{Val}_\tau$, and for all $E\{\_{\tau \rightharpoonup \tau}\, v\} \in \mathrm{ECtx}_1$,

$$E\{(\lambda x{:}\tau.\Pi_i^\tau\,(\Pi_i^\tau\, x))\, v\} \preceq^k E\{\Pi_\omega^\tau\, v\}$$

This can shown by induction on (i, τ) ordered lexicographically. □

Lemma 3.50 *For all $i \geq 0$, $\vdash \lambda x{:}\rho.\pi_i\,(\pi_i\,x) \preceq \pi_\omega : \rho \rightharpoonup \rho$.*

Proof Follows by Lemma 3.49. □

Lemma 3.51 *For all τ, $\vdash \Pi^\tau_\omega \preceq \lambda x{:}\tau.\Pi^\tau_\omega\,(\Pi^\tau_\omega\,x) : \tau \rightharpoonup \tau$.*

Proof By Corollary 3.37, with $C = {\_}_{\tau \rightharpoonup \tau}$, and Lemma 3.47. □

Lemma 3.52 $\vdash \pi_\omega \preceq \lambda x{:}\rho.\pi_\omega\,(\pi_\omega\,x) : \rho \rightharpoonup \rho$.

Proof By Corollary 3.37, with $C = {\_}_{\rho \rightharpoonup \rho}$, and Lemma 3.48. □

Lemma 3.53 *For all τ, $\vdash \lambda x{:}\tau.\Pi^\tau_\omega\,(\Pi^\tau_\omega\,x) \preceq \Pi^\tau_\omega : \tau \rightharpoonup \tau$.*

Proof By Corollary 3.37, with $C = \lambda x{:}\rho.{\_}_1\,({\_}_2\,x)$ with ${\_}_1$ and ${\_}_2$ of type $\tau \rightharpoonup \tau$, and Lemma 3.49. □

Lemma 3.54 $\vdash \lambda x{:}\rho.\pi_\omega\,(\pi_\omega\,x) \preceq \pi_\omega : \rho \rightharpoonup \rho$.

Proof By Corollary 3.37, with $C = \lambda x{:}\rho.{\_}_1\,({\_}_2\,x)$ with ${\_}_1$ and ${\_}_2$ of type $\rho \rightharpoonup \rho$, and Lemma 3.50. □

Corollary 3.55 *For all $e \in Exp_\tau$ and for all $E\{{\_}_\tau\} \in ECtx_{\tau'}$,*

$$\vdash E\{\Pi^\tau_\omega\,(\Pi^\tau_\omega\,e)\} \approx E\{\Pi^\tau_\omega\,e\} : \tau'.$$

Proof Follows by Lemmas 3.51 and 3.53. □

Corollary 3.56 *For all $e \in Exp_\rho$ and for all $E\{{\_}_\rho\} \in ECtx_\tau$,*

$$\vdash E\{\pi_\omega\,(\pi_\omega\,e)\} \approx E\{\pi_\omega\,e\} : \tau.$$

Proof Follows by Lemmas 3.52 and 3.54. □

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 unique $|e|$ (i.e., the compilation relation is a function).

$$\Gamma \vdash x : \tau \Rightarrow \Pi^{\tau}_{\omega}\, x \quad (\Gamma(x) = \tau) \qquad (\textsc{tr-var})$$

$$\Gamma \vdash * : 1 \Rightarrow \Pi^{1}_{\omega}\, * \qquad (\textsc{tr-one})$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \Rightarrow |e_1| \qquad \Gamma \vdash e_2 : \tau_2 \Rightarrow |e_2|}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2 \Rightarrow \Pi^{\tau_1 \times \tau_2}_{\omega}\, (|e_1|, |e_2|)} \qquad (\textsc{tr-prod})$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \Rightarrow |e|}{\Gamma \vdash \mathsf{fst}\; e : \tau_1 \Rightarrow \mathsf{fst}\; |e|} \qquad (\textsc{tr-fst})$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2 \Rightarrow |e|}{\Gamma \vdash \mathsf{snd}\; e : \tau_2 \Rightarrow \mathsf{snd}\; |e|} \qquad (\textsc{tr-snd})$$

$$\frac{\Gamma \vdash e : \tau_1 \Rightarrow |e|}{\Gamma \vdash \mathsf{inl}_{\tau_2}\; e : \tau_1 + \tau_2 \Rightarrow \Pi^{\tau_1 + \tau_2}_{\omega}\, (\mathsf{inl}_{\tau_2}\; |e|)} \qquad (\textsc{tr-inl})$$

$$\frac{\Gamma \vdash e : \tau_2 \Rightarrow |e|}{\Gamma \vdash \mathsf{inr}_{\tau_1}\; e : \tau_1 + \tau_2 \Rightarrow \Pi^{\tau_1 + \tau_2}_{\omega}\, (\mathsf{inr}_{\tau_1}\; |e|)} \qquad (\textsc{tr-inr})$$

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

$$(\textsc{tr-case})$$

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

$$(\textsc{tr-fix})$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \rightharpoonup \tau \Rightarrow |e_1| \qquad \Gamma \vdash e_2 : \tau_2 \Rightarrow |e_2|}{\Gamma \vdash e_1\, e_2 : \tau \Rightarrow |e_1|\, |e_2|} \qquad (\textsc{tr-app})$$

$$\frac{\Gamma \vdash e : \rho \Rightarrow |e|}{\Gamma \vdash \mathsf{out}\; e : \tau_\rho \Rightarrow \mathsf{out}\; |e|} \qquad (\textsc{tr-out})$$

$$\frac{\Gamma \vdash e : \tau_\rho \Rightarrow |e|}{\Gamma \vdash \mathsf{in}\; e : \rho \Rightarrow \Pi^{\rho}_{\omega}\, (\mathsf{in}\; |e|)} \qquad (\textsc{tr-in})$$

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

Lemma 3.57 *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 $|E| \overset{\mathrm{def}}{=} [\_\tau/z]e'$, and by induction on the derivation $|E|\{\_\tau\} \in \mathrm{ECtx}_{\tau'}$.

Lemma 3.58 *For all* $e \in \mathit{Exp}_\tau(\Gamma)$ *and for all expression substitutions* γ *for* Γ*, if* $\Gamma \vdash e : \tau \Rightarrow |e|$*, then* $\vdash \Pi^\tau_\omega\,(\gamma|e|) \approx \gamma|e| : \tau$*.*

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 3.55.

Case TR-FST: By induction we get that

$$\vdash \gamma|e| \approx \Pi^{\tau_1 \times \tau_2}_\omega\,(\gamma|e|) : \tau_1 \times \tau_2 \tag{12}$$

We are to show $\vdash \mathsf{fst}\,(\gamma|e|) \approx \Pi^{\tau_1}_\omega\,(\mathsf{fst}\,(\gamma|e|)) : \tau_1 \times \tau_2$. If $\gamma|e| \Uparrow$ then it follows by Lemma 2.16. 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 (12), Lemmas 3.7 and 3.39 and transitivity of $\approx$,

$$\vdash \gamma|e| \approx (\Pi^{\tau_1}_\omega\, v_1, \Pi^{\tau_2}_\omega\, v_2) : \tau_1 \times \tau_2 \tag{13}$$

By Lemmas 3.7, 3.9, 3.10, and (13), we get

$$\vdash \mathsf{fst}\,(\gamma|e|) \approx \Pi^{\tau_1}_\omega\, v_1 : \tau_1 \tag{14}$$

Further, again using Lemmas 3.7 and 3.10,

$$\vdash \mathsf{fst}\,(\gamma|e|) \approx v_1 : \tau_1 \tag{15}$$

so by composition of evaluation contexts, (15) gives

$$\vdash \Pi^{\tau_1}_\omega\,(\mathsf{fst}\,(\gamma|e|)) \approx \Pi^{\tau_1}_\omega\, v_1 : \tau_1 \tag{16}$$

which together with (14) 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

$$\vdash \Pi^{\tau}_{\omega}\,(\mathsf{case}(\gamma|e_1|, \gamma|e_2|, \gamma|e_3|)) \approx \mathsf{case}(\gamma|e_1|, \gamma|e_2|, \gamma|e_3|) : \tau.$$

If $\gamma|e| \Uparrow$ then it follows by Lemma 2.16. 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 $\vdash \Pi^{\tau}_{\omega}\,(\gamma|e_2|\,(v_1)) \approx \gamma|e_2|\,(v_1) : \tau$. Assume $\gamma|e_2| \mapsto^{*} v$ (otherwise the required follows by Lemma 2.16). By induction we have

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

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

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

Thus it suffices to show

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

But

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

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

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

but this follows from Corollary 3.55.

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

Case TR-APP: Follows by induction and Corollary 3.55.

Case TR-OUT: Follows by induction and Corollary 3.55. □

Lemma 3.59 *For all $e \in \mathit{Exp}_{\tau}(\Gamma)$ 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 3.34 and Lemma 3.46. For rule TR-FIX, by compactness it suffices to show, for all $i \in N$,

$$\vdash \gamma(\mathsf{fix}\, f^{i}(x{:}\tau_1){:}\tau_2.|e|) \preceq \gamma'(\mathsf{fix}\, f^{i}(x{:}\tau_1){:}\tau_2.(e)) : \tau$$

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

Corollary 3.60 *If* $\emptyset \vdash e : \tau \Rightarrow |e|$, *then* $\vdash |e| \preceq e : \tau$.

Proof By Lemma 3.59. □

Corollary 3.61 *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 3.59. □

Lemma 3.62 *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.

□

Lemma 3.63 *For all* τ *and for all* $v \in Val_{\tau}$ *the following holds.*

1. $|v| \Downarrow$
2. $\Pi^{\tau}_{\omega}\, |v| \Downarrow$

Proof By induction on v. □

Lemma 3.64 *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 3.9 repeatedly without explicit mentioning.

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

$$\begin{array}{rcll}
|e| & = & |E\{r\}| & \\
 & \approx & |E|\{|r|\} & \text{by Lemma 3.62, item 2} \\
 & = & |E|\{|\mathsf{out}\ (\mathsf{in}\ v)|\} & \\
 & = & |E|\{\mathsf{out}\ (\Pi^{\rho}_{\omega}\ (\mathsf{in}\ |v|))\} & \text{by definition} \\
 & \approx & |E|\{\mathsf{out}\ (\Pi^{\rho}_{\omega}\ (\mathsf{in}\ v'))\} & \text{by Lemma 3.63, } \exists v' : |v| \mapsto^{*} v' \\
 & \approx & |E|\{\mathsf{out}\ (\mathsf{in}\ (\Pi^{\tau_\rho}_{\omega}\ v'))\} & \text{by Lemmas 3.34 and 3.43} \\
 & \approx & |E|\{\mathsf{out}\ (\mathsf{in}\ (\Pi^{\tau_\rho}_{\omega}\ |v|))\} & \\
 & \approx & |E|\{\mathsf{out}\ (\mathsf{in}\ v'')\} & \text{by Lemma 3.63, } \exists v'' : \Pi^{\tau_\rho}_{\omega}\ |v| \mapsto^{*} v'' \\
 & \approx & |E|\{v''\} & \text{by R-OUT} \\
 & \approx & |E|\{\Pi^{\tau_\rho}_{\omega}\ |v|\} & \\
 & \approx & |E|\{|v|\} & \text{by Lemma 3.58} \\
 & \approx & |e'| &
\end{array}$$

Case R-BETA: We reason as follows.

$$\begin{array}{rcll}
|e| & \approx & |E|\{|(\mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.e_1)\ v|\} & \text{by Lemma 3.62, item 2} \\
 & \approx & |E|\{(\Pi^{\tau_1 \rightharpoonup \tau_2}_{\omega}\ (\mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.|e_1|))\ |v|\} & \text{by definition} \\
 & \approx & |E|\{(\lambda x{:}\tau_1.\Pi^{\tau_2}_{\omega}\ ((\mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.|e_1|)\ (\Pi^{\tau_1}_{\omega}\ x)))\ |v|\} & \text{by Lemma 3.41} \\
 & \approx & |E|\{(\lambda x{:}\tau_1.\Pi^{\tau_2}_{\omega}\ ((\mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.|e_1|)\ (\Pi^{\tau_1}_{\omega}\ x)))\ v'\} & \text{by Lemma 3.63, } \exists v' : |v| \mapsto^{*} v' \\
 & \approx & |E|\{\Pi^{\tau_2}_{\omega}\ ((\mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.|e_1|)\ (\Pi^{\tau_1}_{\omega}\ v'))\} & \text{by R-BETA} \\
 & \approx & |E|\{\Pi^{\tau_2}_{\omega}\ ((\mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.|e_1|)\ (\Pi^{\tau_1}_{\omega}\ |v|))\} & \\
 & \approx & |E|\{\Pi^{\tau_2}_{\omega}\ ((\mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.|e_1|)\ |v|)\} & \text{by Lemma 3.58} \\
 & \approx & |E|\{\Pi^{\tau_2}_{\omega}\ ((\mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.|e_1|)\ v')\} & \\
 & \approx & |E|\{\Pi^{\tau_2}_{\omega}\ ([\mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.|e_1|, v'/f, x]|e_1|)\} & \text{by R-BETA} \\
 & \approx & |E|\{\Pi^{\tau_2}_{\omega}\ ([\mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.|e_1|, |v|/f, x]|e_1|)\} & \text{by Lemma 3.34} \\
 & \approx & |E|\{[\mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.|e_1|, |v|/f, x]|e_1|\} & \text{by Lemma 3.58} \\
 & \approx & |E|\{|[\mathsf{fix}\ f(x{:}\tau_1){:}\tau_2.e_1, v/f, x]e_1|\} & \text{by Lemma 3.62, item 1} \\
 & \approx & |e'| & \text{by Lemma 3.62, item 2}
\end{array}$$

Case R-FST: We reason as follows.

$$\begin{array}{rcll}
|e| & \approx & |E|\{|\mathsf{fst}\ ((v_1, v_2))|\} & \text{by Lemma 3.62, item 2} \\
 & \approx & |E|\{\mathsf{fst}\ ((|v_1|, |v_2|))\} & \text{by definition} \\
 & \approx & |E|\{\mathsf{fst}\ ((v_1', v_2'))\} & \text{by Lemma 3.63, } \exists v_1' : |v_1| \mapsto^{*} v_1' \text{ and } \exists v_2' : |v_2| \mapsto^{*} v_2' \\
 & \approx & |E|\{v_1'\} & \text{by R-FST} \\
 & \approx & |E|\{|v_1|\} & \\
 & \approx & |e'| & \text{by Lemma 3.62, item 2}
\end{array}$$

Case R-SND: Similar to the R-FST case.
Case R-CASE-INL: We reason as follows.

$$\begin{array}{lll}
|e| \approx & |E|\{|\mathsf{case}(\mathsf{inl}_{\tau_2}\ v, e_1, e_2)|\} & \text{by Lemma 3.62, item 2} \\
\approx & |E|\{\mathsf{case}(\mathsf{inl}_{\tau_2}\ |v|, |e_1|, |e_2|)\} & \text{by definition} \\
\approx & |E|\{\mathsf{case}(\mathsf{inl}_{\tau_2}\ v', |e_1|, |e_2|)\} & \text{by Lemma 3.63, } \exists v' : |v| \mapsto^* v' \\
\approx & |E|\{|e_1|\ v'\} & \text{by R-CASE-INL} \\
\approx & |E|\{|e_1|\ |v|\} & \\
\approx & |E|\{|e_1\ v|\} & \text{by definition} \\
\approx & |e'| & \text{by Lemma 3.62, item 2}
\end{array}$$

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

Lemma 3.65 $\vdash \lambda x{:}\rho.x \preceq \pi_\omega : \rho \rightharpoonup \rho$

Proof By Corollary 3.15 and Canonical Forms Lemma (Lemma 2.11) it suffices to show, for all $E\{_{-\rho \rightharpoonup \rho}\,(\mathsf{in}\ v)\} \in \mathrm{ECtx}_1$,

$$E\{(\lambda x{:}\rho.x)\ (\mathsf{in}\ v)\} \preceq^k E\{\pi_\omega\,(\mathsf{in}\ v)\}$$

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

$$E\{\mathsf{in}\ v\} \preceq^k E\{\pi_\omega\,(\mathsf{in}\ v)\}$$

By Corollary 3.60 it then suffices to show,

$$E\{\mathsf{in}\ v\} \preceq^k E\{\pi_\omega\,|\mathsf{in}\ v|\}$$

Since clearly $\vdash \pi_\omega \approx \Pi^\rho_\omega : \rho \rightharpoonup \rho$, by Lemma 3.58 it then suffices to show,

$$E\{\mathsf{in}\ v\} \preceq^k E\{|\mathsf{in}\ v|\} \tag{17}$$

Suppose that

$$E\{\mathsf{in}\ v\} \preceq^k |E\{\mathsf{in}\ v\}| \tag{18}$$

holds. Assuming this, we can reason as follows

$$\begin{array}{llll}
E\{\mathsf{in}\ v\} \mapsto^* * & \Rightarrow & |E\{\mathsf{in}\ v\}| \mapsto^* * & \text{by assumption (18)} \\
& \Rightarrow & |E|\{|\mathsf{in}\ v|\} \mapsto^* * & \text{by Lemma 3.62, item 2} \\
& \Rightarrow & E\{|\mathsf{in}\ v|\} \mapsto^* * & \text{by Corollary 3.61}
\end{array}$$

which gives (17) as required.

Thus we are left with showing (18). 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 $|e| = \Pi^1_\omega * \mapsto^* *$, as required.

Inductive Step: Assume $e \mapsto e' \mapsto^m *$. Then by induction we get that $|e'| \mapsto^* *$. By Lemma 3.64, 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 (Pitts, 1996).

Theorem 3.66 (Syntactic Minimal Invariance) $\vdash \pi_\omega \approx \lambda x{:}\rho.x : \rho \rightharpoonup \rho$

Proof By Lemmas 3.46, 3.65, 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_ω 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 syntactical minimal invariant property plays a central rôle in adapting Pitts' method (Pitts, 1996) 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* $((\forall i \in 1..n : e_i \Uparrow) \vee (\exists v_1, \ldots, v_n : \forall i \in 1..n : e_i \mapsto^* v_i \wedge (v_1, \ldots, v_n) \in R))$

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 $F^i_\omega = \mathsf{fix}\, f(x{:}\tau_1){:}\tau_2.e_i \in Exp_{\tau_1 \rightharpoonup \tau_2}$, and for all $I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|})$,*

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

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 standard formation of function spaces between relations, where an expression, loosely speaking, is related iff it maps related arguments to related results. Indeed, suppose $e \in R_2$ and 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.*

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

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

$$\mathsf{0} \stackrel{\text{def}}{=} \{\, (e_1, \ldots, e_n) \in (\mathit{Exp}_0 /\!\approx)^n \;\mid\; \forall i \in 1..n : e_i \Uparrow \}$$

Definition 4.5

$$\mathsf{1} \stackrel{\text{def}}{=} \{\, (e_1, \ldots, e_n) \in (\mathit{Exp}_1 /\!\approx)^n \;\mid\; (\forall i \in 1..n : e_i \Uparrow) \vee (\forall i \in 1..n : e_i \mapsto^* *) \,\}$$

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

$$\begin{aligned} R_1 \times R_2 \stackrel{\text{def}}{=} \; & \{\, (e_1, \ldots, e_n) \in (\mathit{Exp}_{\tau_1 \times \tau_2} /\!\approx)^n \mid \\ & (\forall i \in 1..n : e_i \Uparrow) \vee \\ & (\exists v_1, \ldots, v_n, v'_1, \ldots, v'_n : \forall i \in 1..n : \; \vdash e_i \approx (v_i, v'_i) : \tau_1 \times \tau_2 \\ & \qquad \wedge (v_1, \ldots, v_n) \in R_1 \wedge (v'_1, \ldots, v'_n) \in R_2) \,\} \end{aligned}$$

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

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

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

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

Lemma 4.9 *For all τ, $(\mathit{Radm}_\tau, \subseteq)$ is a complete lattice.*

Proof By a standard lattice-theory theorem (see, e.g., (Davey & Priestley, 1990, Theorem 2.16(ii))) it suffices to show that the greatest lower bound, $\bigwedge S$, exists for every subset of Radm_τ. Thus let S be an arbitrary subset of Radm_τ. Define $\bigwedge S \stackrel{\text{def}}{=} \bigcap S$. We then have to show

1. $\bigwedge S \in \mathit{Radm}_\tau$
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 $I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|})$, letting*

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

then

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

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 3.21 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). $\square$

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 $\mathsf{0}$, $\mathsf{1}$, $R_1 \times R_2$, $R_1 + R_2$, 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 (Sketch) Suppose $n = 2$ and suppose $C_1\{F^1_{\vec{\omega}}\} \Downarrow$ and $C_2\{F^2_{\vec{\omega}}\} \Uparrow$. Then for some $\vec{m} \in I$, $C_1\{F^1_{\vec{m}}\} \Downarrow$. By assumption, $(C_1\{F^1_{\vec{m}}\}, C_2\{F^2_{\vec{m}}\}) \in R$, so by strictness of R, also $C_2\{F^2_{\vec{m}}\} \Downarrow$, contradicting $C_2\{F^2_{\vec{\omega}}\} \Uparrow$. □

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 3.9.

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

$$\forall \vec{m} \in I : (C_1\{F^1_{\vec{m}}\}, \ldots, C_n\{F^n_{\vec{m}}\}) \in R_1 \times R_2 \tag{19}$$

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 $\exists v_1, \ldots, v_n : \forall i \in 1..n : C_i\{F^i_{\vec{\omega}}\} \mapsto^* v_i$. By Lemma 3.23, 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

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

and

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

Let

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

Clearly, $I'' \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}'|})$. By (19), Lemma 3.7 and definition of I'', we have,

$$\forall \vec{m} \in I'' : (V_1\{F^1_{\vec{m}}\}, \ldots, V_n\{F^n_{\vec{m}}\}) \in R_1 \times R_2 \tag{20}$$

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 (20) and definition of $R_1 \times R_2$ we then have

$$\forall \vec{m} \in I'' : (V_{11}\{F^1_{\vec{m}}\}, \ldots, V_{n1}\{F^n_{\vec{m}}\}) \in R_1 \tag{21}$$

and

$$\forall \vec{m} \in I'' : (V_{12}\{F^1_{\vec{m}}\}, \ldots, V_{n2}\{F^n_{\vec{m}}\}) \in R_2 \tag{22}$$

By admissibility of R_1 and (21) we then get

$$(V_{11}\{F^1_{\vec{\omega}}\}, \ldots, V_{n1}\{F^n_{\vec{\omega}}\}) \in R_1 \tag{23}$$

and by admissibility of R_2 and (22) we get

$$(V_{12}\{F^1_{\vec{\omega}}\}, \ldots, V_{n2}\{F^n_{\vec{\omega}}\}) \in R_2 \tag{24}$$

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

$$(V_1\{F^1_{\vec{\omega}}\}, \ldots, V_n\{F^n_{\vec{\omega}}\}) \in R_1 \times R_2 \tag{25}$$

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.

$\square$

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 3.9.

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

$$\forall \vec{m} \in I : (C_1\{F^1_{\vec{m}}\}, \ldots, C_n\{F^n_{\vec{m}}\}) \in R_1 + R_2 \tag{26}$$

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 $\exists v_1, \ldots, v_n : \forall i \in 1..n : C_i\{F^i_{\vec{\omega}}\} \mapsto^* v_i$. By Lemma 3.23, 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

$$I_i \overset{\text{def}}{=} \{\, \vec{m}\vec{m}' \mid \vec{m} \in I \wedge C_i\{F^i_{\vec{m}}\} \mapsto^* V_i\{F^i_{\vec{m}'}\}\,\} \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|}) \quad (27)$$

and

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

Let

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

Clearly, $I'' \in \mathcal{P}_{\text{cof}}(N^{|\vec{\mathsf{p}}'|})$. By (26), Lemma 3.7 and definition of I'', we have,

$$\forall \vec{m} \in I'' : (V_1\{F^1_{\vec{m}}\}, \ldots, V_n\{F^n_{\vec{m}}\}) \in R_1 + R_2 \quad (28)$$

By Canonical Forms Lemma,

$$\exists V_{11}, \ldots, V_{n1}, V_{12} \cdots V_{n2} : \forall i \in 1..n : (V_i = \mathsf{inl}_{\tau_2}\ V_{i1} \vee V_i = \mathsf{inr}_{\tau_1}\ V_{i2})$$

Claim:

$$(\exists V_{11}, \ldots, V_{n1} : \forall i \in 1..n : V_i = \mathsf{inl}_{\tau_2}\ V_{i1}) \vee (\exists V_{12}, \ldots, V_{n2} : \forall i \in 1..n : V_i = \mathsf{inl}_{\tau_1}\ V_{i2})$$

Proof of Claim: By contradiction (of the assumption (26)), using Lemma 3.7, and (27). *(End of Proof of Claim)*

Thus there are two subcases to consider.

SubCase I: $\exists V_{11}, \ldots, V_{n1} : \forall i \in 1..n : V_i = \mathsf{inl}_{\tau_2}\ V_{i1}$. Now proceed as in the proof of Lemma 4.12, using admissibility of R_1.

SubCase II: $\exists V_{12}, \ldots, V_{n2} : \forall i \in 1..n : V_i = \mathsf{inl}_{\tau_1}\ V_{i2}$. Now proceed as in the proof of Lemma 4.12, using admissibility of R_2.

□

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 3.9.

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

$$\forall \vec{m} \in I : (C_1\{F^1_{\vec{m}}\}, \ldots, C_n\{F^n_{\vec{m}}\}) \in R_1 \rightharpoonup R_2 \tag{29}$$

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 $\exists v_1, \ldots, v_n : \forall i \in 1..n : C_i\{F^i_{\vec{\omega}}\} \mapsto^* v_i$. By Lemma 3.23, 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

$$I_i \stackrel{\mathrm{def}}{=} \{\, \vec{m}\vec{m}' \mid \vec{m} \in I \wedge C_i\{F^i_{\vec{m}}\} \mapsto^* V_i\{F^i_{\vec{m}'}\} \,\} \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|}) \tag{30}$$

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 (29), Lemma 3.7 and definition of I'', we have,

$$\forall \vec{m} \in I'' : (V_1\{F^1_{\vec{m}}\}, \ldots, V_n\{F^n_{\vec{m}}\}) \in R_1 \rightharpoonup R_2 \tag{31}$$

Hence by definition of $R_1 \rightharpoonup R_2$

$$\forall \vec{m} \in I'' : \forall (e'_1, \ldots, e'_n) \in R_1 : (V_1\{F^1_{\vec{m}}\}\, e'_1, \ldots, V_n\{F^n_{\vec{m}}\}\, e'_n) \in R_2 \tag{32}$$

Let $(e'_1, \ldots, e'_n) \in R_1$ be arbitrary. Then by (32) we have

$$\forall \vec{m} \in I'' : (V_1\{F^1_{\vec{m}}\}\, e'_1, \ldots, V_n\{F^n_{\vec{m}}\}\, e'_n) \in R_2 \tag{33}$$

whence by admissibility of R_2, also

$$(V_1\{F^1_{\vec{\omega}}\}\, e'_1, \ldots, V_n\{F^n_{\vec{\omega}}\}\, e'_n) \in R_2 \tag{34}$$

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 $1 \in \mathit{Radm}_1$.

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

Strictness Follows by Lemmas 3.5, 3.7. and 3.9.

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

$$\forall \vec{m} \in I : (C_1\{F^1_{\vec{m}}\}, \ldots, C_n\{F^n_{\vec{m}}\}) \in \mathsf{1} \tag{35}$$

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 $\mathsf{1}$.

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 3.23, 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

$$I_i \overset{\mathrm{def}}{=} \{\, \vec{m}\vec{m}' \mid \vec{m} \in I \wedge C_i\{F^i_{\vec{m}}\} \mapsto^* V_i\{F^i_{\vec{m}'}\}\,\} \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{p}}'|}) \tag{36}$$

and

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

Let

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

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 (35) we have $(*, \ldots, *) \in \mathsf{1}$. Whence, by Lemma 3.7 we have that

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

as required.

□

Lemma 4.16 $\mathsf{0} \in \mathit{Radm}_0$.

Proof Immediate by the definition of $\mathsf{0}$ 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 (Pitts, 1996).

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

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

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*

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

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 component-wise is a complete lattice.*

Proof Follows by Lemma 4.9 □

We write $\sqsubseteq$ for the component-wise ordering on $(Radm^{op} \times Radm)$.

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

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

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*

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

Lemma 5.7 *Ψ is well-defined.*

Proof As in the proof of 5.3. □

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. □

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 $(\Delta^-, \Delta^+) \overset{\text{def}}{=} \mathrm{lfp}(\Psi^\S)$.*

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. □

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 $[\![0]\!]'(\Delta^+, \Delta^-) = [\![0]\!]'(\Delta^-, \Delta^+) = \mathsf{0}$ 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 3.42, so by admissibility of Δ^+, the required follows.

SubSubCase $i > 0$: Then by Lemma 3.43 (applicable 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 3.12, $\vdash \Pi_i^\rho\, e_k \approx \mathsf{in}\ v'_k : \rho$, 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 3.39, $\vdash \Pi_i^\tau\, e_k \approx (\Pi_i^{\tau_1}\, v'_k, \Pi_i^{\tau_1}\, v''_k) : \tau_1 \times \tau_2$, 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 3.39.

SubSubCase $\Pi_i^{\tau_2}\, v''_k \Uparrow$, for all $1 \leq k \leq n$: Easy using Lemma 3.39.

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 3.10, $\vdash \Pi_i^\tau\, e_k \approx (v_{k'}, v_{k''}) : \tau_1 \times \tau_2$, so by definition 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 3.40 and 3.11.

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

$$(e'_1, \ldots, e'_n) \in [\![\tau_1]\!]'(\Delta^-, \Delta^+) \Rightarrow (v_1\, e'_1, \ldots, v_n\, e'_n) \in [\![\tau_2]\!]'(\Delta^+, \Delta^-) \qquad (37)$$

By Lemma 3.41, 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 (37),

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

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. $\square$

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 3.42 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 3.43 (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. $\square$

Lemma 5.14

$$\pi_\omega : \Delta^- \subset \Delta^+$$

Proof Let $(e_1, \ldots, e_n) \in \Delta^-$. We are to show that $(\pi_\omega\, e_1, \ldots, \pi_\omega\, e_n) \in \Delta^+$. 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 3.66 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

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

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

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,*

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

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 $i \in N$,

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

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

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. □

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. □

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 $\Gamma \vdash e \approx e' : \tau$ 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. $\square$

By the above theorem, we may use logical equivalence to prove two expressions experimentally equivalent. 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 (Pitts, 1996) derives such principles. For reasons of space, we shall be less formal in our presentation of these reasoning principles than we are elsewhere. Moreover, we shall allow ourselves to elide some of the explicit typing information.

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, $\pi_\omega : R^- \subset \Delta$ and $\pi_\omega : \Delta \subset R^+$ 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_ω, as in the proof of Lemma 5.14. For the finite unrollings of π_ω, one proceeds as in the proofs of Lemmas 5.12 and 5.13. $\square$

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 (Pitts, 1995) 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. The intuition of the condition* $\mathsf{in} : [\![\tau_\rho]\!]'(R, \Delta) \subset \Delta$ *is that ρ only occurs positively in τ_ρ.*

Proof By Theorem 6.6, 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, as in Example 2.3. 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 $e\ R\ e' \Rightarrow e\ \Delta\ e'$.

Recall the definition of *map* from Example 2.3. We want to show that *map succ* (*map succ e*) is experimentally equivalent to *map* (*succ* ∘ *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 $\vdash e : \rho$. □

In the above example, we could of course equally well have proceeded by induction on the length of the list that the argument expression evaluates to. That option is not available in the following example.

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 (Pitts, 1995) 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:

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

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

The continuation-passing (cps) transformation (Fischer, 1993; Plotkin, 1975; Reynolds, 1972) is a global program transformation used in some compilers for functional languages (Steele, Jr., 1978; Appel, 1992). The main idea of the cps transformation is to make the flow of control in a program explicit through the use of higher-order functions. The translation of an expression is a function that takes as argument another function, its continuation, to which control should be passed upon completion of evaluation of that expression. Sequencing of the steps of evaluation is expressed by an explicit "hand-off" from one continuation to the next in the transformed program. In addition to making the flow of control explicit, the cps transformation also introduces bindings for all intermediate results of a computation, and makes the state of evaluation available for explicit manipulation. The latter property is especially of interest for implementing exceptions (Appel, 1992) and user-level threads (Reppy, 1991).

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 (Meyer & Wand, 1985).

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^*."

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

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

$$\Gamma \vdash * : 1 \rightsquigarrow_v * \qquad \text{(CPS-ONE)}$$

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

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

Figure 3: CPS Transformation — Part I

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

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, $\mathrm{Exp}^{\rho}_{\tau}$ denotes the set of closed expressions of type τ of the source language, whereas $\mathrm{Exp}^{\rho^*}_{\tau}$ 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 $e \in \mathrm{Exp}^{\rho}_1$ and $e' \in \mathrm{Exp}^{\rho^*}_1$, 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 $\Gamma^* \vdash e' : \overline{\tau}$.*

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

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

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

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

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

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

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

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

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

Figure 4: CPS Transformation — Part II

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

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*

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

satisfying

$$\begin{array}{rcl}
e\ \Delta^c_\tau\ e' & \iff & E\{_{-\tau}\}\ \Delta^k_\tau\ v' \Rightarrow E\{e\} \approx^k e'\,v' \\[2ex]
v\ \Delta^v_1\ v' & \iff & v = *, v' = * \\
v\ \Delta^v_0\ v' & & \textit{never} \\
v\ \Delta^v_\rho\ v' & \iff & \vdash v \approx \mathsf{in}\ v_1 : \rho,\ \vdash v' \approx \mathsf{in}\ v'_1 : \rho^*,\ v_1\ \Delta^v_{\tau_\rho}\ v'_1 \\
v\ \Delta^v_{\tau_1 \times \tau_2}\ v' & \iff & \vdash v \approx (v_1, v_2) : \tau_1 \times \tau_2,\ \vdash v' \approx (v'_1, v'_2) : {\tau_1}^* \times {\tau_2}^*, \\
& & v_1\ \Delta^v_{\tau_1}\ v'_1,\ v_2\ \Delta^v_{\tau_2}\ v'_2 \\
v\ \Delta^v_{\tau_1 + \tau_2}\ v' & \iff & \big(\vdash v \approx \mathsf{inl}_{\tau_2}\ v_1 : \tau_1 + \tau_2,\ \vdash v' \approx \mathsf{inl}_{\tau_2{}^*}\ v'_1 : {\tau_1}^* + {\tau_2}^*,\ v_1\ \Delta^v_{\tau_1}\ v'_1\big) \\
& \vee & \big(\vdash v \approx \mathsf{inr}_{\tau_1}\ v_1 : \tau_1 + \tau_2,\ \vdash v' \approx \mathsf{inr}_{\tau_1{}^*}\ v'_1 : {\tau_1}^* + {\tau_2}^*,\ v_1\ \Delta^v_{\tau_2}\ v'_1\big) \\
v\ \Delta^v_{\tau_1 \rightharpoonup \tau_2}\ v' & \iff & v_1\ \Delta^v_{\tau_1}\ v'_1 \Rightarrow v\,v_1\ \Delta^c_{\tau_2}\ v'\,v'_1 \\[2ex]
E\{_{-\tau}\}\ \Delta^k_\tau\ v' & \iff & v_1\ \Delta^v_\tau\ v'_1 \Rightarrow E\{v_1\} \approx^k v'\,v'_1,
\end{array}$$

and

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

(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 $\Gamma \vdash e : \tau \leadsto_c e'$. $\square$

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*

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

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 3.27.

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

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

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

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

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

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

$$\mathsf{0} \stackrel{\rm def}{=} \{\, (e, e') \in (Exp^{\rho}_{\mathsf{0}}/{\approx}) \times (Exp^{\rho^*}_{\mathsf{0}}/{\approx}) \mid e \Uparrow \wedge\, e' \Uparrow \}$$

Definition 7.12

$$\mathsf{1} \stackrel{\rm def}{=} \{\, (e, e') \in (Exp^{\rho}_{\mathsf{1}}/{\approx}) \times (Exp^{\rho^*}_{\mathsf{1}}/{\approx}) \mid (e \Uparrow \wedge e' \Uparrow) \vee (e \mapsto^* * \wedge\, e' \mapsto^* *) \,\}$$

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

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

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

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

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

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

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

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

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

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

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

Note that $R_1 \rightharpoonup R_2$ is anti-monotone 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; $\mathsf{0}$ and $\mathsf{1}$ 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

$$\forall \vec{m}\vec{m}' \in I \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}}|+|\vec{\mathsf{q}}|}) : (C\{F_{\vec{m}}\}, C'\{F'_{\vec{m}'}\}) \in R. \tag{38}$$

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 3.23, there exist $V\{\vec{\mathsf{p}_1}\}$ and $V'\{\vec{\mathsf{q}_1}\}$ such that

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

so

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

and

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

Thus

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

is cofinal, i.e., $I'' \in \mathcal{P}_{\mathrm{cof}}(N^{|\vec{\mathsf{p}_1}|+|\vec{\mathsf{q}_1}|})$. By (38), 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

$$\forall (E_0\{_{-\tau_2}\}, v'_0) \in \Delta^k_{\tau_2}(R_2) : E_0\{V\{F_{\vec{m}'}\}\, e\} \approx^k V'\{F'_{\vec{n}'}\}\, e'\, v'_0. \tag{39}$$

We are to show that

$$\forall (E_0\{_{-\tau_2}\}, v'_0) \in \Delta^k_{\tau_2}(R_2) : E_0\{V\{F_\omega\}\, e\} \approx^k V'\{F'_\omega\}\, e'\, v'_0. \tag{40}$$

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 3.23,

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

Hence

$$I_{11} \overset{\text{def}}{=} \{\, \vec{m}'\vec{n}' \mid \vec{m}'\vec{n}' \in I \wedge C_{11}\{F_{\vec{m}'}\} \mapsto^* * \,\}$$

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 (38), $V'\{F'_{\vec{n}'}\}\, e'\, v'_0 \mapsto^* *$, from which $V'\{F'_\omega\}\, e'\, v'_0 \mapsto^* *$ follows by Lemma 3.24. The other direction is similar, completing the proof of (40). 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 $\tau \in \text{Type}^\rho$ we define an interpretation $[\![\tau]\!]'$ exactly as in Definition 5.5.

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

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

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

We define $\Psi^\S : (\mathit{Radm}_\rho^{op} \times \mathit{Radm}_\rho) \to (\mathit{Radm}_\rho^{op} \times \mathit{Radm}_\rho)$ 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* $\tau \in \text{Type}^\rho$,

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

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

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

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

$$(e_1, e_1') \in [\![\tau_1]\!]'(\Delta^-, \Delta^+) \Rightarrow (v\, e_1, v'\, e_1') \in \Delta^c_{\tau_2}\ ([\![\tau_2]\!]'(\Delta^+, \Delta^-)) \qquad (41)$$

By two applications of Lemma 3.43 we get that

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

and

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

Assume

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

It then suffices to show that

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

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

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

and

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

By induction,

$$\left(\Pi^{\tau_1}_{\rho,i}\, v_1, \Pi^{{\tau_1}^*}_{\rho^*,i}\, v_1'\right) \in [\![\tau_1]\!]'(\Delta^-, \Delta^+).$$

Hence, by (41),

$$\left(v\, \Pi^{\tau_1}_{\rho,i}\, v_1, v'\, \Pi^{{\tau_1}^*}_{\rho^*,i}\, v_1'\right) \in \Delta^c_{\tau_2}\ ([\![\tau_2]\!]'(\Delta^+, \Delta^-)).$$

By admissibility there are two cases to consider.

SubSubSubCase $v\, \Pi^{\tau_1}_{\rho,i}\, v_1 \Uparrow \wedge v'\, \Pi^{{\tau_1}^*}_{\rho^*,i}\, v_1' \Uparrow$: Easy

SubSubSubCase $\vdash v\,\Pi^{\tau_1}_{\rho,i}\,v_1 \approx v_2 : \tau_2 \wedge \vdash v'\,\Pi^{{\tau_1}^*}_{\rho^*,i}\,v'_1 \approx v'_2 : \overline{\tau_2}$ for some $(v_2, v'_2) \in\!\Delta^c_{\tau_2}\ (\llbracket\tau_2\rrbracket'(\Delta^+, \Delta^-))$: Then it suffices to show that

$$(\Pi^{\tau_2}_{\rho,i}\,v_2, \Pi^{\overline{\tau_2}}_{\rho^*,i}\,v'_2) \in\!\Delta^c_{\tau_2}\ (\llbracket\tau_2\rrbracket'(\Delta^-, \Delta^+)).$$

To this end, assume

$$(E_{10}\{\_{\tau_2}\}, v_{10}) \in\!\Delta^k_{\tau_2}\ (\llbracket\tau_2\rrbracket'(\Delta^-, \Delta^+)). \tag{42}$$

We are to show that

$$E_{10}\{\Pi^{\tau_2}_{\rho,i}\,v_2\} \approx^k \Pi^{\overline{\tau_2}}_{\rho^*,i}\,v'_2\,v_{10}.$$

Since

$$\vdash \Pi^{\overline{\tau_2}}_{\rho^*,i}\,v'_2\,v_{10} \approx v'_2\,(\lambda x'{:}{\tau_2}^*.v_{10}\,(\Pi^{{\tau_2}^*}_{\rho^*,i}\,x')) : \overline{\tau_2}$$

it suffices to show

$$E_{10}\{\Pi^{\tau_2}_{\rho,i}\,v_2\} \approx^k v'_2\,(\lambda x'{:}{\tau_2}^*.v_{10}\,(\Pi^{{\tau_2}^*}_{\rho^*,i}\,x')).$$

Hence it suffices to show that

$$\Big(E_{10}\{\Pi^{\tau_2}_{\rho,i}\,\_{\tau_2}\}, \lambda x'{:}{\tau_2}^*.v_{10}\,(\Pi^{{\tau_2}^*}_{\rho^*,i}\,x')\Big) \in\!\Delta^k_{\tau_2}\ (\llbracket\tau_2\rrbracket'(\Delta^+, \Delta^-)).$$

(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

$$(e_{11}, e'_{11}) \in \llbracket\tau_2\rrbracket'(\Delta^+, \Delta^-). \tag{43}$$

We are to show that

$$E_{10}\{\Pi^{\tau_2}_{\rho,i}\,e_{11}\} \approx^k (\lambda x'{:}{\tau_2}^*.v_{10}\,(\Pi^{{\tau_2}^*}_{\rho^*,i}\,x'))\,e'_{11}.$$

Since

$$\vdash (\lambda x'{:}{\tau_2}^*.v_{10}\,(\Pi^{{\tau_2}^*}_{\rho^*,i}\,x'))\,e'_{11} \approx v_{10}\,(\Pi^{{\tau_2}^*}_{\rho^*,i}\,e'_{11}) : 1$$

it suffices to show

$$E_{10}\{\Pi^{\tau_2}_{\rho,i}\,e_{11}\} \approx^k v_{10}\,(\Pi^{{\tau_2}^*}_{\rho^*,i}\,e'_{11}). \tag{44}$$

But by induction on (43),

$$\Big(\Pi^{\tau_2}_{\rho,i}\,e_{11}, \Pi^{{\tau_2}^*}_{\rho^*,i}\,e'_{11}\Big) \in \llbracket\tau_2\rrbracket'(\Delta^-, \Delta^+),$$

so by assumption (42), the required (44) follows. □

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

$$(\pi_{\rho,\omega}, \pi_{\rho^*,\omega}) : \Delta^- \subset \Delta^+$$

Proof As the proof of Lemma 5.14. $\square$

As in Section 5, it now follows that $\Delta^- = \Delta^+$ and we can define $\Delta \overset{\text{def}}{=} \Delta^+$.

Definition 7.23 *For all source types $\tau \in \text{Type}^\rho$, we define*

$$\Delta^e_\tau \overset{\text{def}}{=} [\![\tau]\!]'(\Delta, \Delta).$$

Definition 7.24 *For all source types $\tau \in \text{Type}^\rho$, we define*

$$\begin{array}{lcl}
\Delta^v_\tau & \overset{\text{def}}{=} & \{\, (e, e') \in \Delta^e_\tau \mid e \Downarrow \wedge e' \Downarrow \} \\
\Delta^k_\tau & \overset{\text{def}}{=} & \{\, (E\{\_\tau\}, v') \in (\mathit{ECtx}^\rho_\tau /\approx) \times (\mathit{Val}^{\rho^*}_{\tau^* \rightharpoonup 1} /\approx) \mid (v_1, v_1') \in \Delta^v_\tau \Rightarrow E\{v\} \approx^k v'\, v \,\} \\
\Delta^c_\tau & \overset{\text{def}}{=} & \{\, (e, e') \in (\mathit{Exp}^\rho_\tau /\approx) \times (\mathit{Exp}^{\rho^*}_{\overline{\tau}} /\approx) \mid (E\{\_\tau\}, v') \in \Delta^k_\tau \Rightarrow E\{e\} \approx^k e'\, v' \,\}
\end{array}$$

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 $\mathsf{fix}\, f(x{:}\tau_1){:}\tau_2.e \Downarrow$ and $\mathsf{fix}\, f(x{:}{\tau_1}^*){:}\tau_2.e' \Downarrow$. $\square$

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 (Pitts, 1996) 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 (Mason *et al.* , 1995) 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 (Pitts, 1995). 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 (Reynolds, 1974a) gives a proof for an untyped functional language by working over a domain model given by an inverse limit construction. Meyer & Wand (Meyer & Wand, 1985) 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 passage 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 (Pitts, 1995) 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` (Clinger & Rees, 1991; Harper *et al.* , 1993). 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 (Filinski, 1996) 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, Søren Lassen and the referees for their very 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

Appel, Andrew W. 1992. *Compiling with Continuations.* Cambridge University Press.

Clinger, William, & Rees, Jonathan. 1991. Revised^4 Report on the Algorithmic Language Scheme. *LISP Pointers*, **IV**(3), 1–55.

Davey, B. A., & Priestley, H. A. 1990. *Introduction to Lattices and Order.* Cambridge University Press.

Filinski, Andrzej. 1996 (May). *Controlling Effects.* CMU–CS–96–119, School of Computer Science, Carnegie Mellon University.

Fischer, Michael J. 1993. Lambda-Calculus Schemata. *LISP and Symbolic Computation*, **6**(3/4), 259–288.

Girard, Jean-Yves. 1972. *Interprétation Fonctionnelle et Élimination des Coupures dans l'Arithmétique d'Ordre Supérieure.* Ph.D. thesis, Université Paris VII.

Gunter, Carl A. 1992. *Semantics of Programming Languages. Structures and Techniques.* MIT Press.

Harper, Robert, Duba, Bruce, & MacQueen, David. 1993. Typing First-Class Continuations in ML. *Journal of Functional Programming*, **3**(4), 465–484.

Mason, Ian A., Smith, Scott F., & Talcott, Carolyn L. 1995. From Operational Semantics to Domain Theory. *Information and Computation.* To Appear.

Meyer, Albert R., & Wand, Mitchell. 1985. Continuation Semantics in Typed Lambda Calculi (Summary). *Pages 219–224 of:* Parikh, Rohit (ed), *Logics of Programs.* Lecture Notes in Computer Science, vol. 193. Springer-Verlag.

Milner, Robin, Tofte, Mads, Harper, Robert, & MacQueen, David. 1997. *The Definition of Standard ML (Revised).* MIT Press.

Morrisett, John Gregory. 1995 (December). *Compiling with Types.* Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. (Available as Carnegie Mellon University School of Computer Science technical report CMU–CS–95–226.).

Pitts, Andrew M. 1995 (Sept.). 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.

Pitts, Andrew M. 1996. Relational Properties of Domains. *Information and Computation*, **127**(2), 66–90.

Plotkin, Gordon. 1975. Call-by-Name, Call-by-Value, and the Lambda Calculus. *Theoretical Computer Science*, **1**, 125–159.

Plotkin, Gordon. 1977. LCF Considered as a Programming Language. *Theoretical Computer Science*, **5**, 223–257.

Plotkin, Gordon. 1983. *Domains.* Department of Computer Science. University of Edinburgh.

Reppy, John H. 1991 (June). CML: A Higher-Order Concurrent Language. *Pages 293–305 of: Proc. 1991 ACM SIGPLAN Conference on Programming Language Design and Implementation.*

Reynolds, John C. 1972 (August). Definitional Interpreters for Higher-Order Programming Languages. *Pages 717–740 of: Conference Record of the 25th National ACM Conference.* ACM, Boston.

Reynolds, John C. 1974a. On The Relation Between Direct and Continuation Semantics. *Pages 141–156 of:* Loeckx, J. (ed), *Proceedings of the Second Colloquium on Automata, Languages and Programming, Saarbrücken.* Lecture Notes in Computer Science, vol. 174. Springer-Verlag.

Reynolds, John C. 1974b. Towards a Theory of Type Structure. *Pages 408–423 of: Colloq. sur la Programmation.* Lecture Notes in Computer Science, vol. 19. Springer-Verlag.

Scott, Dana S. 1982. Domains for Denotational Semantics. *Pages 577–613 of:* Nielsen, M., & Schmidt, E. M. (eds), *Proceedings of the 9th International Colloquium on Automata, Languages and Programming.* Lecture Notes in Computer Science, vol. 140. Springer-Verlag.

Shao, Zhong, League, Christopher, & Monnier, Stefan. 1998 (September). Implementing Typed Intermediate Languages. *Pages 313–323 of: Proceedings of the 1998 ACM SIGPLAN International Conference on Functional Programming.* ACM SIGPLAN, Baltimore, MD.

Statman, Richard. 1985. Logical Relations and the Typed λ-Calculus. *Information and Control*, **65**, 85–97.

Steele, Jr., Guy L. 1978. *RABBIT: A Compiler for SCHEME.* Tech. rept. Memo 474. MIT AI Laboratory.

Tarditi, David, Morrisett, Greg, Cheng, Perry, Stone, Chris, Harper, Robert, & Lee, Peter. 1996 (May). TIL: A Type-Directed Optimizing Compiler for ML. *Pages 181–192 of: ACM SIGPLAN Conference on Programming Language Design and Implementation.*