A Model for Formal Parametric Polymorphism: a Per Interpretation for System R

Martin ABADI
Roberto BELLUCCI
Pierre-Louis CURIEN

LIENS - 95 - 19

A Model for Formal Parametric Polymorphism: a Per Interpretation for System R

Martin ABADI[*]
Roberto BELLUCCI[**]
Pierre-Louis CURIEN

LIENS - 95 - 19

July 1995

Laboratoire d'Informatique de l'Ecole Normale Supérieure
45 rue d'Ulm 75230 PARIS Cedex 05

Tel : (33)(1) 44 32 00 00
Adresse électronique : .. @dmi.ens.fr

[*]Digital Equipment Corporation, Systems Research Center, USA

[**]LIENS and
Dipartimento di Matematica, Universita di Siena Italy

A model for formal parametric polymorphism: a per interpretation for system $\mathcal{R}$

Roberto Bellucci [*] Martín Abadi [†] Pierre-Louis Curien [‡]

July 6, 1995

Abstract

System $\mathcal{R}$ is an extension of system F that formalizes Reynolds' notion of relational parametricity. In this paper we describe a semantics for system $\mathcal{R}$. As a first step, we give a careful and general reconstruction of the per model of system F of Bainbridge et al., presenting it as a categorical model in the sense of Seely. Then we interpret system $\mathcal{R}$ in this model.

[*]LIENS, CNRS - Département de Mathématiques et Informatique de l'Ecole Normale Supérieure and Dipartimento di Matematica, Università di Siena, Italy

[†]Digital Equipment Corporation, Systems Research Center

[‡]LIENS, CNRS - Département de Mathématiques et Informatique de l'Ecole Normale Supérieure

Contents

1 From models of F to models of $\mathcal{R}$

The principle of parametricity has gone through many avatars.

Originally Strachey distinguished parametric polymorphism and ad hoc polymorphism [Str67]. Strachey described parametric polymorphism as the pure polymorphism of functions like *append*, which works on lists of any type uniformly. In contrast, a function like *print* examines and branches on the types of its arguments, and hence Strachey deemed its polymorphism ad hoc.

Reynolds formalized Strachey's notion of parametricity [Rey83], in his attempt to define a set-theoretic model for Girard's system F [Gir72]. According to Reynolds' semantic definition, a polymorphic function is parametric if its instances at related types are related. For example, take a polymorphic function f of type $\forall(X).X \rightarrow X$ (the type of the identity function). If X is instantiated to two types A and B with a relation R between them, and a has type A, b has type B, and aRb, then we must obtain $f(a)Rf(b)$. All of the definable functions of system F are parametric, but system F admits models with non-definable, non-parametric elements.

Since Reynolds' work, there have been many studies of parametricity. In particular, Bainbridge et al. introduced a view of parametricity based on dinaturality [BFSS90]. They also developed Reynolds' ideas in a variant of the partial-equivalence-relation (per) model. It seems still unknown whether the standard per model is parametric without modification.

System $\mathcal{R}$ is an extension of system F that formalizes a parametricity requirement. The intent is to capture Reynolds' notion of relational parametricity in a formal system, with simple syntax, at a suitable level of abstraction, and without reference to a particular model. Other formal systems with similar features exist, serving related purposes [Mai91, MR91, PA93]; see [ACC93] for a comparison.

In a preliminary version of system $\mathcal{R}$, quantification over pers and quantification over relations were equated. This equation was rather attractive but too daring; Hasegawa ingeniously exploited it to derive an inconsistency. The inconsistency naturally stimulated our interest in semantics, and particularly in the problem of harmonizing the use of pers and the use of general relations.

Both pers and relations have a place in the semantics and in the logic of parametricity. Pers are important as the denotations of types. From the point of view of formal reasoning, pers are the basis of equational reasoning. Relations enforce the requirement of parametricity in the construction of types, as in Reynolds' original (non-existent) set-theoretic model and in the model of Bainbridge et al. Logically, relations correspond to predicates, and many useful ones can be defined from the graphs of definable functions. Thus, both pers and relations play a role in system $\mathcal{R}$, and a semantics for system $\mathcal{R}$ should shed some light on their interaction.

In this paper we show how a parametric model of system F can be extended to a parametric model of system $\mathcal{R}$. This extension might be possible for any parametric model of system F, but we carry it out for the modified per model of Bainbridge et al. Along the way, we give a precise and general reconstruction of this per model. We present it as a categorical model in the sense of Seely [See87].

The next two sections introduce the syntax of system $\mathcal{R}$ and the necessary categories of pers and relations. Section 4 defines two parametric semantics of system F; then section 5 extends one of these semantics to system $\mathcal{R}$.

2 System $\mathcal{R}$

This section is an introduction to system $\mathcal{R}$, adapted from [ACC93]. A list of all the rules of systems F and $\mathcal{R}$ can be found in the appendix.

System $\mathcal{R}$ is a formal system with judgements and rules in the style of those of F. In order to deal explicitly with relational parametricity, the judgements of $\mathcal{R}$ generalize those of F; they are:

$\vdash_{\mathcal{R}} E$	E is a legal environment
$E \vdash_{\mathcal{R}} \begin{array}{c}\sigma\\ \mathcal{S}\\ \tau\end{array}$	$\mathcal{S}$ is a relation between types σ and τ in E
$E \vdash_{\mathcal{R}} \begin{array}{c}M:\sigma\\ \mathcal{S}\\ N:\tau\end{array}$	$\mathcal{S}$ relates M of type σ and N of type τ in E

A built-in equality judgement on values is not necessary. Instead of writing that M and N are equal in σ, we can turn the type σ into a relation σ^* (intuitively, the identity relation on σ) and write that σ^* relates M and N. Similarly, there is no need for a built-in typing judgement. We write:

$$E \vdash_{\mathcal{R}} M : \sigma \qquad \text{for} \qquad E \vdash_{\mathcal{R}} \begin{array}{c}M:\sigma\\ \sigma^*\\ M:\sigma\end{array}$$

$$E \vdash_{\mathcal{R}} M = N : \sigma \qquad \text{for} \qquad E \vdash_{\mathcal{R}} \begin{array}{c}M:\sigma\\ \sigma^*\\ N:\sigma\end{array}$$

The environments of $\mathcal{R}$ are lists of components of two sorts, directly inspired by the corresponding ones for F environments:

$\begin{array}{c}X\\ \mathcal{W}\\ Y\end{array}$	$\mathcal{W}$ is a relation variable between type variables X (domain) and Y (codomain)
$\begin{array}{c}x:\sigma\\ \mathcal{S}\\ y:\tau\end{array}$	variables x and y have types σ and τ, respectively, and are related by $\mathcal{S}$

With this notation, we now explain some of the rules of $\mathcal{R}$. In the rules, we write $x \notin M$ to mean that x is not a free variable of M. We start with rules that imitate those of F for $\rightarrow$ and $\forall$.

The introduction and elimination rules for $\to$ are, respectively:

$$\frac{E,\ \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array} \ \vdash_{\mathcal{R}}\ \begin{array}{c} M:\tau_1 \\ \mathcal{S} \\ N:\tau_2 \end{array} \qquad \begin{array}{c} x \notin N, \mathcal{S} \\ y \notin M, \mathcal{S} \end{array} \qquad E \vdash_{\mathcal{R}} \begin{array}{c} \tau_1 \\ \mathcal{S} \\ \tau_2 \end{array}}{E \vdash_{\mathcal{R}} \begin{array}{c} \lambda(x:\sigma_1).M : \sigma_1 \to \tau_1 \\ \mathcal{R} \to \mathcal{S} \\ \lambda(x:\sigma_2).N : \sigma_2 \to \tau_2 \end{array}}$$

$$\frac{E \vdash_{\mathcal{R}} \begin{array}{c} M_1 : \sigma_1 \to \tau_1 \\ \mathcal{R} \to \mathcal{S} \\ M_2 : \sigma_2 \to \tau_2 \end{array} \qquad E \vdash_{\mathcal{R}} \begin{array}{c} N_1 : \sigma_1 \\ \mathcal{R} \\ N_2 : \sigma_1 \end{array}}{E \vdash_{\mathcal{R}} \begin{array}{c} M_1 N_1 : \tau_1 \\ \mathcal{S} \\ M_2 N_2 : \tau_2 \end{array}}$$

These rules follow the same pattern as the F rules:

$$\frac{E, x : \sigma \vdash_F M : \tau}{E \vdash_F \lambda(x:\sigma).M : \sigma \to \tau} \qquad \frac{E \vdash_F M : \sigma \to \tau \qquad E \vdash_F N : \sigma}{E \vdash_F MN : \tau}$$

The introduction rule says: Assume that if $\mathcal{R}$ relates x of type σ_1 and y of type σ_2, then $\mathcal{S}$ relates M of type τ_1 and N of type τ_2. For technical reasons, assume also that $\mathcal{S}$ is a relation between τ_1 and τ_2, as one would expect. Then $\mathcal{R} \to \mathcal{S}$, a relation between $\sigma_1 \to \tau_1$ and $\sigma_2 \to \tau_2$, relates the functions $\lambda(x:\sigma_1).M$ of type $\sigma_1 \to \tau_1$ and $\lambda(y:\sigma_2).N$ of type $\sigma_2 \to \tau_2$. The elimination rule works in the opposite direction, applying related functions to related inputs and yielding related outputs.

The introduction and elimination rules for $\forall$ are:

$$\frac{E,\ \begin{array}{c} X \\ \mathcal{W} \\ Y \end{array} \ \vdash_{\mathcal{R}}\ \begin{array}{c} M:\sigma_1 \\ \mathcal{S} \\ N:\sigma_2 \end{array} \quad \begin{array}{c} X \notin \mathcal{S}, N, \sigma_2 \\ Y \notin \mathcal{S}, M, \sigma_1 \end{array}}{E \vdash_{\mathcal{R}} \begin{array}{c} \lambda(X).M : \forall(X).\sigma_1 \\ \forall(\mathcal{W}).\mathcal{S} \\ \lambda(Y).N : \forall(Y).\sigma_2 \end{array}} \qquad \frac{E \vdash_{\mathcal{R}} \begin{array}{c} M : \forall(X).\sigma_1 \\ \forall(\mathcal{W}).\mathcal{S} \\ N : \forall(Y).\sigma_2 \end{array} \qquad E \vdash_{\mathcal{R}} \begin{array}{c} \tau_1 \\ \mathcal{R} \\ \tau_2 \end{array}}{E \vdash_{\mathcal{R}} \begin{array}{c} M\tau_1 : \sigma_1[\tau_1/X] \\ \mathcal{S}[\mathcal{R}/\mathcal{W}] \\ N\tau_2 : \sigma_2[\tau_2/Y] \end{array}}$$

These rules follow the same pattern as the F rules:

$$\frac{E, X \vdash_F M : \sigma}{E \vdash_F \lambda(X).M : \forall(X).\sigma} \qquad \frac{E \vdash_F M : \forall(X).\sigma \qquad E \vdash_F \tau}{E \vdash_F M\tau : \sigma[\tau/X]}$$

The introduction rule says: Assume that if $\mathcal{W}$ is a relation between types X and Y, then $\mathcal{S}$ relates M of type σ_1 and N of type σ_2. Then $\forall(\mathcal{W}).\mathcal{S}$, a relation between $\forall(X).\sigma_1$ and $\forall(Y).\sigma_2$, relates the polymorphic terms $\lambda(X).M$ of type $\forall(X).\sigma_1$ and $\lambda(Y).N$ of type $\forall(Y).\sigma_2$. Again, the elimination rule works in the opposite direction.

Since the relation constructions parallel the type constructions, we can easily define a relation σ^* for every type σ: we replace all quantifiers over types with corresponding quantifiers over relations. This ( )* operation is used in the rules for variables:

(Rel Val $x\mathcal{R}y$)

$$\frac{\vdash_{\mathcal{R}} E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E''}{E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E'' \vdash_{\mathcal{R}} \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}}$$

(Rel Val $\mathcal{R}x$)

$$\frac{\vdash_{\mathcal{R}} E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E''}{E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E'' \vdash_{\mathcal{R}} \begin{array}{c} x:\sigma_1 \\ {\sigma_1}^* \\ x:\sigma_1 \end{array}}$$

(Rel Val $\mathcal{R}y$)

$$\frac{\vdash_{\mathcal{R}} E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E''}{E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E'' \vdash_{\mathcal{R}} \begin{array}{c} y:\sigma_2 \\ {\sigma_2}^* \\ y:\sigma_2 \end{array}}$$

The first rule is straightforward. The other two formalize the parametricity condition. Basically, they assert that the relation σ^* relates to itself any element of a type σ.

The preceding rules, together with the rules of β and η conversion, form the core of the part of $\mathcal{R}$ that deals with relations built from variables, $\rightarrow$, and $\forall$. This part of $\mathcal{R}$ is not a very powerful proof system on its own, but it suffices to encode F. In particular:

$$\text{if } E \vdash_F M = N : \sigma \text{ then } E \vdash_{\mathcal{R}} \begin{array}{c} M:\sigma \\ \sigma^* \\ N:\sigma \end{array}$$

(In the second sequent we use E as an abbreviation of an $\mathcal{R}$ environment; see the appendix for more details.)

In addition, $\mathcal{R}$ has rules for defining relations from functions:

$$\frac{E \vdash_{\mathcal{R}} M:\sigma_1 \rightarrow \sigma_2 \qquad\qquad E \vdash_{\mathcal{R}} N:\sigma_1}{E \vdash_{\mathcal{R}} \begin{array}{c} N:\sigma_1 \\ < M > \\ MN:\sigma_2 \end{array}} \qquad\qquad \frac{E \vdash_{\mathcal{R}} \begin{array}{c} N_1:\sigma_1 \\ < M > \\ N_2:\sigma_2 \end{array}}{E \vdash_{\mathcal{R}} \begin{array}{c} MN_1:\sigma_2 \\ \sigma_2^* \\ N_2:\sigma_2 \end{array}}$$

According to these rules, a function M from σ_1 to σ_2 can be viewed as a relation $< M >$ between σ_1 and σ_2 (intuitively, the graph of M).

Functional relations are essential to the power of $\mathcal{R}$. They are often useful for obtaining "free theorems" as in Wadler's work [Wad89]. They have no analogue in F.

3 Categories of pers and categories of relations

In this section we introduce the main notions and tools that we use for the definition of our model of $\mathcal{R}$. They are all well known, but we review them in order to clarify which assumptions are essential.

First we recall a general definition of model for the untyped λ-calculus. Then we consider the category of partial equivalence relations (pers) and some related categorical constructions. Finally, we introduce the category of saturated relations between pers, extending the categorical constructions previously defined for pers. Our models are based on this last category.

3.1 Combinatory algebras and λ-models

This section is a review of combinatory algebras. For more details, see for example [Bar84].

Definition 3.1 (Partial and Total Combinatory Algebras)

- *A partial combinatory algebra $\langle D, \cdot, k, s\rangle$ consists of a set D together with a partial binary operation $\cdot$ and elements $k, s \in D$ such that for all $a, b, c \in D$: $k \cdot a, s \cdot a, (s \cdot a) \cdot b$ are defined and*

$$\begin{array}{rcl} (k \cdot a) \cdot b & = & a \\ ((s \cdot a) \cdot b) \cdot c & \simeq & (a \cdot c) \cdot (b \cdot c) \end{array}$$

  where $x \simeq y$ means that x is defined if and only if y is defined, and then they are equal.

- *A (total) combinatory algebra is a partial combinatory algebra whose operation $\cdot$ is total. In this case two equations hold:*

$$\begin{array}{rcl} (k \cdot a) \cdot b & = & a \\ ((s \cdot a) \cdot b) \cdot c & = & (a \cdot c) \cdot (b \cdot c) \end{array}$$

- *A λ-model $\langle D, \cdot, k, s, \epsilon\rangle$ consists of a set D together with a binary operation $\cdot$ and elements $k, s, \epsilon \in D$ such that $\langle D, \cdot, k, s\rangle$ is a combinatory algebra and such that for all $a, b \in D$:*

$$\begin{array}{rcl} (\epsilon \cdot a) \cdot b & = & a \cdot b \\ \forall d(a \cdot d = b \cdot d) & \Rightarrow & \epsilon \cdot a = \epsilon \cdot b \quad \textit{(Weak Ext)} \\ \epsilon \cdot \epsilon & = & \epsilon \end{array}$$

Given a structure $\langle D, \ldots\rangle$, a *value assignment* is a mapping from term variables to elements of D. We may call such a value assignment untyped, as D is an untyped structure. Whenever f is a function (for example, a value assignment), we use the notation $f[e/x]$ for the function that maps x to e and is otherwise identical to f.

A combinatory-logic term (or CL-term) is one built from variables, k, and s using only application. Similarly, a CL_ϵ-term is one built from variables, k, s, and ϵ using only application. (In both cases, we simply write MN for the application of term M to term N.)

If $\langle D, \cdot, k, s\rangle$ is a combinatory algebra and ρ a value assignment, it is trivial to define the meaning of a CL-term in D:

$$\begin{array}{rcl} [\![k]\!]_\rho & = & k \\ [\![s]\!]_\rho & = & s \\ [\![x]\!]_\rho & = & \rho(x) \\ [\![M_1 M_2]\!]_\rho & = & [\![M_1]\!]_\rho \cdot [\![M_2]\!]_\rho \end{array}$$

Lemma 3.2 (Substitution Lemma for Combinatory Algebras) *For all M, N, ρ we have:*

$$[\![M[N/x]]\!]_\rho = [\![M]\!]_{\rho[[\![N]\!]_\rho/x]}$$

Furthermore, if D is a λ-model we can define the meaning of a CL_ϵ-term by setting:

$$[\![\epsilon]\!]_\rho \;\; = \;\; \epsilon$$

If $\langle D, \cdot, k, s, \epsilon\rangle$ is a λ-model and ρ a value assignment, it is also possible to define the meaning of an untyped λ-term in D:

$$\begin{array}{rcll} [\![x]\!]_\rho & = & \rho(x) & \\ [\![M_1 M_2]\!]_\rho & = & [\![M_1]\!]_\rho \cdot [\![M_2]\!]_\rho & \\ [\![\lambda x.M]\!]_\rho & = & \epsilon \cdot d & \textit{where } d \textit{ is such that } d \cdot e = [\![M]\!]_{\rho[e/x]} \textit{ for all } e \end{array}$$

The d required in the definition of the semantics of λ-abstraction always exists. In fact, it is a well-known result about combinatory logic that abstraction is internally definable. This means that we can associate a CL-term $[x].M$ with every CL-term M so that for all CL-terms N:

$$([x].M)N \rhd_{CL} M[N/x] \qquad (1)$$

where $\rhd_{CL}$ is the reduction relation on CL-terms obtained by orienting the two equations for k and s from left to right. This applies also to CL_ϵ-terms. There exist several definitions of $[x].M$, guaranteeing (1), all by induction on M. Here is one, which we shall call the naive one:

- $[x].y \equiv ky$ if $x \neq y$
- $[x].x \equiv skk$
- $[x].(M_1 M_2) \equiv s([x].M_1)([x].M_2)$

Proposition 3.3 *The naive abstraction algorithm satisfies the following properties for all M, ρ, e:*

1. $[\![[x].M]\!]_\rho \cdot e = [\![M]\!]_{\rho[e/x]}$
2. $[\![([x].M)N]\!]_\rho = [\![M[N/x]]\!]_\rho$

This definition of abstraction enables us to translate λ-terms to CL_ϵ-terms. Given a λ-term M, we denote its translation by $(M)_{CL_\epsilon}$, and define it by:

- $(x)_{CL_\epsilon} \equiv x$
- $(M_1M_2)_{CL_\epsilon} \equiv (M_1)_{CL_\epsilon}(M_2)_{CL_\epsilon}$
- $(\lambda x.M)_{CL_\epsilon} \equiv \epsilon([x].(M)_{CL_\epsilon})$

Using this translation, we can give an explicit definition of the semantics of λ-abstraction:

$$[\![\lambda x.M]\!]_\rho = \epsilon \cdot d \qquad where \;\; d = [\![[x].(M)_{CL_\epsilon}]\!]_\rho$$

The combinator ϵ makes the meaning of $\lambda x.M$ independent of the specific choice of d. This independence is crucial for establishing the validity of the ξ-rule, which says that if $[\![M]\!] = [\![N]\!]$ then $[\![\lambda x.M]\!] = [\![\lambda x.N]\!]$. The validity of ξ follows from the axiom (Weak Ext).

If we have only a combinatory algebra D we can still define an interpretation of the untyped λ-calculus. This requires a little modification of the translation since now we translate λ-terms to CL-terms. In this case we denote the translation of M by $(M)_{CL}$. The only difference with respect to the previous translation is the case of λ-abstraction, which becomes:

$$(\lambda x.M)_{CL} \equiv [x].(M)_{CL}$$

The meaning of λ-terms is :

$$[\![M]\!]_\rho \equiv [\![(M)_{CL}]\!]_\rho$$

It follows that:

$$\begin{array}{rcl} [\![x]\!]_\rho & = & \rho(x) \\ [\![M_1M_2]\!]_\rho & = & [\![M_1]\!]_\rho \cdot [\![M_2]\!]_\rho \\ [\![\lambda x.M]\!]_\rho \cdot e & = & [\![M]\!]_{\rho[e/x]} \;\; for\ all\ e \end{array}$$

It is easy to check that this translation validates the β-axiom only in a weak sense, that is, given two λ-terms M, N we have that:

$$((\lambda x.M)N)_{CL} \triangleright_{CL} M_{CL}[N_{CL}/x]$$

while, in order to validate the β-axiom we need:

$$((\lambda x.M)N)_{CL} \triangleright_{CL} (M[N/x])_{CL}$$

This is due to the fact that with the naive abstraction algorithm the CL-translation does not commute with substitution, therefore $M_{CL}[N_{CL}/x]$ is not equivalent (in CL) to $(M[N/x])_{CL}$, as shown by the following example. Take $M = \lambda x.y$ and $N = zz$; we have:

$$\begin{array}{rcl} M_{CL}[N_{CL}/y] & = & k(zz) \\ (M[N/y])_{CL} & = & s(kz)(kz) \end{array}$$

and we cannot prove their equality using the rules for CL. The following modification of the abstraction algorithm repairs this: replace the clause $[x].y \equiv ky$ if $x \neq y$ by

$$[x].M \equiv kM \text{ if } x \notin M$$

We call this algorithm the standard one. But the naive algorithm is enough for our purposes: the two terms $M_{CL}[N_{CL}/x]$ and $(M[N/x])_{CL}$ have the same extensional behaviour, that is, when applied to any other *CL*-term they produce the same result: see lemma 4.23.

The difference between the λ-model interpretation and the combinatory-algebra interpretation regards the ξ-rule, which is not valid in arbitrary combinatory algebras (regardless of the choice of the underlying abstraction algorithm).

If the structure D is only a partial combinatory algebra, we cannot define the meaning of general λ-terms. However, we can still define the meaning of some terms, useful in later results.

Lemma 3.4 *For every λ-term M and variable x, the semantics of $(\lambda x.M)_{CL}$ is defined in every partial combinatory algebra.*

Proof. First note that for all *CL*-terms N, if the semantics of N is defined then that of $[x].N$ is also defined, for all x. This can be checked easily by induction on N.

We prove the claim by induction on M. In the case that M is a variable, the claim is obvious. If $M = NP$ then we can apply the induction hypothesis to N and P, obtaining that the semantics of $[x].(N)_{CL}$ and $[x].(P)_{CL}$ are both defined and therefore also that of $(\lambda x.M)_{CL} = s([x].(N)_{CL})([x].(P)_{CL})$ is defined. If $M = (\lambda y.N)$ then we can apply the induction hypothesis to N obtaining that the semantics of $(\lambda y.N)_{CL} = [y].(N)_{CL}$ is defined; using our initial remark, we have that the semantics of $(\lambda x.M)_{CL} = [x].([y].(N)_{CL})$ is defined. ∎

The above lemma fails if the standard abstraction algorithm is used instead of the naive one.

3.2 Categories PER and SAT

Starting from an arbitrary partial combinatory algebra, we define categories of pers and categories of relations.

Definition 3.5 *The category* **PER** *of partial equivalence relations (pers) on a partial combinatory algebra D is defined by:*

- *$A \in Obj(\mathbf{PER})$ iff it is a symmetric and transitive partial relation on D; its domain of definition $\{a : aAa\}$ is denoted $dom(A)$ and the partial quotient of D by A is denoted $Q(A)$, that is, $Q(A)$ is the set of equivalence classes $\{[a]_A | a \in dom(A)\}$;*
- *a morphism $(f : A \to B) \in Mor(\mathbf{PER})$ is a function $f : Q(A) \to Q(B)$ such that there is an element $n \in D$ that computes f, or more formally:*

$$\textit{for all } a \in dom(A) : f([a]_A) = [n \cdot a]_B$$

We say that n is a realizer of f, and write $n \vdash f$.

Definition 3.6 *The category* **SAT** *of saturated relations on a partial combinatory algebra D is defined by:*

- *a saturated relation* $(R : A \nrightarrow B) \in \mathit{Obj}(\mathbf{SAT})$ *is given by two pers A and B over D and a relation $R \subseteq dom(A) \times dom(B)$ such that $R = A; R; B$ or, equivalently:*
  - *R is a relation between $dom(A)$ and $dom(B)$, that is, if aRb then aAa and bBb;*
  - *R is saturated, that is, if aAb, bRc, and cBd, then aRd.*
- *a morphism* $f : (R : A' \nrightarrow A'') \to (S : B' \nrightarrow B'') \in \mathit{Mor}(\mathbf{SAT})$ *between two saturated relations consists of a couple $(f' : A' \to B', f'' : A'' \to B'')$ of per morphisms such that:*

  $$\textit{for all } a \in dom(A'),\ b \in dom(A''),\ \textit{ if } aRb \textit{ then } f'(a)Sf''(b)$$

  If $n \vdash f'$ and $m \vdash f''$, we say that (n, m) is a realizer of f, and write $(n, m) \vdash f$.

We often write R for $R : A \nrightarrow B$, and sometimes refer to a saturated relation simply as a relation. We call A and B the domain and the codomain of $R : A \nrightarrow B$, respectively, and write $A = dom(R)$ and $B = cod(R)$. More generally, given a tuple $\overline{S}$ of saturated relations, we write $dom(\overline{S})$ and $cod(\overline{S})$ for the tuples of domains and codomains of the relations in $\overline{S}$.

The saturation property allows us to see a saturated relation indistinctly as a relation between equivalence classes or as a relation between elements of such classes.

Moreover, since any per can be seen as the identity saturated relation on itself, and every morphism between pers (seen as saturated relations) in **SAT** must be a couple of equal **PER** morphisms, we find **PER** as a full subcategory of **SAT**.

3.3 Constructions in SAT

In this section we generalize the usual categorical constructions of product ($\times$), exponentiation ($\Rightarrow$), and intersection ($\bigcap$) from **PER** to **SAT**.

Products do not appear explicitly in any of our formal systems, but they are part of the necessary categorical structure. In order to define products, we encode the pairing and the projection functions in D. The pairing function $[,] : D \times D \to D$ is the function realized by the interpretation of $\lambda z.\lambda x.\lambda x.zxy$. The projection functions $(\text{-})_1, (\text{-})_2 : D \to D$ are realized by the interpretations of $\lambda z.z(\lambda x.\lambda y.x)$ and $\lambda z.z(\lambda x.\lambda y.y)$, respectively. (By lemma 3.4 their semantics is always defined even in a partial combinatory algebra.) It is immediate to see that $D \times D$ is a retract of D via these two functions, that is, for all $a, b \in D$:

$$\begin{aligned} [a, b]_1 &= a \\ [a, b]_2 &= b \end{aligned}$$

We are interested in functions on relations $F : \mathit{Obj}(\mathbf{SAT})^k \to \mathit{Obj}(\mathbf{SAT})$ with the property that

(SAT-FUNC) if $\overline{S} : \overline{A} \nrightarrow \overline{B}$ then $F(\overline{S}) : F(\overline{A}) \nrightarrow F(\overline{B})$

Note that $F(\overline{S}) : F(\overline{A}) \nrightarrow F(\overline{B})$ implies that $F(\overline{A})$ and $F(\overline{B})$ are pers. We stipulate that a function $F : \mathit{Obj}(\mathbf{SAT})^0 \to \mathit{Obj}(\mathbf{SAT})$ satisfies (SAT-FUNC) iff it is constantly equal to a per.

Next we describe how to define functions on relations by product, exponentiation, and intersection with parameters. All of these constructions preserve property (SAT-FUNC).

Recall the definition of product and exponentiation in **PER**:

$$\begin{array}{rl} a(A \times B)b & \text{iff} \quad a_1 A b_1 \text{ and } a_2 B b_2 \\ a(A \Rightarrow B)b & \text{iff} \quad \text{for all } c, d, \text{ if } cAd \text{ then } (a \cdot c)B(b \cdot d) \end{array}$$

Product and exponentiation in **SAT** are similar:

Definition 3.7 (Product) *Given $R : A \nrightarrow B$ and $S : C \nrightarrow D$, their product $R \times S : (A \times C) \nrightarrow (B \times D)$ is defined by:*

$$a(R \times S)b \quad \textit{iff} \quad \begin{cases} a(A \times C)a \\ a_1 R b_1 \textit{ and } a_2 S b_2 \\ b(B \times D)b \end{cases}$$

Definition 3.8 (Exponentiation) *Given $R : A \nrightarrow B$ and $S : C \nrightarrow D$ their exponentiation $R \Rightarrow S : (A \Rightarrow C) \nrightarrow (B \Rightarrow D)$ is defined by:*

$$a(R \Rightarrow S)b \quad \textit{iff} \quad \begin{cases} a(A \Rightarrow C)a \\ \textit{for all } c, d, \textit{if } cRd \textit{ then } (a \cdot c)S(b \cdot d) \\ b(B \Rightarrow D)b \end{cases}$$

Remark 3.9 *It is easy to prove that the previous constructions $\times$ and $\rightarrow$ are the categorical product and exponentiation in* **SAT**. *Moreover these constructions are extensions of those defined in* **PER**.

Like pers, saturated relations also support infinite intersections:

Definition 3.10 (Intersection with Parameters) *Given a function*

$$F : \mathit{Obj}(\mathbf{SAT})^{k+1} \rightarrow \mathit{Obj}(\mathbf{SAT})$$

with property (SAT-FUNC), we define the intersection of F on its last argument

$$\bigcap_{R:A \nrightarrow B}^{SAT} F[-, R] : \mathit{Obj}(\mathbf{SAT})^k \rightarrow \mathit{Obj}(\mathbf{SAT})$$

with

$$\left(\bigcap_{R:A \nrightarrow B}^{SAT} F[-, R]\right)(\overline{S}) : \left(\bigcap_{R:A \nrightarrow B} F(dom(\overline{S}), R)\right) \nrightarrow \left(\bigcap_{R:A \nrightarrow B} F(cod(\overline{S}), R)\right)$$

by:

$$a\left(\left(\bigcap_{R:A \nrightarrow B}^{SAT} F[-, R]\right)(\overline{S})\right)b \quad \textit{iff} \quad \begin{cases} a\left(\bigcap_{R:A \nrightarrow B} F(dom(\overline{S}), R)\right)a \\ a\left(\bigcap_{R:A \nrightarrow B} F(\overline{S}, R)\right)b \\ b\left(\bigcap_{R:A \nrightarrow B} F(cod(\overline{S}), R)\right)b \end{cases}$$

for all $\overline{S} : \overline{A} \nrightarrow \overline{B} \in \mathit{Obj}(\mathbf{SAT})^k$.

Note that in the right-hand side of the previous definition we used set-theoretical intersection. In section 4.2 we show that infinite intersections are categorical products in **SAT**, as in **PER**. In the meantime, we prove that our definition is proper:

Proposition 3.11 *Intersection with parameters is well defined, that is, it maps tuples of saturated relations to saturated relations. Moreover it satisfies (SAT-FUNC), that is, if* $\overline{S} : \overline{A} \nrightarrow \overline{B}$ *then*

$$\left(\bigcap_{R:A\nrightarrow B}^{SAT} F[-,R](\overline{S}) \right) : \left(\bigcap_{R:A\nrightarrow B}^{SAT} F[-,R](\overline{A}) \right) \nrightarrow \left(\bigcap_{R:A\nrightarrow B}^{SAT} F[-,R](\overline{B}) \right)$$

Proof. We check that the intersection is well-defined. Let G stand for

$$\bigcap_{R:A\nrightarrow B}^{SAT} F[-,R]$$

Assume that $\overline{S} : \overline{A} \nrightarrow \overline{B}$ is a k-tuple of saturated relations. We have to prove that

$$\bigcap_{R:A\nrightarrow B} F(dom(\overline{S}),R) \quad \text{and} \quad \bigcap_{R:A\nrightarrow B} F(cod(\overline{S}),R)$$

are pers, that

$$\bigcap_{R:A\nrightarrow B} F(dom(\overline{S}),R) = G(\overline{A}) \quad \text{and} \quad \bigcap_{R:A\nrightarrow B} F(cod(\overline{S}),R) = G(\overline{B})$$

and that $G(\overline{S})$ is saturated.

- Symmetry of domain and codomain: Suppose

  $$a \left(\bigcap_{R:A\nrightarrow B} F(dom(\overline{S}),R) \right) b$$

  This means that for all $R : A \nrightarrow B$, $aF(dom(\overline{S}),R)b$. We have to prove that

  $$b \left(\bigcap_{R:A\nrightarrow B} F(dom(\overline{S}),R) \right) a$$

  that is, $bF(dom(\overline{S}),R)a$ for all $R : A \nrightarrow B$. By (SAT-FUNC) we know that:

  $$F(dom(\overline{S}),R) : F(dom(\overline{S}),dom(R)) \nrightarrow F(dom(\overline{S}),cod(R))$$

  If we instantiate the hypothesis with $dom(R)$ and $cod(R)$ we obtain:

  $$aF(dom(\overline{S}),dom(R))b \quad \text{and} \quad aF(dom(\overline{S}),cod(R))b$$

  By hypothesis, both $F(dom(\overline{S}),dom(R))$ and $F(dom(\overline{S}),cod(R))$ are pers, therefore by their symmetry we have:

  $$bF(dom(\overline{S}),dom(R))a \quad \text{and} \quad bF(dom(\overline{S}),cod(R))a$$

  Finally, we can use the saturation of $F(dom(\overline{S}),R)$ to conclude

  $$bF(dom(\overline{S}),R)a$$

  for all $R : A \nrightarrow B$.

  The proof for the codomain is similar.

- Transitivity of domain and codomain: Suppose that

$$a \left(\bigcap_{R:A \nrightarrow B} F(dom(\overline{S}), R) \right) b \left(\bigcap_{R:A \nrightarrow B} F(dom(\overline{S}), R) \right) c$$

  We have to prove that

$$a \left(\bigcap_{R:A \nrightarrow B} F(dom(\overline{S}), R) \right) c$$

  that is, $aF(dom(\overline{S}), R)c$ for all $R : A \nrightarrow B$. As before, we have by (SAT-FUNC):

$$F(dom(\overline{S}), R) : F(dom(\overline{S}), dom(R)) \nrightarrow F(dom(\overline{S}), cod(R))$$

  From the first assumption we know that $aF(dom(\overline{S}), R)b$ for all $R : A \nrightarrow B$, and instantiating the second assumption with $cod(R)$ we have $bF(dom(\overline{S}), cod(R))c$. By the saturation of $F(dom(\overline{S}), R)$ we obtain $aF(dom(\overline{S}), R)c$.

  Again, the proof for the codomain is similar.

- Correct domain and codomain: By definition we know that

$$G(\overline{A}) = \bigcap_{R:A \nrightarrow B} F(\overline{A}, R)$$

$$G(\overline{B}) = \bigcap_{R:A \nrightarrow B} F(\overline{B}, R)$$

  Therefore, we can conclude

$$G(\overline{A}) = \bigcap_{R:A \nrightarrow B} F(dom(\overline{S}), R)$$

$$G(\overline{B}) = \bigcap_{R:A \nrightarrow B} F(cod(\overline{S}), R)$$

  Now suppose that $aG(\overline{S})b$. We have to prove that $aG(\overline{A})a$ and $bG(\overline{B})b$. Both of these properties are obvious by expanding the definition of G.

- Saturation: Suppose that

$$a \left(\bigcap_{R:A \nrightarrow B} F(dom(\overline{S}), R) \right) bG(\overline{S})c \left(\bigcap_{R:A \nrightarrow B} F(cod(\overline{S}), R) \right) d$$

  We have to prove that $aG(\overline{S})d$. Expanding the definition of G, we find the conditions:

  - $a \left(\bigcap_{R:A \nrightarrow B} F(dom(\overline{S}), R) \right) a$: This follows from

$$a \left(\bigcap_{R:A \nrightarrow B} F(dom(\overline{S}), R) \right) b$$

    by symmetry and transitivity.

– $d\left(\bigcap_{R:A\nrightarrow B} F(cod(\overline{S}),R)\right) d$: This is proved similarly.

– $a\left(\bigcap_{R:A\nrightarrow B} F(\overline{S},R)\right) d$: We have to prove that $aF(\overline{S},R)d$ for all $R : A \nrightarrow B$. Instantiating the first, the second, and the third assumptions with $dom(R)$, R, and $cod(R)$, respectively, we obtain: $aF(dom(\overline{S}), dom(R))b$, $bF(\overline{S},R)c$, and $cF(cod(\overline{S}), cod(R))d$. We derive $aF(\overline{S},R)d$ using the saturation of $F(\overline{S},R)$.

■

We now have the ingredients for our parametric interpretation of systems F and $\mathcal{R}$, described in sections 4 and 5, respectively.

4 Parametric semantics of system F

We define an interpretation of system F, which we call the parametric per models of F, or **SAT** models of F.

In the first part of this section, we define the interpretation concretely, by directly interpreting the judgements of the syntax of F in **SAT**. This is a reconstruction of the one sketched in [BFSS90]. The second part of the section gives a more general categorical description of the **SAT** model. The main difference in outcome is that the second construction requires less of the underlying structure—namely, only a partial combinatory algebra is needed.

4.1 Concrete Models

Next we give a concrete definition for a model of F. Throughout this subsection, we assume that $\langle D, \cdot, k, s\rangle$ is at least a total combinatory algebra.

4.1.1 Types

In the parametric per model, quantification over types is understood as quantification over all relations. In order to interpret the quantification on a type variable as a quantification over all relations, we must be able to instantiate the variable with arbitrary relations. Therefore, we must be able to provide the semantics of types in a general context where type variables are interpreted as relations.

We denote by *TypeExp* the syntactic category of type expressions of F, obtained from type variables by the $\rightarrow$ and $\forall$ constructors. We interpret type expressions by a function

$$[\![-]\!] : \mathit{TypeExp} \rightarrow \mathit{RelAssign} \rightarrow \mathit{Obj}(\mathbf{SAT})$$

where *RelAssign* is the domain of (total) *relation assignments* which map type variables to relations. If a relation assignment maps every type variable to a per, then we call it a *per assignment*. This subset of *RelAssign* will be denoted by *PerAssign*. Typically, ε

denotes a relation assignment and η a per assignment. The operators $\uparrow$ and $\downarrow$ transform an arbitrary relation assignment ε into a per assignment:

$$\begin{aligned} \varepsilon \uparrow (X) &= dom(\varepsilon(X)) \\ \varepsilon \downarrow (X) &= cod(\varepsilon(X)) \end{aligned}$$

The interpretation is:

Definition 4.12 (Semantics of Type Expressions) *The semantics of type expressions is defined by:*

- $[\![X]\!]_\varepsilon = \varepsilon(X)$
- $[\![\sigma_1 \to \sigma_2]\!]_\varepsilon = [\![\sigma_1]\!]_\varepsilon \Rightarrow [\![\sigma_2]\!]_\varepsilon$
- $[\![\forall(X)\sigma]\!]_\varepsilon = \bigcap^{SAT}_{R:A \nrightarrow B} G(R)$

  where $G : Obj(\mathbf{SAT}) \to Obj(\mathbf{SAT})$ *is defined by* $G(R) \equiv [\![\sigma]\!]_{\varepsilon[R/X]}$.

Proposition 4.13 *The previous definition is proper, that is:*

1. *for all type expressions σ and all per environments η:*

$$[\![\sigma]\!]_\eta \in Obj(\mathbf{PER})$$

2. *for all type expressions σ and all relation environments ε:*

$$[\![\sigma]\!]_\varepsilon : [\![\sigma]\!]_{\varepsilon\uparrow} \nrightarrow [\![\sigma]\!]_{\varepsilon\downarrow}$$

This proposition easily follows from remark 3.9 and proposition 3.11. Part 1 amounts to what is known in the literature as the Identity Extension Lemma. It is stated in [Rey83] for an hypothetical set-theoretic semantics and is proved in [BFSS90] for the **SAT** model.

We close this section with a standard substitution lemma:

Lemma 4.14 (Substitution Lemma for Types) *For all type expressions σ and τ and all relation assignments ε:*

$$[\![\sigma[\tau/X]]\!]_\varepsilon = [\![\sigma]\!]_{\varepsilon[[\![\tau]\!]_\varepsilon/X]}$$

Proof. By a straightforward induction on σ. ■

4.1.2 Typings

In what follows, we consider mostly derivable judgements, and often write judgement instead of derivable judgement. We denote by *TypingJudge* the syntactical category of typing judgements derivable in F. We interpret typing judgements by a function

$$[\![-]\!] : TypingJudge \to PerAssign \to ValAssign \to \bigcup_{A \in Obj(PER)} Q(A)$$

where *ValAssign* is the domain of *value assignments*, which map each term variable to an equivalence class of some per:

$$\mathit{ValVar} \to \bigcup_{A \in Obj(PER)} Q(A)$$

and *ValVar* is the collection of all term variables. Typically ρ denotes a value assignment.

Definition 4.15

- *A value assignment ρ satisfies an environment E with respect to a per assignment η iff, for all $(x : \sigma) \in E$, $\rho(x) \in Q(\llbracket\sigma\rrbracket_\eta)$. This is denoted $\rho \models_\eta E$.*
- *Two value assignments ρ_1 and ρ_2 satisfy an environment E with respect to a relation assignment ε iff, for all $(x : \sigma) \in E$, $\rho_1(x)\ \llbracket\sigma\rrbracket_\varepsilon\ \rho_2(x)$. This is denoted $\rho_1, \rho_2 \models_\varepsilon E$.*

Proposition 4.16 *The assignments enjoys the properties:*

(a) $\eta\uparrow = \eta$, $\eta\downarrow = \eta$,

(b) if $\rho_1, \rho_2 \models_\varepsilon E$ *then* $\rho_1 \models_{\varepsilon\uparrow} E$ *and* $\rho_2 \models_{\varepsilon\downarrow} E$,

(c) if $\rho_1, \rho_2 \models_\eta E$ *then* $\rho_1 \models_\eta E$ *and* $\rho_2 \models_\eta E$,

(d) if $\rho \models_\eta E$ *then* $\rho, \rho \models_\eta E$.

Definition 4.17 *Given a value assignment ρ, we can obtain from it a new (untyped) value assignment $\rho^\star$ such that $\rho^\star(x) \in \rho(x)$, that is, $\rho^\star$ maps every term variable x to an (arbitrary) element of the equivalence class $\rho(x)$.*

The *erase* function associates to every typed term M the untyped term *erase*(M) obtained by erasing all type decorations present in M:

$$\begin{array}{rcl} erase(x) & = & x \\ erase(\lambda(x:\sigma).M) & = & \lambda x.erase(M) \\ erase(MN) & = & erase(M)erase(N) \\ erase(\lambda(X).M) & = & erase(M) \\ erase(M\sigma) & = & erase(M) \end{array}$$

What follows is a version in our relational setting of a well-known semantical property of erasures of typed terms in **PER** models (see [Mit90, CL91]). It is usually stated for **PER** models over a λ-model D, but the proof actually uses only the fact that the partial combinatory algebra structure is total: the totality is used to make sure that the interpretation of untyped terms in D is always defined.

Theorem 4.18 *For all typing judgements $E \vdash_F M : \sigma$ and all assignments η and ρ such that $\rho \models_\eta E$,*

$$\left(\llbracket erase(M)\rrbracket_{\rho^\star}\right) \llbracket\sigma\rrbracket_\eta \left(\llbracket erase(M)\rrbracket^D_{\rho^\star}\right)$$

Moreover, this is independent from the specific choice of $\rho^\star$.

To prove this result we need a more general statement.

Theorem 4.19 (Abstraction Theorem for Erasures) *For all typing judgements $E \vdash_F M : \sigma$ and all assignments ε, ρ_1, and ρ_2 such that $\rho_1, \rho_2 \models_\varepsilon E$:*

$$\left(\llbracket erase(M)\rrbracket_{\rho_1^\star}\right) \llbracket\sigma\rrbracket_\varepsilon \left(\llbracket erase(M)\rrbracket_{\rho_2^\star}\right)$$

Moreover, this is independent from the specific choice of $\rho_1^\star$ and $\rho_2^\star$.

Proof. We prove this statement by induction on the derivation of the judgement.

- Case (Val x): This case follows directly from the hypothesis on the assignments.
- Case (Val Fun): We have to prove:

  $$\left(\llbracket erase(\lambda(x:\sigma).M)\rrbracket_{\rho_1^\star}\right) \llbracket\sigma \to \tau\rrbracket_\varepsilon \left(\llbracket erase(\lambda(x:\sigma).M)\rrbracket_{\rho_2^\star}\right)$$

  This is equivalent to proving:

  (i) $\llbracket \lambda x.erase(M)\rrbracket_{\rho_1^\star} \in dom(\llbracket\sigma\rrbracket_{\varepsilon\uparrow} \Rightarrow \llbracket\tau\rrbracket_{\varepsilon\uparrow})$
  (ii) if $a, b \in D$ and $a \llbracket\sigma\rrbracket_\varepsilon b$
  then $(\llbracket \lambda x.erase(M)\rrbracket_{\rho_1^\star} \cdot a) \llbracket\tau\rrbracket_\varepsilon (\llbracket \lambda x.erase(M)\rrbracket_{\rho_2^\star} \cdot b)$
  (iii) $\llbracket \lambda x.erase(M)\rrbracket_{\rho_2^\star} \in dom(\llbracket\sigma\rrbracket_{\varepsilon\downarrow} \Rightarrow \llbracket\tau\rrbracket_{\varepsilon\downarrow})$

  These three statement are instances of the claim: for all $a, b \in D$ and all $\widetilde{\varepsilon}$, $\widetilde{\rho_1}$, and $\widetilde{\rho_2}$,

  $$\text{if } a \llbracket\sigma\rrbracket_{\widetilde{\varepsilon}} b \text{ then } (\llbracket \lambda x.erase(M)\rrbracket_{\widetilde{\rho_1}^\star} \cdot a) \llbracket\tau\rrbracket_{\widetilde{\varepsilon}} (\llbracket \lambda x.erase(M)\rrbracket_{\widetilde{\rho_2}^\star} \cdot b) \qquad (2)$$

  provided

  $$\widetilde{\rho_1}, \widetilde{\rho_2} \models_{\widetilde{\varepsilon}} E \qquad (3)$$

  The three different instantiations are:

  $$(\tilde{\varepsilon}, \widetilde{\rho_1}, \widetilde{\rho_2}) = \begin{cases} (\varepsilon\uparrow, \rho_1, \rho_1) \\ (\varepsilon, \rho_1, \rho_2) \\ (\varepsilon\downarrow, \rho_2, \rho_2) \end{cases}$$

  and in each case (3) easily follows from the hypothesis on the assignments and their properties (proposition 4.16).

  Next we prove the more general claim. From the definition of the semantics of abstraction for untyped terms, we know:

  $$\llbracket \lambda x.erase(M)\rrbracket_{\widetilde{\rho_1}^\star} \cdot a = \llbracket erase(M)\rrbracket_{\widetilde{\rho_1}^\star[a/x]}$$

  and analogously for $\llbracket \lambda x.erase(M)\rrbracket_{\widetilde{\rho_2}^\star} \cdot b$. Now from (3) and the premise of (2) we have:

  $$\widetilde{\rho_1}\left[[a]_{\llbracket\sigma\rrbracket_{\widetilde{\varepsilon}\uparrow}}/x\right], \widetilde{\rho_2}\left[[b]_{\llbracket\sigma\rrbracket_{\widetilde{\varepsilon}\downarrow}}/x\right] \models_{\widetilde{\varepsilon}} E, x : \sigma$$

  At this point we can apply the induction hypothesis to the premise of the rule (Val Fun) to obtain the desired result.

- Case (Val Fun2): We have to prove:

$$\left(\llbracket erase(\lambda(X).M)\rrbracket_{\rho_1^\star}\right) \llbracket\forall(X).\sigma\rrbracket_\varepsilon \left(\llbracket erase(\lambda(X).M)\rrbracket_{\rho_2^\star}\right)$$

and this is equivalent to proving:

(i) for all $R : A \nrightarrow B$, $\left(\llbracket erase(M)\rrbracket_{\rho_1^\star}\right) \llbracket\sigma\rrbracket_{\varepsilon\uparrow[R/X]} \left(\llbracket erase(M)\rrbracket_{\rho_1^\star}\right)$

(ii) for all $R : A \nrightarrow B$, $\left(\llbracket erase(M)\rrbracket_{\rho_1^\star}\right) \llbracket\sigma\rrbracket_{\varepsilon[R/X]} \left(\llbracket erase(M)\rrbracket_{\rho_2^\star}\right)$

(iii) for all $R : A \nrightarrow B$, $\left(\llbracket erase(M)\rrbracket_{\rho_2^\star}\right) \llbracket\sigma\rrbracket_{\varepsilon\downarrow[R/X]} \left(\llbracket erase(M)\rrbracket_{\rho_2^\star}\right)$

since $erase(\lambda(X).M) = erase(M)$. As for the previous case these three statements are instances of

$$\text{for all } R : A \nrightarrow B,\ \left(\llbracket erase(M)\rrbracket_{\widetilde{\rho_1}^\star}\right) \llbracket\sigma\rrbracket_{\tilde{\varepsilon}[R/X]} \left(\llbracket erase(M)\rrbracket_{\widetilde{\rho_2}^\star}\right) \tag{4}$$

with

$$\widetilde{\rho_1}, \widetilde{\rho_2} \models_{\tilde{\varepsilon}} E \tag{5}$$

To prove this claim, we first observe that the premise of the rule (Val Fun2) implies that X does not appear in E, and therefore is not free in the type of any of the free term variables of $erase(M)$. Then (5) implies

$$\widetilde{\rho_1}, \widetilde{\rho_2} \models_{\tilde{\varepsilon}[R/X]} E, X \tag{6}$$

since $\llbracket\tau\rrbracket_{\tilde{\varepsilon}[R/X]} = \llbracket\tau\rrbracket_{\tilde{\varepsilon}}$ because $X \notin \tau$ for all $(x : \tau) \in E$. Now we can apply the induction hypothesis to the premise of the rule (Val Fun2) with (6) to obtain the desired result.

- Case (Val Appl): We have to prove:

$$\left(\llbracket erase(MN)\rrbracket_{\rho_1^\star}\right) \llbracket\tau\rrbracket_\varepsilon \left(\llbracket erase(MN)\rrbracket_{\rho_2^\star}\right)$$

but by induction hypothesis applied to the premises of the rule we know:

$$\left(\llbracket erase(M)\rrbracket_{\rho_1^\star}\right) \llbracket\sigma \to \tau\rrbracket_\varepsilon \left(\llbracket erase(M)\rrbracket_{\rho_2^\star}\right)$$

$$\left(\llbracket erase(N)\rrbracket_{\rho_1^\star}\right) \llbracket\sigma\rrbracket_\varepsilon \left(\llbracket erase(N)\rrbracket_{\rho_2^\star}\right)$$

and, since $erase(MN) = erase(M)erase(N)$, the result follows by the definition of exponentiation of relations.

- Case (Val Appl2): We have to prove:

$$\left(\llbracket erase(M\tau)\rrbracket_{\rho_1^\star}\right) \llbracket\sigma[\tau/X]\rrbracket_\varepsilon \left(\llbracket erase(M\tau)\rrbracket_{\rho_2^\star}\right)$$

From lemma 4.14 (Substitution Lemma for Types) we know that $\llbracket\sigma[\tau/X]\rrbracket_\varepsilon = \llbracket\sigma\rrbracket_{\varepsilon[\llbracket\tau\rrbracket_\varepsilon/X]}$. Since $erase(M\tau) = erase(M)$ we can apply the induction hypothesis to the premise of the rule (Val Appl2) to obtain that

$$\left(\llbracket erase(M)\rrbracket_{\rho_1^\star}\right) \llbracket\sigma\rrbracket_{\varepsilon[R/X]} \left(\llbracket erase(M)\rrbracket_{\rho_2^\star}\right)$$

for all $R : A \nrightarrow B$. The result follows when we instantiate R with $\llbracket\tau\rrbracket_\varepsilon$.

■

The previous theorem allows us to define the semantics of typed terms from the semantics of their erasures, provided that the **SAT** model is based on a total combinatory algebra:

Definition 4.20 (Semantics of Typing Judgements) *Given a per assignment η and a value assignment ρ such that $\rho \models_\eta E$, we can define the semantics of the judgement $E \vdash_F M : \sigma$ by:*

$$[\![E \vdash_F M : \sigma]\!]_{\eta\rho} = \Big[[\![erase(M)]\!]_{\rho^\star}\Big]_{[\![\sigma]\!]_\eta}$$

Using the Abstraction Theorem for Erasures it is easy to verify that the previous definition is proper. We obtain:

Corollary 4.21 (Abstraction Theorem) *For all typing judgements $E \vdash_F M : \sigma$ and all assignments ε, ρ_1, and ρ_2 such that $\rho_1, \rho_2 \models_\varepsilon E$:*

$$\left([\![E \vdash_F M : \sigma]\!]_{\varepsilon\uparrow\rho_1}\right) [\![\sigma]\!]_\varepsilon \left([\![E \vdash_F M : \sigma]\!]_{\varepsilon\downarrow\rho_2}\right)$$

To close this section, we prove a standard substitution lemma.

Lemma 4.22 (Substitution Lemma for Terms) *For all typing judgements*

$$E, \overline{X} \vdash_F M : \sigma$$

all type expressions $\overline{\tau}$ whose free variables are in the domain of E, and all assignments η and ρ such that $\rho \models_\eta E$, we have:

$$[\![E \vdash_F M[\overline{\tau}/\overline{X}] : \sigma[\overline{\tau}/\overline{X}]]\!]_{\eta\rho} = [\![E, \overline{X} \vdash_F M : \sigma]\!]_{\eta[\overline{[\![\tau]\!]_\eta}/\overline{X}]\rho}$$

Proof. We apply definition 4.20; then the statement is an immediate consequence of the lemma 4.14. ■

4.1.3 Equalities

As previously remarked at the end of section 3.1, the typed β-rule is validated by our model even if its untyped counterpart is not valid into the (untyped) model. The key property in order to prove this result is the following lemma which allows, in the typed case, the commutation between substitution and CL-translation.

Lemma 4.23 (Commutation) *For all terms M, N such that $E, x : \sigma \vdash_F M : \tau$ and $E \vdash_F N : \sigma$, and for all assignments η, ρ_1, ρ_2 such that $\rho_1, \rho_2 \models_\eta E$ we have:*

$$[\![erase(M)_{CL}[erase(N)_{CL}/x]]\!]_{\rho_1^\star} [\![\tau]\!]_\eta [\![(erase(M)[erase(N)/x])_{CL}]\!]_{\rho_2^\star}$$

Proof. By induction on the length of M. If M is a variable different from x, then we can easily conclude by the hypothesis on the assignments. If M is equal to x then the statement follows by the abstraction theorem 4.19. The case of application follows directly by the induction hypothesis.

If M is an abstraction, for example $M = \lambda y : \tau_1.P$ of type $\tau_1 \to \tau_2$, then we can apply the induction hypothesis to P, obtaining that for all a, b such that $a \llbracket \tau_1 \rrbracket_\eta \, b$:

$$\llbracket erase(P)_{CL}[erase(N)_{CL}/x] \rrbracket_{\rho_1^\star[a/y]} \llbracket \tau_2 \rrbracket_\eta \llbracket (erase(P)[erase(N)/x])_{CL} \rrbracket_{\rho_2^\star[b/y]} \tag{7}$$

We have to prove that

$$\llbracket ([y].erase(P)_{CL})[erase(N)_{CL}/x] \rrbracket_{\rho_1^\star} \llbracket \tau_1 \to \tau_2 \rrbracket_\eta \llbracket [y].(erase(P)[erase(N)/x])_{CL} \rrbracket_{\rho_2^\star} \tag{8}$$

On the other hand, by the substitution lemma 3.2, we have that

$$\llbracket ([y].erase(P)_{CL})[erase(N)_{CL}/x] \rrbracket_{\rho_1^\star} = \llbracket [y].erase(P)_{CL} \rrbracket_{\rho_1^\star[\llbracket erase(N)_{CL} \rrbracket_{\rho_1^\star}/x]} \tag{9}$$

In order to prove (8) we have to show that for all a, b such that $a \llbracket \tau_1 \rrbracket_\eta \, b$:

$$(\llbracket ([y].erase(P)_{CL})[erase(N)_{CL}/x] \rrbracket_{\rho_1^\star} \cdot a) \llbracket \tau_2 \rrbracket_\eta (\llbracket [y].(erase(P)[erase(N)/x])_{CL} \rrbracket_{\rho_2^\star} \cdot b)$$

and, using (9), this is equivalent to

$$(\llbracket [y].erase(P)_{CL} \rrbracket_{\rho_1^\star[\llbracket erase(N)_{CL} \rrbracket_{\rho_1^\star}/x]} \cdot a) \llbracket \tau_2 \rrbracket_\eta (\llbracket [y].(erase(P)[erase(N)/x])_{CL} \rrbracket_{\rho_2^\star} \cdot b)$$

which, by lemma 3.3 is equivalent to

$$\llbracket erase(P)_{CL} \rrbracket_{\rho_1^\star[\llbracket erase(N)_{CL} \rrbracket_{\rho_1^\star}, a/x, y]} \llbracket \tau_2 \rrbracket_\eta \llbracket (erase(P)[erase(N)/x])_{CL} \rrbracket_{\rho_2^\star[b/y]}$$

Now, using the substitution lemma 3.2 for the untyped semantics again (in the opposite direction), we have

$$\llbracket erase(P)_{CL}[erase(N)_{CL}/x] \rrbracket_{\rho_1^\star[a/y]} \llbracket \tau_2 \rrbracket_\eta \llbracket (erase(P)[erase(N)/x])_{CL} \rrbracket_{\rho_2^\star[b/y]}$$

At this point we are done since this is exactly (7), which has been previously shown.
■

Now we can prove the soundness of the **SAT** model with respect to the equality rules of F.

Theorem 4.24 (Soundness for F Equalities) *Given an equality judgement $E \vdash_F M = N : \sigma$, a per assignment η and a value assignment ρ such that $\rho \models_\eta E$, we have:*

$$\llbracket E \vdash_F M : \sigma \rrbracket_{\eta\rho} \llbracket \rho \rrbracket_\eta \llbracket E \vdash_F N : \sigma \rrbracket_{\eta\rho}$$

Proof. We only sketch the proof. If the combinatory algebra D is actually a λ-model, then the statement is obvious since the equality already holds in the untyped semantics:

$$\llbracket erase(M) \rrbracket_{\rho^\star} = \llbracket erase(N) \rrbracket_{\rho^\star}$$

The general case requires more care. We look only at (Val Eq Beta) and (Val Eq Fun), the latter corresponding to the ξ-rule of the untyped λ-calculus.

- Case (Val Eq Beta):

$$\begin{array}{rcl}
[\![E \vdash_F (\lambda(x:\sigma).M)N : \tau]\!]_{\eta\rho} & = & \text{(by def.)} \\
[\![([x].erase(M)_{CL})erase(N)_{CL}]\!]_{\rho^\star} & = & (\beta\text{-axiom for } CL) \\
[\![erase(M)_{CL}[erase(N)_{CL}/x]]\!]_{\rho^\star} & &
\end{array}$$

  Now using the commutation lemma 4.23, we have that

$$[\![erase(M)_{CL}[erase(N)_{CL}/x]]\!]_{\rho^\star}\, [\![\tau]\!]_\eta\, [\![(erase(M)[erase(N)/x])_{CL}]\!]_{\rho^\star}$$

  and we are done since

$$[\![E \vdash_F M[N/x] : \tau]\!]_{\eta\rho} = [\![(erase(M)[erase(N)/x])_{CL}]\!]_{\rho^\star}$$

- Case (Val Eq Fun):
  We have to prove

$$[\![\lambda x.erase(M)]\!]_{\rho^\star}\, [\![\sigma \to \tau]\!]_\eta\, [\![\lambda x.erase(N)]\!]_{\rho^\star}$$

  that is, for all a, b:

$$\text{if } a\, [\![\sigma]\!]_\eta\, b \text{ then } \left([\![\lambda x.erase(M)]\!]_{\rho^\star} \cdot a\right) [\![\tau]\!]_\eta \left([\![\lambda x.erase(N)]\!]_{\rho^\star} \cdot b\right)$$

  By proposition 3.3 we can rewrite this as

$$\text{if } a\, [\![\sigma]\!]_\eta\, b \text{ then } [\![erase(M)]\!]_{\rho^\star[a/x]}\, [\![\tau]\!]_\eta\, [\![erase(N)]\!]_{\rho^\star[b/x]}$$

  We conclude by applying the induction hypothesis and by transitivity, since by theorem 4.19 we have:

$$[\![erase(N)]\!]_{\rho^\star[a/x]}\, [\![\tau]\!]_\eta\, [\![erase(N)]\!]_{\rho^\star[b/x]}$$

■

What makes the difference between the untyped and the typed semantics is that while the representatives for λ-abstractions are not canonical they are nevertheless in the same equivalence class.

4.2 Categorical Models

In the previous subsection we have been forced to assume a total combinatory algebra in order to define the semantics of typed terms from the semantics of their erasures. In this section we overcome this limitation. We build a **SAT** model for system F starting from an arbitrary partial combinatory algebra. This will be possible by moving from an untyped semantics for typed terms (that is, a semantics based on erasures) to a typed semantics. The typed semantics is presented as a categorical model

Categorical models of system F are based on the quantifiers-as-adjoints paradigm, which goes back to Lawvere [Law69]; Seely has defined them under the name of PL-categories [See83, See87].

Next we review the definition of PL-category. Then we describe the model **SAT** as a PL-category. In the appendix we review how this is done for the usual **PER** model.

4.2.1 PL-categories

PL-categories are an algebraic generalization of the models of simply typed λ-calculus in a bi-dimensional universe of cartesian closed categories indexed over a global category. PL-categories are sometimes referred to as *external* models of F in contrast with the *internal* ones which use the internal category theory. We do not consider here the internal models and we refer the interested readers to [AM92], where detailed descriptions and relationships between the two kinds of models are provided.

We assume some acquaintance with the notion of indexed category and of categorical models of system F, but provide the main definitions.

The definition of external model is based on that of indexed category. A model is given via a contravariant functor $\mathbf{G}$ from a category $\mathbf{E}$ to $\mathbf{CCCat}$, the category of all (small) cartesian closed categories and cartesian closed functors between them. The category $\mathbf{E}$ is cartesian and has a distinguished object Ω, interpreting the collection of types. Products Ω^p are used to give meanings to environments E declaring p type variables. Types legal in E are interpreted by arrows in $\mathbf{E}[\Omega^p, \Omega]$. The functor $\mathbf{G} : \mathbf{E}^{op} \to \mathbf{CCCat}$ takes Ω^p in $\mathbf{E}$ to a (local) category $\mathbf{G}(\Omega^p)$ whose objects are the types legal in E. Thus, types appear both as arrows in $\mathbf{E}$ and as objects in the local category $\mathbf{G}(\Omega^p)$ and we require

$$Obj((\mathbf{G}(\Omega^p)) = \mathbf{E}[\Omega^p, \Omega]$$

The arrows of the local category $\mathbf{G}(\Omega^p)$ interpret the terms of system F whose free type variables are in E. Every local category is required to be a model of the simply typed λ-calculus, that is, a cartesian closed category. The abstraction on type variables is described as the right adjoint to the diagonal functor.

The following definition is obtained from the one presented in [AM92] by dropping the unnecessary requirement of the cartesian closedness of the global category, as done for example in [Cur89].

Definition 4.25 (PL-category) *A PL-category is a triple* $(\mathbf{E}, \mathbf{G}, \Omega)$ *such that:*

- $\mathbf{E}$ *is a cartesian category whose objects are* Ω *and its powers; we abbreviate the object* Ω^p *by* p *and the product of* Ω^p *and* Ω^q *by* $p + q$. *In particular* 1 *is* Ω.
- $\mathbf{G} : \mathbf{E}^{op} \to \mathbf{CCCat}$ *is a functor (indexed category) such that*
  - *(a) For each object* p *in* $\mathbf{E}$, $Obj(\mathbf{G}(p)) = \mathbf{E}[p, 1]$ *and for each morphism* $f \in \mathbf{E}[p, q]$, $\mathbf{G}(f)$ *acts on the objects of* $\mathbf{G}(q)$ *as the pre-composition with* f, *that is* $\mathbf{G}(f)(g) = g \circ f$.
  - *(b) For every* $f \in \mathbf{E}[p, q]$, *the functor* $\mathbf{G}(f) : \mathbf{G}(q) \to \mathbf{G}(p)$ *preserves the cartesian closed structure "on the nose" (and not just up to isomorphism); that is, for all* $h, l \in Obj(\mathbf{G}(q)) = \mathbf{E}[q, 1]$:
    - *(1)* $\mathbf{G}(f)(t_{\mathbf{G}(q)}) = t_{\mathbf{G}(p)}$ *where* $t_{\mathbf{G}(p)}$ *is the terminal object of* $\mathbf{G}(p)$
    - *(2)* $\mathbf{G}(f)(t_h) = t_{\mathbf{G}(f)(h)}$ *where* $t_h : h \to t_{\mathbf{G}(q)}$ *is the unique arrow from* h *to the terminal object of* $\mathbf{G}(q)$
    - *(3)* $\mathbf{G}(f)(h \times_{\mathbf{G}(q)} l) = \mathbf{G}(f)(h) \times_{\mathbf{G}(p)} \mathbf{G}(f)(l)$ *where* $\times_{\mathbf{G}(p)}$ *is the product in* $\mathbf{G}(p)$
    - *(4)* $\mathbf{G}(f)(fst_{h,l}) = fst_{\mathbf{G}(f)(h), \mathbf{G}(f)(l)}$

(5) $\mathbf{G}(f)(snd_{\,h,l}) = snd_{\,\mathbf{G}(f)(h),\mathbf{G}(f)(l)}$

(6) $\mathbf{G}(f)([h,l]_{\mathbf{G}(q)}) = [\mathbf{G}(f)(h),\mathbf{G}(f)(l)]_{\mathbf{G}(p)}$ *where* $[,]_{\mathbf{G}(p)}$ *is the exponentiation in* $\mathbf{G}(p)$

(7) $\mathbf{G}(f)(eval_{h,l}) = eval_{\mathbf{G}(f)(h),\mathbf{G}(f)(l)}$

(c) Let $\mathbf{G}^{\Omega}$ *and* $\mathbf{Fst}$ *be defined by:*

(1) $\mathbf{G}^{\Omega} : \mathbf{E}^{op} \to \mathbf{CCCat}$ *is the functor (indexed category) such that*

$$\begin{aligned} \mathbf{G}^{\Omega}(p) &= \mathbf{G}(p+1) \\ \mathbf{G}^{\Omega}(f) &= \mathbf{G}(f \times id) \end{aligned}$$

(2) $\mathbf{Fst}(p) = \mathbf{G}(fst_{\,p,1}) : \mathbf{G}^{\Omega}(p) \to \mathbf{G}(p)$

Then there exists an $\mathbf{E}$*-indexed adjunction* $\langle \mathbf{Fst}, \forall, \boldsymbol{\Delta} \rangle : \mathbf{G} \rightharpoonup \mathbf{G}^{\Omega}$*, that is, for all* $f \in \mathbf{E}[p,q]$*:*

* $\langle \mathbf{Fst}(p), \forall(p), \boldsymbol{\Delta}(p) \rangle : \mathbf{G}(p) \rightharpoonup \mathbf{G}^{\Omega}(p)$ *is an adjunction in the usual sense, and, moreover,*
* $\boldsymbol{\Delta}(p) \circ \mathbf{G}^{\Omega}(f) = \mathbf{G}(f) \circ \boldsymbol{\Delta}(q)$

The last equation in the definition expresses the naturality of the isomorphism $\boldsymbol{\Delta}$ with respect to the index p. Informally, it states the naturality for substitution under $\lambda(X)$:

$$[\![\lambda(X).(M[\tau/Y])]\!] = [\![(\lambda(X).M)[\tau/Y])]\!]$$

for all τ (modulo the obvious renamings of bound variables).

4.2.2 SAT as a PL-category

We now recast the **SAT** model as a PL-category. We follow a pattern of definitions that applies also to the **PER** model; we review the categorical presentation of the **PER** model in the appendix, for comparison.

Definition 4.26 (Global Category) *The objects of the global category* $\mathbf{E}$ *are the set Obj(*$\mathbf{SAT}$*) and its powers. The set of morphisms is defined in two steps: first we define*

$$\mathbf{E}[p,1] = \{F : Obj(\mathbf{SAT})^p \to Obj(\mathbf{SAT}) \mid F \textit{ satisfies } (SAT - FUNC)\}$$

and then

$$\mathbf{E}[p,q] = \mathbf{E}[p,1]^q$$

so the morphisms in $\mathbf{E}[p,q]$ *are q-tuples of morphisms in* $\mathbf{E}[p,1]$*.*

As usual the property $F(\overline{R}) : F(dom(\overline{R})) \nrightarrow F(cod(\overline{R}))$ implies that for all p-tuples of pers $\overline{A}$ we have $F(\overline{A}) \in Obj(\mathbf{PER})$.

Definition 4.27 (Indexed Category) *The collection of objects of the indexed category* $\mathbf{G}(p)$ *is* $\mathbf{E}[p,1]$*, by the definition of PL-categories. The morphisms in* $\mathbf{G}(p)$ *are the uniformly realized arrows between objects, that is, the arrows* $\Phi : F \to H$ *such that:*

* $F, H \in Obj(\mathbf{G}(p))$*;*
* $\Phi : p \to Mor(\mathbf{SAT})$ *and for all* $\overline{R} \in p$*,* $\Phi(\overline{R}) : F(\overline{R}) \to H(\overline{R})$ *in Mor(*$\mathbf{SAT}$*).*

- *There exists $n \in D$ that computes uniformly Φ in the sense that for all $\overline{R} \in p$, (n, n) realizes $\Phi(\overline{R})$ in Mor(**SAT**). We say that n is a realizer of $\Phi : F \to H$, and write $n \vdash (\Phi : F \to H)$.*

Given an $L \in \mathbf{E}[p, q]$, we define $\mathbf{G}(L)$ as the functor from $\mathbf{G}(q)$ to $\mathbf{G}(p)$ that acts on both objects and morphisms as the pre-composition with L. Namely, if F is an object of $\mathbf{G}(q)$, then $\mathbf{G}(L)(F) = F \circ L$, and if $\Phi : F \to H$ is a morphism of $\mathbf{G}(q)$ realized by n, then $\mathbf{G}(L)(\tau : F \to H)$ is the uniquely determined morphism, denoted $(\Phi \circ L) : (F \circ L) \to (H \circ L)$, realized by n.

*Product and exponentiation in the fibers $\mathbf{G}(p)$ are defined componentwise using those of **SAT** as follows. For all $F, G, H \in \mathbf{G}(p)$ and $\overline{R} \in p$:*

- $(F \times G)(\overline{R}) = F(\overline{R}) \times G(\overline{R})$
- $fst_{F,G}(\overline{R}) = fst_{F(\overline{R}),G(\overline{R})}$ *and* $snd_{F,G}(\overline{R}) = snd_{F(\overline{R}),G(\overline{R})}$
- *the terminal object is the constant function $T_p(\overline{R}) = t$ where t is the terminal object of* **SAT**
- $F^G(\overline{R}) = F(\overline{R})^{G(\overline{R})}$
- $eval_{F,G}(\overline{R}) = eval_{F(\overline{R}),G(\overline{R})}$
- $\Lambda(\Phi)(\overline{R}) = \Lambda(\Phi(\overline{R}))$ *for all* $\Phi : F \times G \to H$

It is easy to verify the existence of a uniform realizer for each one of the previous arrows in the fibers. For example consider $eval_{F,G}$ which, by the previous definition, is the family of arrows:

$$\{eval_{F(\overline{R}),G(\overline{R})} : F(\overline{R})^{G(\overline{R})} \times G(\overline{R}) \to F(\overline{R}) \mid \overline{R} \in p\}$$

Each member of this family is realized by the interpretation of $\lambda x.\lambda y.xy$ in the partial combinatory algebra D.

Note that $\mathbf{G}(0)$ is isomorphic to **PER**. This corresponds to the fact that closed type expressions are interpreted as pers.

We interpret the quantification functor $\forall$ on objects by the intersection operator introduced in definition 3.10.

Definition 4.28 (Indexed Adjunction) *Given an object L of $\mathbf{G}(p + 1)$, we define*

$$\forall(p)(L) = \bigcap_{R:A \twoheadrightarrow B}^{SAT} L[-, R]$$

The behavior of $\forall(p)$ on morphisms is that if n realizes the morphism $\Phi : F \to H$ in $\mathbf{G}(p+1)$ then $\forall(p)(\Phi : F \to H)$ is the arrow from $\forall(p)(F)$ to $\forall(p)(H)$ in $\mathbf{G}(p)$ realized by n. For every p, $F \in Obj(\mathbf{G}(p))$ and $H \in Obj(\mathbf{G}(p + 1))$, we define the isomorphism

$$\boldsymbol{\Delta}(p) : \mathbf{G}(p + 1)[\mathbf{Fst}(p)(F), H] \xrightarrow{\cong} \mathbf{G}(p)[F, \forall(p)(H)]$$

so that it sends a morphism $\Phi : \mathbf{Fst}(p)(F) \to H$ realized by n to the unique morphism from F to $\forall(p)(H)$ in $\mathbf{G}(p)$ realized by n.

Also in this case it is easy to verify that the morphism part of the quantification functor and the previous isomorphism are well defined. For the naturality of the isomorphism with respect to the global parameter, we have to check that for all morphisms $\Phi : \mathbf{Fst}(p)(F) \to H$ in $\mathbf{G}(p+1)$ and $L \in \mathbf{E}[q,p]$:

$$\boldsymbol{\Delta}(q)(\Phi \circ (L \times id)) = (\boldsymbol{\Delta}(p)(\Phi) \circ L) : (F \circ L) \to (\forall(q)(H) \circ L)$$

This equality is proved as follows: if n realizes $\Phi : \mathbf{Fst}(p)(F) \to H$ in $\mathbf{G}(p+1)$ then both sides of the previous equality are realized by n and so must be equal.

For any $F \in \mathbf{G}(p+1)$ we will denote by $\mathbf{Proj}^F(p)$ the counit of the indexed adjunction, that is, the following arrow in $\mathbf{G}(p+1)$:

$$\mathbf{Proj}^F(p) = \boldsymbol{\Delta}^{-1}(p)(id_{\ \forall(F)}) : \mathbf{Fst}(p) \circ \forall(p)(F) \to F$$

Since the presence of all such indexes makes the notation very heavy, from now on we will omit them when their values can be inferred from context.

4.2.3 Categorical interpretation of types and terms

Given a type judgement $E \vdash_F \sigma$ with exactly p type variables declared in E, we define the semantics of σ in E, denoted by $[\![\sigma]\!]_p$ as an object of the fiber $\mathbf{G}(p)$, that is, an arrow $Obj(\mathbf{SAT})^p \to Obj(\mathbf{SAT})$.

Definition 4.29 (Semantics of Types)

$$\begin{array}{rcl} [\![X]\!]_p & = & snd \circ fst^{\,i-1} \\ & & \textit{where } X \textit{ is the i-th type variable} \\ & & \textit{declared in } E \\ [\![\sigma \to \tau]\!]_p & = & [\![\tau]\!]_p^{[\![\sigma]\!]_p} \\ [\![\forall(X).\sigma]\!]_p & = & \forall([\![\sigma]\!]_{p+1}) \end{array}$$

We do the same for environments, so if $\vdash_F E$ is a correct environment expression, with p type variables declared in E, then its semantics, denoted by $[\![\vdash_F E]\!]_p$, will be an element of the fiber $\mathbf{G}(p)$, namely the product of the semantics of the types of the term variables declared in it.

Definition 4.30 (Semantics of Environments)

$$\begin{array}{rcl} [\![\vdash_F \emptyset]\!]_p & = & T_p \\ [\![\vdash_F E, X]\!]_p & = & [\![\vdash_F E]\!]_p \\ [\![\vdash_F E, x : \sigma]\!]_p & = & [\![\vdash_F E]\!]_p \times [\![\sigma]\!]_p \end{array}$$

Finally, we define the semantics of typing judgements $E \vdash_F M : \sigma$ with exactly p type variables declared in E, denoted by $[\![E \vdash_F M : \sigma]\!]_p$, as a morphism in the fiber $\mathbf{G}(p)$ from $[\![\vdash_F E]\!]_p$ to $[\![\sigma]\!]_p$.

Definition 4.31 (Semantics of Typing Judgements)

$$\begin{array}{rcl}
[\![E', x:\sigma, E'' \vdash_F x:\sigma]\!]_p & = & \mathit{snd} \circ \mathit{fst}^{\,n-1} \\
 & & \textit{where } x \textit{ is the } n\textit{-th term variable} \\
 & & \textit{declared in } E \\
[\![E \vdash_F \lambda(x:\sigma).M : \sigma \to \tau]\!]_p & = & \Lambda([\![E, x:\sigma \vdash_F M:\tau]\!]_p) \\
[\![E \vdash_F \lambda(X).M : \forall(X).\sigma]\!]_p & = & \boldsymbol{\Delta}([\![E, X \vdash_F M:\sigma]\!]_p) \\
[\![E \vdash_F MN : \tau]\!]_p & = & \mathit{eval} \circ \\
 & & < [\![E \vdash_F M:\sigma \to \tau]\!]_p , [\![E \vdash_F N:\tau]\!]_p > \\
[\![E \vdash_F M\tau : \sigma[\tau/X]]\!]_p & = & \mathbf{G}(<\mathit{id}\ , [\![\tau]\!]_p >)(\mathbf{Proj}^{[\![\sigma]\!]_{p+1}}) \circ \\
 & & [\![E \vdash_F M : \forall(X).\sigma]\!]_p
\end{array}$$

All the cases are straightforward, except the last one. The arrow

$$\mathbf{G}(<\mathit{id}\ , [\![\tau]\!]_p >)(\mathbf{Proj}^{[\![\sigma]\!]_p}) : [\![\forall(X).\sigma]\!]_p \to [\![\sigma[\tau/X]]\!]_p$$

instantiates the counit $\mathbf{Proj}^{[\![\sigma]\!]_p}$ of the adjunction to $[\![\tau]\!]_p$ which, in turn, is used to instantiate the semantics of the polymorphic term M

Remark 4.32 *With this kind of semantics the Abstraction Theorem comes "for free" since we have that*

$$[\![E \vdash_F M:\sigma]\!]_p : [\![\vdash_F E]\!]_p \to [\![\sigma]\!]_p$$

This means, in particular, that the semantics of M maps related values for its free term variables to related values.

5 Semantics of system $\mathcal{R}$

In this section we extend the first **SAT** model of F to a model of $\mathcal{R}$. We believe that an analogous extension is possible for the categorical **SAT** model of F. We prefer treating the first model because the structure of $\mathcal{R}$ complicates the notations for categorical models even further. The complications arise from the dependence of relation expressions upon term variables. Because of this dependence, the semantics of relation expressions is defined only under correct assignments to term variables; this is hard to express in a categorical style.

We rely on the constructions of the previous sections to define an interpretation of $\mathcal{R}$ in the **SAT** model.

5.1 Related types

Since in system $\mathcal{R}$ type and relation variables are both present explicitly, we define relation assignments on both type and relation variables in such a way that type variables are mapped to pers and relation variables are mapped to relations. The domain *RelAssign* of relation assignments becomes:

$$\mathit{RelVar} \cup \mathit{TypeVar} \to \mathit{Obj}(\mathbf{SAT})$$

We denote $\eta \updownarrow$ the restriction of a relation assignment η to the set of type variables. Value assignments remain the same as in system F.

We let *RelTypesJudge* denote the syntactic category of related types judgements, and interpret the related types judgements by a function:

$$[\![-]\!] : \mathit{RelTypesJudge} \to \mathit{RelAssign} \to \mathit{ValAssign} \to \mathit{Obj}(\mathbf{SAT})$$

Defining the semantics of relation expressions is a little more difficult than defining the semantics of type expressions. One technical reason for this is that relation expressions may contain term variables, so we are forced to make their meaning depend upon term variable assignments. Moreover, the presence of term variables inside relation expressions implies that not all relation expressions are meaningful, since we must add some hypothesis on the interpretation of term variables which, in turn, depend on the particular (syntactic) environment considered. In order to define the semantics of relation expressions, then, we would like to know in which environment we are working and that the relation expression is well-formed (that is, derivable) in this environment. For these reasons, it seems best to define the semantics of entire related types judgements, as we do next.

To interpret functional-relation expressions, we use an auxiliary function:

Definition 5.33 *Let FRel* : $\mathit{Mor}(\mathbf{PER}) \to \mathit{Obj}(\mathbf{SAT})$ *be the function that maps an* $f : A \to B$ *to the relation* $\mathit{FRel}(f) : A \nrightarrow B$ *such that:*

$$a(\mathit{FRel}(f))b \quad \textit{iff} \quad b \in f([a]_A)$$

Note that, by construction, $\mathit{FRel}(f)$ is always saturated.

Definition 5.34 *Given a derivable environment judgement* $\vdash_{\mathcal{R}} E$ *and a relation assignment* η, *we say that* η *is an assignment for* E *iff*

$$\textit{for all} \begin{pmatrix} X \\ \mathcal{W} \\ Y \end{pmatrix} \in E, \quad \eta(\mathcal{W}) : \eta(X) \nrightarrow \eta(Y)$$

Definition 5.35 *We define the satisfaction of an environment judgement by a relation assignment and a value assignment, and the meaning of a related types judgement, with a joint inductive definition:*

- *Given an environment judgement* $\vdash_{\mathcal{R}} E$, *a relation assignment* η *for it, and a value assignment* ρ, *we say that* ρ *satisfies* E *with respect to* η *iff*

$$\textit{for all} \begin{pmatrix} x : \sigma_1 \\ \mathcal{R} \\ y : \sigma_2 \end{pmatrix} \in E, \quad \rho(x) \left[\!\!\left[E' \vdash_{\mathcal{R}} \begin{matrix} \sigma_1 \\ \mathcal{R} \\ \sigma_2 \end{matrix} \right]\!\!\right]_{\eta\rho} \rho(y)$$

where E' *is such that* $E = E', \begin{matrix} x : \sigma_1 \\ \mathcal{R} \\ y : \sigma_2 \end{matrix}, E''$. *We write this* $\rho \models_\eta E$.

- *Given assignments η and ρ such that $\rho \models_\eta E$, we define the semantics of the related types judgement $E \vdash_{\mathcal{R}} \begin{matrix}\sigma_1\\ \mathcal{S}\\ \sigma_2\end{matrix}$ by induction on its derivation:*

$$\begin{array}{rcl}
\left[\!\left[E', \begin{matrix}X\\ \mathcal{W}\\ Y\end{matrix}, E'' \vdash_{\mathcal{R}} \begin{matrix}X\\ \mathcal{W}\\ Y\end{matrix} \right]\!\right]_{\eta\rho} & = & \eta(\mathcal{W}) \\
\left[\!\left[E', \begin{matrix}X\\ \mathcal{W}\\ Y\end{matrix}, E'' \vdash_{\mathcal{R}} X \right]\!\right]_{\eta\rho} & = & \eta(X) \\
\left[\!\left[E', \begin{matrix}X\\ \mathcal{W}\\ Y\end{matrix}, E'' \vdash_{\mathcal{R}} Y \right]\!\right]_{\eta\rho} & = & \eta(Y) \\
\left[\!\left[E \vdash_{\mathcal{R}} \begin{matrix}\sigma_1 \to \tau_1\\ \mathcal{S}_1 \to \mathcal{S}_2\\ \sigma_2 \to \tau_2\end{matrix} \right]\!\right]_{\eta\rho} & = & \left[\!\left[E \vdash_{\mathcal{R}} \begin{matrix}\sigma_1\\ \mathcal{S}_1\\ \sigma_2\end{matrix} \right]\!\right]_{\eta\rho} \Rightarrow \left[\!\left[E \vdash_{\mathcal{R}} \begin{matrix}\tau_1\\ \mathcal{S}_2\\ \tau_2\end{matrix} \right]\!\right]_{\eta\rho} \\
a \left[\!\left[E \vdash_{\mathcal{R}} \begin{matrix}\forall(X).\sigma_1\\ \forall(\mathcal{W}).\mathcal{S}\\ \forall(Y).\sigma_2\end{matrix} \right]\!\right]_{\eta\rho} b & \textit{iff} & \left\{ \begin{array}{l} a \, [\![\forall(X).\sigma_1]\!]^F_{\eta\updownarrow} \, a \\ a \left[\!\left[E, \begin{matrix}X\\ \mathcal{W}\\ Y\end{matrix} \vdash_{\mathcal{R}} \begin{matrix}\sigma_1\\ \mathcal{S}\\ \sigma_2\end{matrix} \right]\!\right]_{\tilde{\eta}\rho} b \\ \textit{for all } R : A \twoheadrightarrow B \\ \textit{where } \tilde{\eta} = \eta[R, A, B/\mathcal{W}, X, Y] \\ b \, [\![\forall(Y).\sigma_2]\!]^F_{\eta\updownarrow} \, b \end{array} \right. \\
\left[\!\left[E \vdash_{\mathcal{R}} \begin{matrix}\sigma_1\\ < M >\\ \sigma_2\end{matrix} \right]\!\right]_{\eta\rho} & = & \mathit{FRel}([\![E_F \vdash_F M : \sigma_1 \to \sigma_2]\!]^F_{\eta\updownarrow,\rho})
\end{array}$$

Note that for the semantics of quantification and functional relations we have used the parametric semantics of system F (denoted $[\![-]\!]^F$). Thus, the semantics of the corresponding related types judgements do not depend upon their derivations. Moreover, the derivation of a related types judgement always follows exactly the structure of the relation except in the case of functional relations. Therefore, the semantics of a related types judgement is always independent of its derivation.

In the judgement $E_F \vdash_F M : \sigma_1 \to \sigma_2$, E_F stands for the F flattening of E, obtained by retaining the type variables and the term variables of E, and removing the relations. In [ACC93] appears a lemma, called (Flattened F derivations from $\mathcal{R}$ derivations), that asserts that if $E \vdash_{\mathcal{R}} M : \sigma_1 \to \sigma_2$ is provable in $\mathcal{R}$, then $E_F \vdash_F M : \sigma_1 \to \sigma_2$ is provable in F.

Remark 5.36 *The presence of functional relations prevents us from using the intersection operator of definition 3.10 since it is no longer true that*

$$F(\overline{S}) : F(\overline{A}) \twoheadrightarrow F(\overline{B})$$

Indeed, consider the case where the function F is constantly equal to a functional relation: it does not maps pers to pers.

Instead, we have given a direct, pointwise definition of the meaning of intersection.

This interpretation is sound:

Theorem 5.37 (Soundness for Related Types Judgements)
For every related types judgement $E \vdash_{\mathcal{R}} \begin{matrix}\sigma_1\\ \mathcal{S}\\ \sigma_2\end{matrix}$ *and assignments η and ρ such that $\rho \models_\eta E$:*

$$\left[\!\!\left[E \vdash_{\mathcal{R}} \begin{matrix}\sigma_1\\ \mathcal{S}\\ \sigma_2\end{matrix} \right]\!\!\right]_{\eta\rho} : [\![\sigma_1]\!]^F_{\eta\updownarrow} \nrightarrow [\![\sigma_2]\!]^F_{\eta\updownarrow}$$

Proof. By induction on the complexity of the relation expression $\mathcal{S}$. We check only the case of quantification over relation variables.

Suppose that for all $R : A \nrightarrow B$:

$$a\left([\![\forall(X).\sigma_1]\!]^F_{\eta\updownarrow}\right) b \left[\!\!\left[E,\ \begin{matrix}X\\ \mathcal{W}\\ Y\end{matrix} \vdash_{\mathcal{R}} \begin{matrix}\sigma_1\\ \mathcal{S}\\ \sigma_2\end{matrix} \right]\!\!\right]_{\tilde{\eta}\rho} c\left([\![\forall(Y).\sigma_2]\!]^F_{\eta\updownarrow}\right) d$$

where $\tilde{\eta} = \eta[R, A, B/\mathcal{W}, X, Y]$. We have to prove:

$$a \left[\!\!\left[E,\ \begin{matrix}X\\ \mathcal{W}\\ Y\end{matrix} \vdash_{\mathcal{R}} \begin{matrix}\sigma_1\\ \mathcal{S}\\ \sigma_2\end{matrix} \right]\!\!\right]_{\tilde{\eta}\rho} d$$

By induction hypothesis we have:

$$\left[\!\!\left[E,\ \begin{matrix}X\\ \mathcal{W}\\ Y\end{matrix} \vdash_{\mathcal{R}} \begin{matrix}\sigma_1\\ \mathcal{S}\\ \sigma_2\end{matrix} \right]\!\!\right]_{\tilde{\eta}\rho} : [\![\sigma_1]\!]^F_{\eta\updownarrow[A/X]} \nrightarrow [\![\sigma_2]\!]^F_{\eta\updownarrow[B/Y]}$$

since the side conditions ensure that X is not free in σ_2 and that Y is not free in σ_1. We are done, since by assumption we know that $a\,[\![\sigma_1]\!]^F_{\eta\updownarrow[A/X]}\,b$ and $c\,[\![\sigma_2]\!]^F_{\eta\updownarrow[B/Y]}\,d$. ■

Given a type expression σ, we expect its F semantics to be equal to the $\mathcal{R}$ semantics of σ^*. This turns out to be an essential property.

Proposition 5.38 *If $E \vdash_{\mathcal{R}} \sigma$ then for all assignments η and ρ such that $\rho \models_\eta E$ we have*

$$\left[\!\!\left[E \vdash_{\mathcal{R}} \begin{matrix}\sigma\\ \sigma^*\\ \sigma\end{matrix} \right]\!\!\right]_{\eta\rho} = [\![\sigma]\!]^F_{\eta\updownarrow}$$

This proposition is a corollary of a more general statement:

Lemma 5.39 *If* $E,\ \begin{array}{c}\overline{X'}\\ \overline{\mathcal{W}}\\ \overline{X''}\end{array} \vdash_{\mathcal{R}} \begin{array}{c}\sigma[\overline{X'}/\overline{X}]\\ \sigma^*[\overline{\mathcal{W}}/\overline{X}]\\ \sigma[\overline{X''}/\overline{X}]\end{array}$ *then for all assignments* η *and* ρ *such that* $\rho \models_\eta E$*, and for all tuples* $\overline{S} : \overline{A} \nrightarrow \overline{B}$ *of relations, we have:*

$$\left\llbracket E,\ \begin{array}{c}\overline{X'}\\ \overline{\mathcal{W}}\\ \overline{X''}\end{array} \vdash_{\mathcal{R}} \begin{array}{c}\sigma[\overline{X'}/\overline{X}]\\ \sigma^*[\overline{\mathcal{W}}/\overline{X}]\\ \sigma[\overline{X''}/\overline{X}]\end{array}\right\rrbracket_{\tilde{\eta},\rho} = \llbracket\sigma\rrbracket^F_{\eta\updownarrow[\overline{S}/\overline{X}]}$$

where $\tilde{\eta} \equiv \eta[\overline{S},\overline{A},\overline{B}/\overline{\mathcal{W}},\overline{X'},\overline{X''}]$

Proof. By induction on σ. The cases of type variables and arrow types are immediate. Suppose $\sigma \equiv \forall(Y).\sigma_1$. We introduce the following abbreviations: $\sigma_1' \equiv \sigma_1[\overline{X'}/\overline{X}]$, $\sigma_1'' \equiv \sigma_1[\overline{X''}/\overline{X}]$, and $\widehat{\sigma_1^*} \equiv \sigma_1^*[\overline{\mathcal{W}}/\overline{X}]$. By definition we have:

$$a \left\llbracket E,\ \begin{array}{c}\overline{X'}\\ \overline{\mathcal{W}}\\ \overline{X''}\end{array} \vdash_{\mathcal{R}} \begin{array}{c}\forall(Y).\sigma_1'\\ \forall(\mathcal{W}_0).\widehat{\sigma_1^*}[\mathcal{W}_0/Y]\\ \forall(Y).\sigma_1''\end{array}\right\rrbracket_{\tilde{\eta},\rho} b \quad \textit{iff}$$

$$\left\{\begin{array}{l} a\,\llbracket\forall(Y).\sigma_1\rrbracket^F_{\eta\updownarrow[\overline{A}/\overline{X}]}\,a \\ a\left\llbracket E,\ \begin{array}{c}\overline{X'}\\ \overline{\mathcal{W}}\\ \overline{X''}\end{array},\ \begin{array}{c}Y'\\ \mathcal{W}_0\\ Y''\end{array} \vdash_{\mathcal{R}} \begin{array}{c}\sigma_1'[Y'/Y]\\ \widehat{\sigma_1^*}[\mathcal{W}_0/Y]\\ \sigma_1''[Y''/Y]\end{array}\right\rrbracket_{\tilde{\eta}'',\rho} b \\ \quad \textit{for all}\ \ S_0 : A_0 \nrightarrow B_0 \\ b\,\llbracket\forall(Y).\sigma_1\rrbracket^F_{\eta\updownarrow[\overline{B}/\overline{X}]}\,b \end{array}\right.$$

where $\tilde{\eta}'' \equiv \tilde{\eta}[S_0, A_0, B_0/\mathcal{W}_0, Y', Y'']$; and:

$$a\,\llbracket\forall(Y).\sigma_1\rrbracket^F_{\eta\updownarrow[\overline{S}/\overline{X}]}\,b \quad \textit{iff} \quad \left\{\begin{array}{l} a\,\llbracket\forall(Y).\sigma_1\rrbracket^F_{\eta\updownarrow[\overline{A}/\overline{X}]}\,a \\ a\,\llbracket\sigma_1\rrbracket^F_{\eta\updownarrow[S_0,\overline{S}/Y,\overline{X}]}\,b \qquad \textit{for all}\ \ S_0 : A_0 \nrightarrow B_0 \\ b\,\llbracket\forall(Y).\sigma_1\rrbracket^F_{\eta\updownarrow[\overline{B}/\overline{X}]}\,b \end{array}\right.$$

We compare the right-hand sides of the two definitions. The upper and bottom rows are exactly the same, while the middle rows are equal by induction hypothesis. ■

As usual, we have also a substitution lemma:

Lemma 5.40 (Substitution Lemma for Relations)
For all related types judgements $E,\ \begin{array}{c}\overline{X}\\ \overline{\mathcal{W}}\\ \overline{Y}\end{array} \vdash_{\mathcal{R}} \begin{array}{c}\sigma_1\\ \mathcal{R}\\ \sigma_2\end{array}$ *and*$E \vdash_{\mathcal{R}} \begin{array}{c}\overline{\tau_1}\\ \overline{\mathcal{S}}\\ \overline{\tau_2}\end{array}$ *and all assignments* η *and* ρ *such that* $\rho \models_\eta E$*, we have:*

$$\left\llbracket E \vdash_{\mathcal{R}} \begin{array}{c}\sigma_1[\overline{\tau_1},\overline{\tau_2}/\overline{X},\overline{Y}]\\ \mathcal{R}[\overline{\mathcal{S}},\overline{\tau_1},\overline{\tau_2}/\overline{\mathcal{W}},\overline{X},\overline{Y}]\\ \sigma_2[\overline{\tau_1},\overline{\tau_2}/\overline{X},\overline{Y}]\end{array}\right\rrbracket_{\eta,\rho} = \left\llbracket E,\ \begin{array}{c}\overline{X}\\ \overline{\mathcal{W}}\\ \overline{Y}\end{array} \vdash_{\mathcal{R}} \begin{array}{c}\sigma_1\\ \mathcal{R}\\ \sigma_2\end{array}\right\rrbracket_{\eta[\overline{S},\overline{A},\overline{B}/\overline{\mathcal{W}},\overline{X},\overline{Y}],\rho}$$

where

$$\left[\!\!\left[E \vdash_{\mathcal{R}} \begin{array}{c}\overline{\tau_1}\\ \overline{\mathcal{S}}\\ \overline{\tau_2}\end{array} \right]\!\!\right]_{\eta,\rho} = \overline{S} : \overline{A} \nrightarrow \overline{B}$$

Proof. The proof is by induction on the proof of $E, \begin{array}{c}\overline{X}\\ \overline{\mathcal{W}}\\ \overline{Y}\end{array} \vdash_{\mathcal{R}} \begin{array}{c}\sigma_1\\ \mathcal{R}\\ \sigma_2\end{array}$. We limit ourselves to the case (Rel FRel). Thus suppose $\mathcal{R} = < M >$, and let E' stand for $E, \begin{array}{c}\overline{X}\\ \overline{\mathcal{W}}\\ \overline{Y}\end{array}$. The premise of (Rel FRel) is $E' \vdash_{\mathcal{R}} M : \sigma_1 \rightarrow \sigma_2$. We have $E'_F \vdash_F M : \sigma_1 \rightarrow \sigma_2$ by lemma (Flattened F derivations from $\mathcal{R}$ derivations) of [ACC93]. Then we can apply the substitution lemma 4.22 (Substitution Lemma for Terms), and obtain:

$$[\![E_F \vdash_F M[\overline{\tau_1}, \overline{\tau_2}/\overline{X}, \overline{Y}] : \sigma_1[\overline{\tau_1}, \overline{\tau_2}/\overline{X}, \overline{Y}] \rightarrow \sigma_2[\overline{\tau_1}, \overline{\tau_2}/\overline{X}, \overline{Y}]]\!]_{\eta\updownarrow\rho} =$$
$$[\![E, \overline{X} \vdash_F M : \sigma]\!]_{\eta\updownarrow[\overline{[\![\tau_1]\!]_\eta}, \overline{[\![\tau_1]\!]_\eta}/\overline{X}, \overline{Y}]\rho}$$

The conclusion follows by the definition of the interpretation of functional relations. ■

5.2 Related values

In order to check the validity of a related values judgement, we interpret the two terms of the judgement in the parametric per model of F, and we prove that these interpretations are related by the semantics of the relation expression of the judgement.

Thus we first interpret a related values judgement by a function

$$[\![-]\!] : \mathit{RelValJudge} \rightarrow \mathit{RelAssign} \rightarrow \mathit{ValAssign} \rightarrow \bigcup_{A,B \in Obj(PER)} Q(A) \times Q(B)$$

$$\left[\!\!\left[E \vdash_{\mathcal{R}} \begin{array}{c} M : \sigma_1\\ \mathcal{S}\\ N : \sigma_2\end{array} \right]\!\!\right]_{\eta\rho} = ([\![E_F \vdash_F M : \sigma_1]\!]^F_{\eta\updownarrow\rho}, [\![E_F \vdash_F N : \sigma_1]\!]^F_{\eta\updownarrow\rho})$$

Next we prove that the two components of this interpretation are related.

Theorem 5.41 (Soundness for Related Values Judgements)
Given a related values judgement $E \vdash_{\mathcal{R}} \begin{array}{c} M : \sigma_1\\ \mathcal{S}\\ N : \sigma_2\end{array}$ *and assignments* η *and* ρ *such that* $\rho \models_\eta E$, *we have:*

$$[\![E_F \vdash_F M : \sigma_1]\!]^F_{\eta\updownarrow\rho} \left[\!\!\left[E \vdash_{\mathcal{R}} \begin{array}{c}\sigma_1\\ \mathcal{S}\\ \sigma_2\end{array} \right]\!\!\right]_{\eta\rho} [\![E_F \vdash_F N : \sigma_2]\!]^F_{\eta\updownarrow\rho}$$

Proof. By induction on the derivation.

- Case (Rel Val Symm): By proposition 5.38 $\left[\!\left[E \vdash_{\mathcal{R}} \begin{array}{c}\sigma\\ \sigma^*\\ \sigma\end{array} \right]\!\right]$ is a per, so symmetry holds.

- Cases (Rel Val Saturation Left), (Rel Val Saturation Right): By theorem 5.37 we know that

$$\left[\!\left[E \vdash_{\mathcal{R}} \begin{array}{c}\sigma_1\\ \mathcal{S}\\ \sigma_2\end{array} \right]\!\right]_{\eta\rho} : [\![\sigma_1]\!]^F_{\eta\updownarrow} \nrightarrow [\![\sigma_2]\!]^F_{\eta\updownarrow}$$

so in both cases the results follow by proposition 5.38.

- Case (Rel Val $x\mathcal{R}y$): Follows immediately from the hypothesis on the assignments.

- Cases (Rel Val $\mathcal{R}x$), (Rel Val $\mathcal{R}y$): Follow by the hypothesis on the assignments and by proposition 5.38.

- Case (Rel Val Fun): Our goal is:

$$[\![\lambda x.erase(M)]\!]_{\rho^\star} \left[\!\left[E \vdash_{\mathcal{R}} \begin{array}{c}\sigma_1 \to \sigma_2\\ \mathcal{R} \to \mathcal{S}\\ \tau_1 \to \tau_2\end{array} \right]\!\right]_{\eta,\rho} [\![\lambda y.erase(N)]\!]_{\rho^\star}$$

This is equivalent to:

$$\begin{array}{ll} \text{if} & a \left[\!\left[E \vdash_{\mathcal{R}} \begin{array}{c}\sigma_1\\ \mathcal{R}\\ \tau_1\end{array} \right]\!\right]_{\eta,\rho} b \\ \text{then} & [\![erase(M)]\!]^D_{\rho^\star[a/x]} \left[\!\left[E \vdash_{\mathcal{R}} \begin{array}{c}\tau_1\\ \mathcal{S}\\ \tau_2\end{array} \right]\!\right]_{\eta,\rho} [\![erase(N)]\!]^D_{\rho^\star[b/x]} \end{array}$$

The premise of this implication implies:

$$\rho[[a]_{[\![\sigma_1]\!]_\eta}, [b]_{[\![\tau_1]\!]_\eta}/x, y] \models_\eta E, \begin{array}{c} x : \sigma_1\\ \mathcal{R}\\ y : \tau_1\end{array}$$

and the conclusion follows by induction hypothesis.

- Case (Rel Val Appl): Follows immediately by applying the induction hypothesis to the premises.

- Case (Rel Val Fun2): We have to prove

$$[\![erase(M)]\!]_{\rho^\star} \left[\!\left[E \vdash_{\mathcal{R}} \begin{array}{c}\forall(X).\sigma_1\\ \forall(\mathcal{W}).\mathcal{S}\\ \forall(Y).\sigma_2\end{array} \right]\!\right]_{\eta,\rho} [\![erase(N)]\!]_{\rho^\star}$$

that is, for all $R : A \nrightarrow B$:

$$[\![erase(M)]\!]_{\rho^\star} \left[\!\!\left[E \vdash_{\mathcal{R}} \begin{array}{c}\sigma_1 \\ \mathcal{S} \\ \sigma_2\end{array} \right]\!\!\right]_{\eta[R,A,B/\mathcal{W},X,Y],\rho} [\![erase(N)]\!]_{\rho^\star}$$

and this follows by applying the induction hypothesis to the premise.

- Case (Rel Val Appl2): We have to prove

$$[\![erase(M)]\!]_{\rho^\star} \left[\!\!\left[E \vdash_{\mathcal{R}} \begin{array}{c}\sigma_1[\tau_1/X] \\ \mathcal{S}[\mathcal{R}/\mathcal{W}] \\ \sigma_2[\tau_2/Y]\end{array} \right]\!\!\right]_{\eta,\rho} [\![erase(N)]\!]_{\rho^\star}$$

By lemma 5.40 this is equal to

$$[\![erase(M)]\!]_{\rho^\star} \left[\!\!\left[E \vdash_{\mathcal{R}} \begin{array}{c}\sigma_1 \\ \mathcal{S} \\ \sigma_2\end{array} \right]\!\!\right]_{\eta[R',A',B'/\mathcal{W},X,Y],\rho} [\![erase(N)]\!]_{\rho^\star}$$

where

$$R' : A' \nrightarrow B' \equiv \left[\!\!\left[E \vdash_{\mathcal{R}} \begin{array}{c}\tau_1 \\ \mathcal{R} \\ \tau_2\end{array} \right]\!\!\right]_{\eta,\rho}$$

Now applying the induction hypothesis to the premise we have, for all $R : A \nrightarrow B$:

$$[\![erase(M)]\!]_{\rho^\star} \left[\!\!\left[E \vdash_{\mathcal{R}} \begin{array}{c}\sigma_1 \\ \mathcal{S} \\ \sigma_2\end{array} \right]\!\!\right]_{\eta[R,A,B/\mathcal{W},X,Y],\rho} [\![erase(N)]\!]_{\rho^\star}$$

Therefore, the result follows by instantiating R with R'.

- Cases (Rel Val FRel Intro), (Rel Val FRel Elim): The first one follows from the definition of the meaning of functional relations while for the second we need also proposition 5.38.

- Cases (Rel Val Beta), (Rel Val Beta2), (Rel Val Eta), (Rel Val Eta2): Their soundness follow, as usual, by proposition 5.38 and by the fact that they are valid in the parametric model of F.

■

This concludes the proof of soundness for our interpretation of $\mathcal{R}$. This proof has been rather concrete. It might be interesting to have an abstract characterization of the notion of model for $\mathcal{R}$, and then to recast our proof in more abstract terms.

6 Conclusion

We have defined two parametric models of system F and used one of them as a basis for an interpretation of system $\mathcal{R}$. In hindsight, our results may not seem surprising. However, the definitions include a number of tricky, "obvious" details. Details of this sort were left implicit in the work of Bainbridge et al. [BFSS90], and misunderstood in the first, inconsistent version of $\mathcal{R}$. Therefore, we feel that a careful interpretation of $\mathcal{R}$ is important.

The interpretation has been helpful both in understanding $\mathcal{R}$ and in thinking about other formal systems for reasoning about polymorphic programs. Several other formal systems come to mind. Following a suggestion of [PA93], we have started to consider a formal system with relations of arities other than 2. Reynolds' discussed relations of all arities in his original work, but binary relations have been preferred more recently (e.g., in [BFSS90]), in part arbitrarily. It seems interesting to extend the model presented here to support relations of all arities.

Acknowledgements

Ryu Hasegawa and Eugenio Moggi both made useful suggestions.

Appendix

System F

Environments

(Env $\emptyset$)
$$\vdash_F \emptyset$$

(Env X)
$$\frac{\vdash_F E \qquad X \notin E}{\vdash_F E, X}$$

(Env x)
$$\frac{E \vdash_F \sigma \qquad x \notin E}{\vdash_F E, x : \sigma}$$

Types

(Type X)
$$\frac{\vdash_F E', X, E''}{E', X, E'' \vdash_F X}$$

(Type Arrow)
$$\frac{E \vdash_F \sigma \qquad E \vdash_F \tau}{E \vdash_F \sigma \rightarrow \tau}$$

(Type Forall)
$$\frac{E, X \vdash_F \sigma}{E \vdash_F \forall(X).\sigma}$$

Values

(Val x)
$$\frac{\vdash_F E', x : \sigma, E''}{E', x : \sigma, E'' \vdash_F x : \sigma}$$

(Val Fun)
$$\frac{E, x : \sigma \vdash_F M : \tau}{E \vdash_F \lambda(x : \sigma).M : \sigma \rightarrow \tau}$$

(Val Fun2)
$$\frac{E, X \vdash_F M : \sigma}{E \vdash_F \lambda(X).M : \forall(X).\sigma}$$

(Val Appl)
$$\frac{E \vdash_F M : \sigma \rightarrow \tau \qquad E \vdash_F N : \sigma}{E \vdash_F MN : \tau}$$

(Val Appl2)
$$\frac{E \vdash_F M : \forall(X).\sigma \qquad E \vdash_F \tau}{E \vdash_F M\tau : \sigma[\tau/X]}$$

Value equalities

(Val Eq Symm)
$$\frac{E \vdash_F M = N : \sigma}{E \vdash_F N = M : \sigma}$$

(Val Eq Trans)
$$\frac{E \vdash_F M = N : \sigma \quad E \vdash_F N = P : \sigma}{E \vdash_F M = P : \sigma}$$

(Val Eq x)
$$\frac{\vdash_F E', x : \sigma, E''}{E', x : \sigma, E'' \vdash_F x = x : \sigma}$$

(Val Eq Appl)
$$\frac{E \vdash_F M = N : \sigma \to \tau \quad E \vdash_F P = Q : \sigma}{E \vdash_F MP = NQ : \tau}$$

(Val Eq Fun)
$$\frac{E, x : \sigma \vdash_F M = N : \tau}{E \vdash_F \lambda(x : \sigma).M = \lambda(x : \sigma).N : \sigma \to \tau}$$

(Val Eq Appl2)
$$\frac{E \vdash_F M = N : \forall(X).\sigma \qquad E \vdash_F \tau}{E \vdash_F M\tau = N\tau : \sigma[\tau/X]}$$

(Val Eq Fun2)
$$\frac{E, X \vdash_F M = N : \sigma}{E \vdash_F \lambda(X).M = \lambda(X).N : \forall(X).\sigma}$$

(Val Eq Beta)
$$\frac{E, x : \sigma \vdash_F M = N : \tau \quad E \vdash_F P = Q : \sigma}{E \vdash_F (\lambda(x : \sigma).M)(P) = N[Q/x] : \tau}$$

(Val Eq Beta2)
$$\frac{E, X \vdash_F M = N : \sigma \quad E \vdash_F \tau}{E \vdash_F (\lambda(X).M)(\tau) = N[\tau/X] : \sigma[\tau/X]}$$

(Val Eq Eta)
$$\frac{E \vdash_F M = N : \sigma \to \tau \qquad x \notin E}{E \vdash_F \lambda(x : \sigma).Mx = N : \sigma \to \tau}$$

(Val Eq Eta2)
$$\frac{E \vdash_F M = N : \forall(X).\sigma \qquad X \notin E}{E \vdash_F \lambda(X).MX = N : \forall(X).\sigma}$$

System $\mathcal{R}$

We use the following abbreviations:

- $E \vdash_{\mathcal{R}} \sigma$ for $E \vdash_{\mathcal{R}} \begin{array}{c} \sigma \\ \sigma^* \\ \sigma \end{array}$

- $E \vdash_{\mathcal{R}} M : \sigma$ for $E \vdash_{\mathcal{R}} \begin{array}{c} M : \sigma \\ \sigma^* \\ M : \sigma \end{array}$

- E, X, E' for the environment $E, \begin{array}{c} X \\ \mathcal{W} \\ X' \end{array}, E'$ where X' and $\mathcal{W}$ are fresh variables,

- $E, x : \sigma, E'$ for the environment $E, \begin{array}{c} x : \sigma \\ \sigma^* \\ x' : \sigma \end{array}, E'$ where x' is a fresh variable.

Environments

(Env $\emptyset$)

$\vdash_{\mathcal{R}} \emptyset$

(Env $X\mathcal{W}Y$)

$$\frac{\vdash_{\mathcal{R}} E \qquad \begin{array}{c} X,\mathcal{W},Y \notin E \\ X,\mathcal{W},Y \ \mathit{distinct} \end{array}}{\vdash_{\mathcal{R}} E,\ \begin{array}{c} X \\ \mathcal{W} \\ Y \end{array}}$$

(Env $x\mathcal{R}y$)

$$\frac{E \vdash_{\mathcal{R}} \begin{array}{c} \sigma_1 \\ \mathcal{R} \\ \sigma_2 \end{array} \qquad \begin{array}{c} x,y \notin E \\ x,y \ \mathit{distinct} \end{array}}{\vdash_{\mathcal{R}} E,\ \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}}$$

Related types

(Rel $\mathcal{W}$)

$$\frac{\vdash_{\mathcal{R}} E',\ \begin{array}{c} X \\ \mathcal{W} \\ Y \end{array},E''}{E',\ \begin{array}{c} X \\ \mathcal{W} \\ Y \end{array},E'' \vdash_{\mathcal{R}} \begin{array}{c} X \\ \mathcal{W} \\ Y \end{array}}$$

(Rel $\mathcal{W}X$)

$$\frac{\vdash_{\mathcal{R}} E',\ \begin{array}{c} X \\ \mathcal{W} \\ Y \end{array},E''}{E',\ \begin{array}{c} X \\ \mathcal{W} \\ Y \end{array},E'' \vdash_{\mathcal{R}} X}$$

(Rel $\mathcal{W}Y$)

$$\frac{\vdash_{\mathcal{R}} E',\ \begin{array}{c} X \\ \mathcal{W} \\ Y \end{array},E''}{E',\ \begin{array}{c} X \\ \mathcal{W} \\ Y \end{array},E'' \vdash_{\mathcal{R}} Y}$$

(Rel Arrow)

$$\frac{E \vdash_{\mathcal{R}} \begin{array}{c} \sigma_1 \\ \mathcal{R} \\ \sigma_2 \end{array} \qquad E \vdash_{\mathcal{R}} \begin{array}{c} \tau_1 \\ \mathcal{S} \\ \tau_2 \end{array}}{E \vdash_{\mathcal{R}} \begin{array}{c} \sigma_1 \rightarrow \tau_1 \\ \mathcal{R} \rightarrow \mathcal{S} \\ \sigma_2 \rightarrow \tau_2 \end{array}}$$

(Rel FRel)

$$\frac{E \vdash_{\mathcal{R}} M : \sigma \rightarrow \tau}{E \vdash_{\mathcal{R}} \begin{array}{c} \sigma \\ < M > \\ \tau \end{array}}$$

(Rel Forall)

$$\frac{E,\ \begin{array}{c} X \\ \mathcal{W} \\ Y \end{array} \vdash_{\mathcal{R}} \begin{array}{c} \sigma_1 \\ \mathcal{S} \\ \sigma_2 \end{array} \qquad \begin{array}{c} X \notin \mathcal{S},\sigma_2 \\ Y \notin \mathcal{S},\sigma_1 \end{array}}{E \vdash_{\mathcal{R}} \begin{array}{c} \forall(X).\sigma_1 \\ \forall(\mathcal{W}).\mathcal{S} \\ \forall(Y).\sigma_2 \end{array}}$$

Related values

(Rel Val Saturation Left)

$$\frac{E \vdash_{\mathcal{R}} \begin{array}{c} M:\sigma \\ \sigma^* \\ N:\sigma \end{array} \qquad E \vdash_{\mathcal{R}} \begin{array}{c} N:\sigma \\ \mathcal{R} \\ P:\tau \end{array}}{E \vdash_{\mathcal{R}} \begin{array}{c} M:\sigma \\ \mathcal{R} \\ P:\tau \end{array}}$$

(Rel Val Saturation Right)

$$\frac{E \vdash_{\mathcal{R}} \begin{array}{c} N:\sigma \\ \mathcal{R} \\ P:\tau \end{array} \qquad E \vdash_{\mathcal{R}} \begin{array}{c} P:\tau \\ \tau^* \\ Q:\tau \end{array}}{E \vdash_{\mathcal{R}} \begin{array}{c} N:\sigma \\ \mathcal{R} \\ Q:\tau \end{array}}$$

(Rel Val Symm)

$$\frac{E \vdash_{\mathcal{R}} \begin{array}{c} M:\sigma \\ \sigma^* \\ N:\sigma \end{array}}{E \vdash_{\mathcal{R}} \begin{array}{c} N:\sigma \\ \sigma^* \\ M:\sigma \end{array}}$$

(Rel Val $x\mathcal{R}y$)

$$\frac{\vdash_{\mathcal{R}} E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E''}{E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E'' \vdash_{\mathcal{R}} \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}}$$

(Rel Val $\mathcal{R}x$)

$$\frac{\vdash_{\mathcal{R}} E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E''}{E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E'' \vdash_{\mathcal{R}} x:\sigma_1}$$

(Rel Val $\mathcal{R}y$)

$$\frac{\vdash_{\mathcal{R}} E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E''}{E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E'' \vdash_{\mathcal{R}} y:\sigma_2}$$

(Rel Val Fun)

$$\frac{E, \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array} \vdash_{\mathcal{R}} \begin{array}{c} M:\tau_1 \\ \mathcal{S} \\ N:\tau_2 \end{array} \qquad \begin{array}{c} x \notin N, \mathcal{S} \\ y \notin M, \mathcal{S} \end{array} \qquad E \vdash_{\mathcal{R}} \begin{array}{c} \tau_1 \\ \mathcal{S} \\ \tau_2 \end{array}}{E \vdash_{\mathcal{R}} \begin{array}{c} \lambda(x:\sigma_1).M : \sigma_1 \to \tau_1 \\ \mathcal{R} \to \mathcal{S} \\ \lambda(x:\sigma_2).N : \sigma_2 \to \tau_2 \end{array}}$$

(Rel Val Appl)

$$\frac{E \vdash_{\mathcal{R}} \begin{array}{c} M_1:\sigma_1 \to \tau_1 \\ \mathcal{R} \to \mathcal{S} \\ M_2:\sigma_2 \to \tau_2 \end{array} \qquad E \vdash_{\mathcal{R}} \begin{array}{c} N_1:\sigma_1 \\ \mathcal{R} \\ N_2:\sigma_1 \end{array}}{E \vdash_{\mathcal{R}} \begin{array}{c} M_1 N_1:\tau_1 \\ \mathcal{S} \\ M_2 N_2:\tau_2 \end{array}}$$

(Rel Val Fun2)

$$\frac{E,\ \begin{array}{c}X\\ \mathcal{W}\\ Y\end{array} \vdash_{\mathcal{R}} \begin{array}{c}M:\sigma_1\\ \mathcal{S}\\ N:\sigma_2\end{array} \qquad \begin{array}{c}\\ X\notin\mathcal{S},N,\sigma_2\\ Y\notin\mathcal{S},M,\sigma_1\end{array}}{E\vdash_{\mathcal{R}} \begin{array}{c}\lambda(X).M:\forall(X).\sigma_1\\ \forall(\mathcal{W}).\mathcal{S}\\ \lambda(Y).N:\forall(Y).\sigma_2\end{array}}$$

(Rel Val Appl2)

$$\frac{E\vdash_{\mathcal{R}} \begin{array}{c}M:\forall(X).\sigma_1\\ \forall(\mathcal{W}).\mathcal{S}\\ N:\forall(Y).\sigma_2\end{array} \qquad E\vdash_{\mathcal{R}} \begin{array}{c}\tau_1\\ \mathcal{R}\\ \tau_2\end{array}}{E\vdash_{\mathcal{R}} \begin{array}{c}M\tau_1:\sigma_1[\tau_1/X]\\ \mathcal{S}[\mathcal{R}/\mathcal{W}]\\ N\tau_2:\sigma_2[\tau_2/Y]\end{array}}$$

(Rel Val FRel Intro)

$$\frac{E\vdash_{\mathcal{R}} M:\sigma_1\to\sigma_2 \qquad E\vdash_{\mathcal{R}} N:\sigma_1}{E\vdash_{\mathcal{R}} \begin{array}{c}N:\sigma_1\\ <M>\\ M(N):\sigma_2\end{array}}$$

(Rel Val FRel Elim)

$$\frac{E\vdash_{\mathcal{R}} \begin{array}{c}N_1:\sigma_1\\ <M>\\ N_2:\sigma_2\end{array}}{E\vdash_{\mathcal{R}} \begin{array}{c}M(N_1):\sigma_2\\ \sigma_2^*\\ N_2:\sigma_2\end{array}}$$

(Rel Val Beta)

$$\frac{E,x:\tau\vdash_{\mathcal{R}} M:\sigma \qquad E\vdash_{\mathcal{R}} N:\tau}{E\vdash_{\mathcal{R}} \begin{array}{c}(\lambda(x:\tau).M)N:\sigma\\ \sigma^*\\ M[N/x]:\sigma\end{array}}$$

(Rel Val Beta2)

$$\frac{E,X\vdash_{\mathcal{R}} M:\sigma \qquad E\vdash_{\mathcal{R}} \tau}{E\vdash_{\mathcal{R}} \begin{array}{c}(\lambda(X).M)\tau:\sigma[\tau/X]\\ \sigma^*[\tau^*/X]\\ M[\tau/X]:\sigma[\tau/X]\end{array}}$$

(Rel Val Eta)

$$\frac{E\vdash_{\mathcal{R}} M:\tau\to\sigma \qquad x\notin E}{E\vdash_{\mathcal{R}} \begin{array}{c}\lambda(x:\tau).Mx:\tau\to\sigma\\ (\tau\to\sigma)^*\\ M:\tau\to\sigma\end{array}}$$

(Rel Val Eta2)

$$\frac{E\vdash_{\mathcal{R}} M:\forall(X).\sigma \qquad X\notin E}{E\vdash_{\mathcal{R}} \begin{array}{c}\lambda(X).MX:\forall(X).\sigma\\ (\forall(X).\sigma)^*\\ M:\forall(X).\sigma\end{array}}$$

PER as a PL-category

We review the definition of the standard **PER** model of system F as a PL-category.

Definition 6.42 (Global Category) *The objects of the global category* $\mathbf{E}$ *are generated by the set* $\Omega = Obj(\mathbf{PER})$. *The set of morphisms is defined by*

$$\mathbf{E}[p,q] = \{F : Obj(\mathbf{PER})^p \to Obj(\mathbf{PER})^q\}$$

Definition 6.43 (Indexed Category) *The collection of objects of the indexed category* $\mathbf{G}(p)$ *is* $\mathbf{E}[p,1]$, *by definition of a PL-category. The morphisms in* $\mathbf{G}(p)$ *are the uniformly realized arrows between objects, that is, an arrow* $\Phi : F \to H$ *such that:*

- $F, H \in Obj(\mathbf{G}(p))$
- $\Phi : p \to Mor(\mathbf{PER})$ *such that for all* $\overline{A} \in p$, $\Phi(\overline{A}) : F(\overline{A}) \to H(\overline{A})$ *in* $Mor(\mathbf{PER})$

- *there exists $n \in D$ that computes uniformly Φ in the sense that for all $\overline{A} \in p$, n realizes $\Phi(\overline{A})$ in Mor(**PER**). We say that n is a realizer of $\Phi : F \to H$, and write $n \vdash (\Phi : F \to H)$*

Given $L \in \mathbf{E}[p, q]$, we define $\mathbf{G}(L)$ as the functor from $\mathbf{G}(q)$ to $\mathbf{G}(p)$ that acts on both objects and morphisms as the pre-composition with L. Namely, if F is an object of $\mathbf{G}(q)$, then $\mathbf{G}(L)(F) = F \circ L$, and if $\Phi : F \to H$ is a morphism of $\mathbf{G}(q)$ realized by n, then $\mathbf{G}(L)(\Phi : F \to H)$ is the uniquely determined morphism, denoted $(\Phi \circ L) : (F \circ L) \to (H \circ L)$, realized by n.

Note that $\mathbf{G}(0)$ is isomorphic to **PER**.

Definition 6.44 (Indexed Adjunction) *Given an object L of $\mathbf{G}(p+1)$, we define*

$$\forall(p)(L) = \bigcap_{A \in Obj(PER)} L[-, A]$$

The behavior of $\forall(p)$ on morphisms is that if n realizes the morphism $\Phi : F \to H$ in $\mathbf{G}(p+1)$, then $\forall(p)(\Phi : F \to H)$ is the (unique) arrow from $\forall(p)(F)$ to $\forall(p)(H)$ in $\mathbf{G}(p)$ realized by n. For every p, $F \in Obj(\mathbf{G}(p))$ and $H \in Obj(\mathbf{G}(p+1))$, we define the isomorphism

$$\boldsymbol{\Delta}(p) : \mathbf{G}(p+1)[\mathbf{Fst}(p)(F), H] \xrightarrow{\cong} \mathbf{G}(p)[F, \forall(p)(H)]$$

so that it sends a morphism $\Phi : \mathbf{Fst}(p)(F) \to H$ realized by n to the unique morphism from F to $\forall(p)(H)$ in $\mathbf{G}(p)$ realized by n.

References

[ACC93] M. Abadi, L. Cardelli, and P.-L. Curien. Formal parametric polymorphism. In *A Collection of Contributions in Honour of Corrado Böhm on the Occasion of his 70th Birthday*, volume 121 of *Theoretical Computer Science*, pages 9–58, 1993. An early version appeared in the Proceedings of the 20th Ann. ACM Symp. on Principles of Programming Languages.

[AM92] A. Asperti and S. Martini. Categorical models of polymorphism. *Information and Computation*, 99:1–79, 1992.

[Bar84] H.P. Barendregt. *The Lambda Calculus. Its Syntax and Semantics.* Number 103 in Studies in Logic and the Foundations of Mathematics. North-Holland, 1984. Revised edition.

[BFSS90] E.S. Bainbridge, P. Freyd, A. Scedrov, and P.J. Scott. Functorial polymorphism. *Theoretical Computer Science*, 70:35–64, 1990.

[CL91] L. Cardelli and G. Longo. A semantic basis for QUEST. *Journal of Functional Programming*, 1(4):417–458, October 1991.

[Cur89] P.-L. Curien. Alpha-conversion, conditions on variables and categorical logic. *STUDIA LOGICA*, XLVIII(3), September 1989.

[Gir72] J.-Y. Girard. *Interprétation fonctionnelle et élimination des coupures dans l'arithmétique d'ordre supérieur.* Thèse de Doctorat d'Etat, Université Paris VII, 1972.

[Has93] R. Hasegawa. Categorical data types in parametric polymorphism. *Mathematical Structures in Computer Science*, 1993. To appear.

[Law69] F.W. Lawvere. Adjointness in foundations. *Dialectica*, 23(3-4):281–296, 1969.

[Mai91] H.G. Mairson. Outline of a proof theory of parametricity. In J. Hughes, editor, *Proceedings of the 5th International Conference on Functional Programming Languages and Computer Architecture,* Cambridge, MA, USA, August 1991, number 523 in Lecture Notes in Computer Science, pages 313–327. Springer-Verlag, 1991.

[Mit90] J.C. Mitchell. A type-inference approach to reduction properties and semantics of polymorphic expressions. In G. Huet, editor, *Logical Foundations of Functional Programming,* Reading, MA, USA, pages 195–212. Addison-Wesley, 1990.

[MR91] QingMing Ma and J.C. Reynolds. Types, abstraction and parametric polymorphism, part 2. In S. Brookes, M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, *Proceedings of the Int. Conf. on Mathematical Foundations of Programming Semantics,* Pittsburgh, PA, USA, number 598 in Lecture Notes in Computer Science. Springer-Verlag, 1991.

[PA93] G. Plotkin and M. Abadi. A logic for parametric polymorphism. In *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, March 1993, Utrecht, NL, number 664 in Lecture Notes in Computer Science, pages 361–375. Springer-Verlag, 1993.

[Rey83] J.C. Reynolds. Types, abstraction and parametric polymorphism. In R.E.A. Mason, editor, *INFORMATION PROCESSING '83*, pages 513–523. Elsevier Science Publishers B.V.North-Holland, 1983.

[See83] R.A.G. Seely. Hyperdoctrines, natural deduction and the Beck condition. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 29:505–542, 1983.

[See87] R.A.G. Seely. Categorical semantics for higher order polymorphic lambda calculus. *The Journal of Symbolic Logic*, 52(4):969–989, December 1987.

[Str67] C. Strachey. Fundamental concepts in programming languages. Lecture Notes, International Summer School in Programming Languages, Copenhagen, Denmark, Unpublished, August 1967.

[Wad89] P. Wadler. Theorems for free! In *Proceedings of the Fourth International Conference on Functional Programming Languages and Computer Architecture*, pages 347–359. ACM press, 1989.