A Model for Formal Parametric Polymorphism: A PER Interpretation for System $\mathcal{R}$

Roberto Bellucci[1,3], Martín Abadi[2], Pierre-Louis Curien[1]

[1] LIENS, CNRS - Département de Mathématiques et Informatique de l'Ecole Normale Supérieure 45, rue d'Ulm, 75005 Paris, France

[2] Digital Equipment Corporation, Systems Research Center, 130 Lytton Avenue, Palo Alto, California 94301, USA

[3] Dipartimento di Matematica, Via del Capitano 15, 53100 Siena, Italy

Abstract. System $\mathcal{R}$ is an extension of system F that formalizes Reynolds' notion of relational parametricity. In system $\mathcal{R}$, considerably more λ-terms can be proved equal than in system F: for example, the encoded weak products of F are strong products in $\mathcal{R}$. Also, many "theorems for free" à la Wadler can be proved formally in $\mathcal{R}$. In this paper we describe a semantics for system $\mathcal{R}$. As a first step, we give a precise 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.

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

The principle of parametricity has gone through many avatars. First 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, let us take a polymorphic function f of type $\forall(X).X \to X$ (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 without reference to a particular model. Other formal systems with similar features exist, serving related purposes; 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 [ACC93]. The inconsistency 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]. 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:

$$\vdash_{\mathcal{R}} E \qquad\qquad E \vdash_{\mathcal{R}} \begin{array}{c}\sigma\\ \mathcal{S}\\ \tau\end{array} \qquad\qquad E \vdash_{\mathcal{R}} \begin{array}{c}M:\sigma\\ \mathcal{S}\\ N:\tau\end{array}$$

which mean, respectively, E is a legal environment; $\mathcal{S}$ is a relation between types σ and τ in E; and $\mathcal{S}$ 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:

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

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

$$\begin{array}{c}X\\ \mathcal{W}\\ Y\end{array} \qquad \text{and} \qquad \begin{array}{c}x:\sigma\\ \mathcal{S}\\ y:\tau\end{array}$$

which mean, respectively, $\mathcal{W}$ is a relation variable between type variables X (domain) and Y (codomain); and 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 for related values judgements of $\mathcal{R}$. 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 $\rightarrow$ are, respectively:

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

$$\frac{E \vdash_{\mathcal{R}} \begin{array}{c}M_1:\sigma_1 \rightarrow \tau_1\\ \mathcal{R} \rightarrow \mathcal{S}\\ M_2:\sigma_2 \rightarrow \tau_2\end{array} \qquad E \vdash_{\mathcal{R}} \begin{array}{c}N_1:\sigma_1\\ \mathcal{R}\\ N_2:\sigma_1\end{array}}{E \vdash_{\mathcal{R}} \begin{array}{c}M_1N_1:\tau_1\\ \mathcal{S}\\ M_2N_2:\tau_2\end{array}}$$

These rules follow the same pattern as the F rules:

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

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:

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

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:

$$\begin{array}{ccc} \text{(Rel Val } x\mathcal{R}y\text{)} & \text{(Rel Val } \mathcal{R}x\text{)} & \text{(Rel Val } \mathcal{R}y\text{)} \\ \dfrac{\vdash_{\mathcal{R}} E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E''}{E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E'' \vdash_{\mathcal{R}} \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}} & \dfrac{\vdash_{\mathcal{R}} E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E''}{E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E'' \vdash_{\mathcal{R}} \begin{array}{c} x:\sigma_1 \\ {\sigma_1}^* \\ x:\sigma_1 \end{array}} & \dfrac{\vdash_{\mathcal{R}} E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E''}{E', \begin{array}{c} x:\sigma_1 \\ \mathcal{R} \\ y:\sigma_2 \end{array}, E'' \vdash_{\mathcal{R}} \begin{array}{c} y:\sigma_2 \\ {\sigma_2}^* \\ y:\sigma_2 \end{array}} \end{array}$$

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, $\to$, 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 [ACC93].

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

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

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.

In $\mathcal{R}$, free theorems can be stated and proved in a logical framework and without any reference to particular classes of models. An easy "theorem for free" asserts that the type $\forall(X).X \to Bool$ contains only constant functions; this is not provable in F. There are many more substantial examples: for example, encoded products, sums, algebras, quantifications are strong. Various metatheorems can also be obtained, such as syntactic versions of Reynolds' abstraction theorem and identity extension lemma.

The full sets of rules for systems F and $\mathcal{R}$ can be found in [ACC93]. Below we list only the rules for the related types judgements.

(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 $\mathcal{W}X$)

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

(Rel $\mathcal{W}Y$)

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

(Rel Arrow)

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

(Rel FRel)

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

(Rel Forall)

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

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 $\mathcal{R}$. For more details, see for example [Bar84].

First we recall the definition of interpretation (over a combinatory algebra) 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.

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:

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

where $\triangleright_{CL}$ is the reduction relation on *CL*-terms obtained by orienting the two equations for k and s from left to right.

This enables us to translate λ-terms to *CL*-terms. Given a λ-term M, we denote its translation by $(M)_{CL}$. Using the translation, we can define the semantics of a λ-term M by $[\![M]\!]_\rho = [\![(M)_{CL}]\!]_\rho$. This interpretation satisfies the rule of β-conversion. The difference between the usual interpretation of the λ-calculus based on λ-models and the combinatory-algebra interpretation regards the ξ rule (if $M = N$ then $\lambda(x).M = \lambda(x).N$), which is not valid in arbitrary combinatory algebras.

In a partial combinatory algebra, the application $\cdot$ is partial and for all a, b, c it satisfies: $(k \cdot a) \cdot b = a$ and $((s \cdot a) \cdot b) \cdot c \simeq (a \cdot c) \cdot (b \cdot c)$; this implies that we cannot define the meaning of general λ-terms. However, we can still define the meaning of some terms, useful in later results:

Lemma 1. *For every λ-term M and variable x, the semantics of $(\lambda x.M)_{CL}$ is defined in every partial combinatory algebra.*

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

Definition 2. The category **PER** of partial equivalence relations (pers) on a partial combinatory algebra D is defined by:

- $A \in \mathit{Obj}(\mathbf{PER})$ iff it is a symmetric and transitive partial relation on D; its domain of definition is denoted $dom(A)$ and the partial quotient of D by A is denoted $Q(A)$;
- a morphism $(f : A \to B) \in \mathit{Mor}(\mathbf{PER})$ is a function $f : Q(A) \to Q(B)$ such that there is an element $n \in D$ that realizes f, that is, for all $a \in dom(A) : f([a]_A) = [n \cdot a]_B$.

Definition 3. The category **SAT** of saturated relations on a partial combinatory algebra D is defined by:

- a saturated relation $(R : A \not\twoheadrightarrow B) \in \mathit{Obj}(\mathbf{SAT})$ 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 $f : (R : A \not\twoheadrightarrow A') \to (S : B \not\twoheadrightarrow B') \in \mathit{Mor}(\mathbf{SAT})$ between two saturated relations consists of a couple $(f' : A \to B, f'' : A' \to B')$ of per morphisms such that:

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

We often write R for $R : A \not\twoheadrightarrow B$, and sometimes refer to a saturated relation simply as a relation. We call A and B the domain and the codomain of $R : A \not\twoheadrightarrow B$, 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**.

Next, 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 projections $(\text{-})_1, (\text{-})_2 : D \to D$ in D as the functions 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 1 their semantics is defined even in a partial combinatory algebra.)

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

(SAT-FUNC) if $\overline{S} : \overline{A} \not\rightarrow \overline{B}$ then $F(\overline{S}) : F(\overline{A}) \not\rightarrow F(\overline{B})$

Note that $F(\overline{S}) : F(\overline{A}) \not\rightarrow F(\overline{B})$ implies that $F(\overline{A})$ and $F(\overline{B})$ are pers. We stipulate that a function $F : Obj(\mathbf{SAT})^0 \to Obj(\mathbf{SAT})$ 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**:

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

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

Definition 4 (Product). Given $R : A \not\rightarrow B$ and $S : C \not\rightarrow D$, their product $R \times S : (A \times C) \not\rightarrow (B \times D)$ is defined by:

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

Definition 5 (Exponentiation). Given $R : A \not\rightarrow B$ and $S : C \not\rightarrow D$ their exponentiation $R \Rightarrow S : (A \Rightarrow C) \not\rightarrow (B \Rightarrow D)$ is defined by:

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

It is easy to prove that these constructions are the categorical product and exponentiation in **SAT**, and extend those defined in **PER**.

Like pers, saturated relations also support infinite intersections:

Definition 6 (Intersection with Parameters). Given a function

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

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

$$\bigcap_{R:A \not\rightarrow B}^{SAT} F[-, R] : Obj(\mathbf{SAT})^k \to Obj(\mathbf{SAT})$$

with

$$\left(\bigcap_{R:A \not\rightarrow B}^{SAT} F[-, R] \right) (\overline{S}) : \left(\bigcap_{R:A \not\rightarrow B} F(dom(\overline{S}), R) \right) \not\rightarrow \left(\bigcap_{R:A \not\rightarrow B} F(cod(\overline{S}), R) \right)$$

by:

$$a \left(\left(\bigcap_{R:A \not\to B}^{SAT} F[-,R] \right) (\overline{S}) \right) b \ \text{ iff } \ \begin{cases} a \left(\bigcap_{R:A \not\to B} F(dom(\overline{S}), R) \right) a \\ a \left(\bigcap_{R:A \not\to B} F(\overline{S}, R) \right) b \\ b \left(\bigcap_{R:A \not\to B} F(cod(\overline{S}), R) \right) b \end{cases}$$

for all $\overline{S} : \overline{A} \not\to \overline{B} \in Obj(\mathbf{SAT})^k$.

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**.

Proposition 7. *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 $\overline{S} : \overline{A} \not\to \overline{B}$ then*

$$\left(\bigcap_{R:A \not\to B}^{SAT} F[-,R](\overline{S}) \right) : \left(\bigcap_{R:A \not\to B}^{SAT} F[-,R](\overline{A}) \right) \not\to \left(\bigcap_{R:A \not\to B}^{SAT} F[-,R](\overline{B}) \right)$$

In particular, this proposition says that even if $F[\overline{A}, R]$ is in general a saturated relation, its intersection over all saturated relations R is a per. This is the crucial property that allows us, in the next section, to interpret quantification over type variables as ranging over all saturated relations and not only over pers.

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 model of F, or **SAT** model 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 model 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

Throughout this subsection, we assume that D is at least a total combinatory algebra.

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.

The type expressions of F are obtained from type variables by the $\to$ and $\forall$ constructors. We interpret a type expression σ by a function

$$[\![\sigma]\!] : RelAssign \to Obj(\mathbf{SAT})$$

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*. The subset of per assignments is $PerAssign$. 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:

$$\varepsilon\uparrow(X) = dom(\varepsilon(X)) \qquad\qquad \varepsilon\downarrow(X) = cod(\varepsilon(X))$$

Definition 8 (Semantics of Type Expressions). The semantics of type expressions is defined by:

$$[\![X]\!]_\varepsilon = \varepsilon(X) \qquad\qquad [\![\sigma_1 \to \sigma_2]\!]_\varepsilon = [\![\sigma_1]\!]_\varepsilon \Rightarrow [\![\sigma_2]\!]_\varepsilon \qquad\qquad [\![\forall(X)\sigma]\!]_\varepsilon = \bigcap_{R:A \not\leftrightarrow B}^{SAT} [\![\sigma]\!]_{\varepsilon[R/X]}$$

Proposition 9. *The previous definition is proper, that is:*

1. *for all type expressions σ and all per environments η:* $\quad [\![\sigma]\!]_\eta \in Obj(\mathbf{PER})$
2. *for all type expressions σ and all relation environments ε:* $\quad [\![\sigma]\!]_\varepsilon : [\![\sigma]\!]_{\varepsilon\uparrow} \not\leftrightarrow [\![\sigma]\!]_{\varepsilon\downarrow}$

This proposition easily follows from Proposition 7 together with the remark that product and exponentiation in **SAT** extend those defined in **PER**. 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 10 (Substitution Lemma for Types). *For all type expressions σ and τ and all relation assignments ε:*

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

Typings. In what follows, we consider mostly derivable judgements, and often write judgement instead of derivable judgement. We interpret the typing judgements $E \vdash_F M : \sigma$ by functions

$$[\![E \vdash_F M : \sigma]\!] : PerAssign \to ValAssign \to \bigcup_{A \in Obj(PER)} Q(A)$$

where $ValAssign$ is the domain of *value assignments*, which map each term variable to an equivalence class of some per. Typically ρ denotes a value assignment.

Definition 11.

- A value assignment ρ satisfies an environment E with respect to a per assignment η iff, for all $(x : \sigma) \in E$, $\rho(x) \in Q([\![\sigma]\!]_\eta)$. 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 $(x : \sigma) \in E$, $\rho_1(x)\ [\![\sigma]\!]_\varepsilon\ \rho_2(x)$. This is denoted $\rho_1, \rho_2 \models_\varepsilon E$.

Definition 12. 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. 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]). It is usually stated for **PER** models over a λ-model D, but the proof actually uses only the total combinatory algebra structure: the totality is used to make sure that the interpretation of untyped terms in D is always defined.

Theorem 13. *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 14 (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 $\rho_2^\star$.

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 15 (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:

$$\llbracket E \vdash_F M : \sigma \rrbracket_{\eta\rho} = \left[\llbracket erase(M) \rrbracket_{\rho^\star}\right]_{\llbracket \sigma \rrbracket_\eta}$$

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

Corollary 16 (Abstraction Theorem). *For all typing judgements $E \vdash_F M : \sigma$ and all assignments ε, ρ_1, and ρ_2 such that $\rho_1, \rho_2 \models_\varepsilon E$:*

$$\left(\llbracket E \vdash_F M : \sigma \rrbracket_{\varepsilon\uparrow\rho_1}\right) \llbracket \sigma \rrbracket_\varepsilon \left(\llbracket E \vdash_F M : \sigma \rrbracket_{\varepsilon\downarrow\rho_2}\right)$$

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

Lemma 17 (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:*

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

Equalities. The **SAT** model is sound with respect to the equality rules of F.

Theorem 18 (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:*

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

4.2 Categorical Models

In the previous subsection we have assumed 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 [See87].

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 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 (denoted by p from now) 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.

SAT as a PL-category. We now recast the **SAT** model as a PL-category.

Definition 19 (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

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

and then

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

Definition 20 (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 realizes uniformly Φ in the sense that for all $\overline{R} \in p$, (n, n) realizes $\Phi(\overline{R})$ in $Mor(\mathbf{SAT})$.

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.

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

- $(F \times G)(\overline{R}) = F(\overline{R}) \times G(\overline{R})$
- $F^G(\overline{R}) = F(\overline{R})^{G(\overline{R})}$

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

We interpret quantification by the intersection operator introduced in Definition 6.

Definition 21 (Indexed Adjunction). Given an object L of $\mathbf{G}(p+1)$, we define

$$\forall(p)(L) = \bigcap_{R:A \not\leftrightarrow B}^{SAT} L[-, R]$$

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.

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

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

Categorical Interpretation of Types and Terms. Using the standard machinery of categorical semantics over an indexed category it is straightforward to define the interpretation of the judgements of system F. For more details see [AM92].

Given a type judgement $E \vdash_F \sigma$ with p type variables declared in E, the semantics of σ in E, denoted by $[\![\sigma]\!]_p$, is an object of the fiber $\mathbf{G}(p)$, that is, an arrow $Obj(\mathbf{SAT})^p \to Obj(\mathbf{SAT})$. We do the same for environments, so if $\vdash_F E$ is an environment judgment with p type variables declared in E, then its semantics, denoted by $[\![\vdash_F E]\!]_p$, is an element of the fiber $\mathbf{G}(p)$, namely the product of the semantics of the types of the term variables declared in it. Finally, the semantics of a typing judgement $E \vdash_F M : \sigma$ with p type variables declared in E, denoted by $[\![E \vdash_F M : \sigma]\!]_p$, is a morphism in the fiber $\mathbf{G}(p)$ from $[\![\vdash_F E]\!]_p$ to $[\![\sigma]\!]_p$.

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.

Proposition 22. *In the case that the combinatory algebra D is total the categorical model coincides with the concrete one.*

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.

5.1 Related Types

Since type and relation variables are both present explicitly in system $\mathcal{R}$, 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 interpret the related types judgements by functions of type:

$$\left[\!\!\left[E \vdash_{\mathcal{R}} \begin{matrix} \sigma \\ \mathcal{S} \\ \tau \end{matrix} \right]\!\!\right] : \mathit{RelAssign} \to \mathit{ValAssign} \to \mathit{Obj}(\mathbf{SAT})$$

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 23. Let $\mathit{FRel} : \mathit{Mor}(\mathbf{PER}) \to \mathit{Obj}(\mathbf{SAT})$ be the function that maps an $f : A \to B$ to the relation $\mathit{FRel}(f) : A \not\leftrightarrow B$ such that:

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

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

Definition 24. Given a derivable environment judgement $\vdash_{\mathcal{R}} E$ and a relation assignment η, we say that η is an assignment for E iff:

$$\text{for all} \begin{pmatrix} X \\ \mathcal{W} \\ Y \end{pmatrix} \in E, \quad \eta(\mathcal{W}) : \eta(X) \not\leftrightarrow \eta(Y).$$

Definition 25. 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:

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

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:

$$\left[\!\!\left[E', \begin{matrix}X\\ \mathcal{W}\\ Y\end{matrix}, E'' \vdash_{\mathcal{R}} \begin{matrix}X\\ \mathcal{W}\\ Y\end{matrix} \right]\!\!\right]_{\eta\rho} = \eta(\mathcal{W})$$

$$\left[\!\!\left[E', \begin{matrix}X\\ \mathcal{W}\\ Y\end{matrix}, E'' \vdash_{\mathcal{R}} X \right]\!\!\right]_{\eta\rho} = \eta(X) \qquad \left[\!\!\left[E', \begin{matrix}X\\ \mathcal{W}\\ Y\end{matrix}, E'' \vdash_{\mathcal{R}} Y \right]\!\!\right]_{\eta\rho} = \eta(Y)$$

$$\left[\!\!\left[E \vdash_{\mathcal{R}} \begin{matrix}\sigma_1 \to \tau_1\\ \mathcal{S}_1 \to \mathcal{S}_2\\ \sigma_2 \to \tau_2\end{matrix} \right]\!\!\right]_{\eta\rho} = \left[\!\!\left[E \vdash_{\mathcal{R}} \begin{matrix}\sigma_1\\ \mathcal{S}_1\\ \sigma_2\end{matrix} \right]\!\!\right]_{\eta\rho} \Rightarrow \left[\!\!\left[E \vdash_{\mathcal{R}} \begin{matrix}\tau_1\\ \mathcal{S}_2\\ \tau_2\end{matrix} \right]\!\!\right]_{\eta\rho}$$

$$a \left[\!\!\left[E \vdash_{\mathcal{R}} \begin{matrix}\forall(X).\sigma_1\\ \forall(\mathcal{W}).\mathcal{S}\\ \forall(Y).\sigma_2\end{matrix} \right]\!\!\right]_{\eta\rho} b \text{ iff } \begin{cases} a\,[\![\forall(X).\sigma_1]\!]^F_{\eta\ddagger}\, a \\ \text{for all } R : A \not\leftrightarrow B \qquad a \left[\!\!\left[E, \begin{matrix}X\\ \mathcal{W}\\ Y\end{matrix} \vdash_{\mathcal{R}} \begin{matrix}\sigma_1\\ \mathcal{S}\\ \sigma_2\end{matrix} \right]\!\!\right]_{\tilde{\eta}\rho} b \\ \qquad (\text{where } \tilde{\eta} = \eta[R, A, B/\mathcal{W}, X, Y]) \\ b\,[\![\forall(Y).\sigma_2]\!]^F_{\eta\ddagger}\, b \end{cases}$$

$$\left[\!\!\left[E \vdash_{\mathcal{R}} \begin{matrix}\sigma_1\\ < M >\\ \sigma_2\end{matrix} \right]\!\!\right]_{\eta\rho} = FRel([\![E_F \vdash_F M : \sigma_1 \to \sigma_2]\!]^F_{\eta\ddagger,\rho})$$

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.

The presence of functional relations prevents us from using the intersection operator of Definition 6 since it is no longer true that $F(\overline{S}) : F(\overline{A}) \not\leftrightarrow F(\overline{B})$. Indeed, consider the case where the function F is constantly equal to a functional relation: it does not map pers to pers. Instead, we have given a direct, pointwise definition of the meaning of intersection.

Theorem 26 (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* $\rho \models_\eta E$*:*

$$\left[\!\!\left[E \vdash_{\mathcal{R}} \begin{matrix}\sigma_1\\ \mathcal{S}\\ \sigma_2\end{matrix} \right]\!\!\right]_{\eta\rho} : [\![\sigma_1]\!]^F_{\eta\ddagger} \not\leftrightarrow [\![\sigma_2]\!]^F_{\eta\ddagger}$$

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

$$\left[\!\!\left[E \vdash_{\mathcal{R}} \begin{array}{c} \sigma \\ \sigma^* \\ \sigma \end{array} \right]\!\!\right]_{\eta\rho} = [\![\sigma]\!]^{F}_{\eta\ddagger}$$

This proposition is a corollary of a more general statement:

Lemma 28. *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 $\overline{S} : \overline{A} \not\rightarrow \overline{B}$ of relations, we have:*

$$\left[\!\!\left[E, \begin{array}{c} \overline{X'} \\ \overline{\mathcal{W}} \\ \overline{X''} \end{array} \vdash_{\mathcal{R}} \begin{array}{c} \sigma[\overline{X'}/\overline{X}] \\ \sigma^*[\overline{\mathcal{W}}/\overline{X}] \\ \sigma[\overline{X''}/\overline{X}] \end{array} \right]\!\!\right]_{\eta[\overline{S},\overline{A},\overline{B}/\overline{\mathcal{W}},\overline{X'},\overline{X''}],\rho} = [\![\sigma]\!]^{F}_{\eta\ddagger[\overline{S}/\overline{X}]}$$

As usual, we have also a substitution lemma:

Lemma 29 (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:

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

where

$$\left[\!\!\left[E \vdash_{\mathcal{R}} \begin{array}{c} \overline{\tau_1} \\ \overline{\mathcal{S}} \\ \overline{\tau_2} \end{array} \right]\!\!\right]_{\eta,\rho} = \overline{S} : \overline{A} \not\rightarrow \overline{B}$$

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:

$$\left[\!\!\left[E \vdash_{\mathcal{R}} \begin{array}{c} M : \sigma_1 \\ \mathcal{S} \\ N : \sigma_2 \end{array} \right]\!\!\right] : \mathit{RelAssign} \rightarrow \mathit{ValAssign} \rightarrow \bigcup_{A,B \in Obj(PER)} Q(A) \times Q(B)$$

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

Finally, we obtain that the two components of this interpretation are related:

Theorem 30 (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:

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

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 $\mathcal{R}$. The concrete interpretation of $\mathcal{R}$ carried out here establishes the soundness of $\mathcal{R}$. It would 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.

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.

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.

[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.

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

[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.

[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.

[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.