Formal Parametric Polymorphism

Martín Abadi
Digital Equipment Corporation
Systems Research Center

Luca Cardelli
Digital Equipment Corporation
Systems Research Center

Pierre-Louis Curien
CNRS
Ecole Normale Supérieure

Abstract

A polymorphic function is parametric if its behavior does not depend on the type at which it is instantiated. Starting with Reynolds's work, the study of parametricity is typically semantic. In this paper, we develop a syntactic approach to parametricity, and a formal system that embodies this approach, called system *R* . Girard's system F deals with terms and types; *R* is an extension of F that deals also with relations between types.

In *R* , it is possible to derive theorems about functions from their types, or "theorems for free", as Wadler calls them. An easy "theorem for free" asserts that the type $\forall(X)X\rightarrow$Bool contains only constant functions; this is not provable in F. There are many harder and more substantial examples. Various metatheorems can also be obtained, such as a syntactic version of Reynolds's abstraction theorem.

1. Explicit relations

A polymorphic function is parametric if its behavior does not depend on the type at which it is instantiated [Strachey 1967]. A function that reverses lists, for example, is parametric because it does not look at the types of the elements of the lists given as inputs. There are important non-parametric polymorphic functions, such as a print function that maps values of any type to text representations. With this caveat, it can be argued that "truly" polymorphic functions are parametric, and in any case it is the parametric polymorphic functions that form the core of languages such as ML [Milner, Tofte, Harper 1989].

Reynolds's work provides a precise counterpart to the informal definition of parametricity just given [Reynolds 1983; Ma, Reynolds 1991]. Reynolds's abstraction theorem concerns a language similar to Girard's system F [Girard, Lafont, Taylor 1989], and implies that the instances of a polymorphic function at different types behave in "related" ways. For example, let f be an expressible function of type $\forall(X)X\rightarrow X$ (the type of the identity function), and let f(A) and f(B) be its instantiations at types A and B, respectively. In this case, the theorem says that, for any relation S between A and B, if $\langle a,b\rangle\in S$ then $\langle f(A)(a),f(B)(b)\rangle\in S$. A bit of calculation reveals that the identity function is the only function with this property, so f must be the identity function. This is what Wadler would call a "theorem for free" [Wadler 1989]: a result about a function that is obtained by examining only its type, and not its code. Reynolds's results about his system suggest that, more generally, one should view a function as parametric if and only if its instances at related types behave in related ways.

In the preceding discussion, functions, types, and relations are all semantic objects. Reynolds's results concern the models of polymorphic languages, such as F, and only indirectly their syntax. Similarly, Wadler's free theorems concern semantic objects in these models, and do not immediately refer to the world of syntax, where they might serve in proving properties of programs.

In this paper we develop a syntactic approach to parametricity. This approach is embodied in an extension of F, called R, where relations between types are constructed and treated formally. In R, the free theorems can be stated and proved in a logical framework and without reference to particular classes of models. Several of these free theorems come from Wadler's work, and we hope that our detailed, formal treatment illuminates their proofs; others seem to be new and intriguing. Various metatheorems about R can also be obtained, for example a syntactic version of the abstraction theorem. In all cases our results are not limited to closed terms.

The study of R seems to help in clarifying the notions of parametricity and the properties of parametric models. Semantic explorations steer a difficult course between heavyweight categorical constructions and lightweight fuzzy explanations; in contrast, we use a precise, elementary syntax. With this syntax, it is possible to formulate results and conjectures that relate the intuitive definition of parametricity ("types are not needed at run time") with Reynolds's mathematical one.

The remainder of this introduction contains an informal technical introduction and a comparison with a few recent related works. Sections 2 and 3 introduce R, its theory, and then some of the free theorems. In the conclusions we discuss further work, briefly touching on the semantics of R. The appendix contains the complete set of rules of the system. It also describes a proof, due to Hasegawa, of the inconsistency of an earlier version of R.

1.1 Parametricity

As an introduction to parametricity and to R, we give an example: we prove that all parametric functions of type $\forall(X)X\rightarrow Bool$ are constant. (Here Bool is the type of booleans as encoded in F: $\forall(X)X\rightarrow X\rightarrow X$.) We start with an informal discussion of the functions of this type, then make the reasoning a little more precise, and later, in section 1.2, we introduce the judgments and some of the rules of R, which enable us to formalize the reasoning for this and other free theorems.

Throughout, we focus on total functions. All computations are assumed to terminate. It is well known that the interaction of recursion and parametricity is not entirely trivial, and clearly some strictness conditions should be added to the relations we consider below in the presence of recursion.

At the very least, a function f in $\forall(X)X\rightarrow$Bool maps values of any type to booleans. More precisely:

(i) If A is a type and b has type A,
then f(A) maps b to a boolean.

The primary examples of functions that satisfy (i) are the constant functions whose instances map any input to either true or false. But, in some models, there are additional functions that satisfy (i) and that may be considered as belonging to $\forall(X)X\rightarrow$Bool, such as a function zero-p with instances that always map 0 to true and any other input to false. It is hard to code these additional functions in such a way that a type-checker would accept them, and the resulting code requires the use of types at run time. Hence, none of these functions can be considered parametric. Only the constant functions remain.

The sort of discussion of parametric functions that we just went through, to exclude for example zero-p, is vague and not entirely satisfactory; it depends on the use of particular models and on implementation intuitions. Reynolds's more satisfactory approach is based on relations between types. But before we discuss relations in general, it is convenient to introduce the per model [Longo, Moggi 1991], which is based on special relations.

In per semantics, types are interpreted as pers, that is, as partial equivalence relations (symmetric and transitive relations on the universe of values). Intuitively, b and c are related by the type A if they are equal elements of A, and in particular b is related to itself if it is an element of A. For example, A may be the type of all records with a field n of type Nat, and b and c may be two records that have a field n with the value 3, but differ on other fields; in this case b and c are related by A. We write b[A]c for $\langle b,c\rangle\in A$.

Given two pers A and B, the set of all functions from A to B is also represented as a per:

$$f[A\rightarrow B]g \text{ iff for all } x, y, \text{ if } x[A]y \text{ then } f(x)[B]g(y)$$

That is, two functions are equal in $A\rightarrow B$ if they map inputs equal in A to results equal in B. Universal quantification is interpreted as intersection, with bound variables ranging over pers.

For example, in the language of pers, the condition for f to be in the type $\forall(X)X\rightarrow$Bool is that f[$\forall(X)X\rightarrow$Bool]f. It follows that f(A)[$A\rightarrow$Bool]f(A), for all A, and then:

(ii) If b and c are equal as elements of A,
then f(A) maps b and c to the same boolean.

In the per model, the only functions of type $\forall(X)X\rightarrow$Bool are the two obvious constant functions (but this does not follow from (ii) alone). When A is a record type, for instance, requirement (ii) implies that f(A)(b) cannot depend on fields in b not shown in the definition of A.

Reynolds's work does not assume a per semantics, but his notion of parametricity can be seen as a strengthening of requirement (ii); in this example, it says:

(iii) If *S* is a relation between types A and B, with a in A, b in B, and *S* relating a and b,
then f(A)(a) and f(B)(b) are equal booleans.

Requirement (ii) corresponds to the special case where A = B, and *S* is the identity relation on A.

Intuitively, as Reynolds suggests, we may think of A and B as two different representations of the same type, and of a and b as two different representations of the same value; then requirement (iii) means that the function f respects representation abstractions and, for each input, f returns results independently of the representation of the input.

In order to state the general form of (iii), we extend the operations $\rightarrow$ and $\forall$. They are defined on arbitrary relations just as they were on pers, except that the variables bound by $\forall$ (now written *U*, *V*, *W*, *X*, ...) range over all relations, not just over pers. With this notation, there is a natural relation A* associated with each type expression A. This is the relation denoted by the type expression A where all quantified variables are interpreted as ranging over arbitrary relations rather than over pers. For example, the relation $(\forall(X)X\rightarrow\text{Bool})^*$ is $\forall(W)W \rightarrow \text{Bool}^*$, and $(\forall(X)X\rightarrow Y)^*$ is $\forall(W)W \rightarrow Y$.

The general form of (iii) can now be stated:

> An element of type A is related to itself by the associated relation A*.

Essentially, Reynolds's abstraction theorem says that all the functions expressible in F satisfy this property. Thus, according to the abstraction theorem, if f is expressible with type $\forall(X)X\rightarrow\text{Bool}$, then f must be related to itself by $\forall(W)W \rightarrow \text{Bool}^*$. It follows that if A and B are two types and S is a relation between them, then f(A) and f(B) are related in $S\rightarrow\text{Bool}^*$, and so if S relates a and b it follows that Bool* relates f(A)(a) and f(B)(b), as stated in (iii).

With (iii), it is simple to prove that constant functions are the only elements of the type considered: Let f be a function of this type, let A be a type, and let S be the relation between A and Bool that associates every element of A with true. Then f(A) and f(Bool) are related by $S\rightarrow\text{Bool}^*$, and if a is an element of A then f(A)(a) and f(Bool)(true) are related by Bool*, that is, f(A)(a) is equal to the fixed boolean f(Bool)(true), independently of A and a. By extensionality, f is one of the two constant functions. (The use of Bool and true is arbitrary; they can be replaced with any other closed type and closed term of that type.)

Reynolds's notion of parametricity is not limited to binary relations. We consider only binary relations for simplicity, and because they are powerful enough in deriving all the familiar consequences of parametricity.

1.2 Formalizing parametricity

Reynolds's relational approach to parametricity lends itself to a syntactic treatment. System $\mathcal{R}$ provides such a treatment, based on judgments and rules in the style of those of F.

Three judgments generalize those of system F (described in the appendix):

$\vdash E$	E is a legal environment
$E \vdash \begin{array}{c} A \\ R \\ B \end{array}$	R is a relation between types A and B in E
$E \vdash \begin{array}{c} a : A \\ R \\ b : B \end{array}$	R relates a of type A and b of type B in E

An equality relation on values is not needed. Instead of writing that b and c are equal in A, we can promote the type A to a relation A* (between A and A; intuitively, the identity relation) and write that A* relates b and c. As a consequence we write:

$$E \vdash \begin{array}{c} b : A \\ A^* \\ c : A \end{array} \qquad \text{corresponding to the F judgment} \quad E \vdash b = c : A$$

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

$\begin{array}{c} X \\ W \\ Y \end{array}$	W is a relation variable between type variables X (domain) and Y (codomain)
$\begin{array}{c} x : A \\ R \\ y : B \end{array}$	the variables x and y have types A and B, respectively, and are related by R

Using these judgments, we now review some of the central rules of R . We start with rules that imitate those of F for $\rightarrow$ and $\forall$.

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

$$\dfrac{E,\ \begin{array}{c} x : A \\ \mathit{R} \\ x' : A' \end{array} \vdash \begin{array}{c} b : B \\ \mathit{S} \\ b' : B' \end{array} \qquad E \vdash \begin{array}{c} B \\ \mathit{S} \\ B' \end{array} \qquad \begin{array}{c} x \notin b' \\ x' \notin b \end{array}}{E \vdash \begin{array}{c} \lambda(x : A)b : A \rightarrow B \\ \mathit{R} \rightarrow \mathit{S} \\ \lambda(x' : A')b' : A' \rightarrow B' \end{array}} \qquad \dfrac{E \vdash \begin{array}{c} b : A \rightarrow B \\ \mathit{R} \rightarrow \mathit{S} \\ b' : A' \rightarrow B' \end{array} \qquad E \vdash \begin{array}{c} a : A \\ \mathit{R} \\ a' : A' \end{array}}{E \vdash \begin{array}{c} b(a) : B \\ \mathit{S} \\ b'(a') : B' \end{array}}$$

These rules follow the same pattern as the F rules:

$$\dfrac{E,\ x : A \vdash b : B}{E \vdash \lambda(x : A)b : A \rightarrow B} \qquad \dfrac{E \vdash b : A \rightarrow B \qquad E \vdash a : A}{E \vdash b(a) : B}$$

The introduction rule says: Assume that if R relates x of type A and x' of type A', then S relates b of type B and b' of type B'. Then $\mathit{R} \rightarrow \mathit{S}$, a relation between A$\rightarrow$B and A'$\rightarrow$B', relates the functions λ(x:A)b of type A$\rightarrow$B and λ(x':A')b' of type A'$\rightarrow$B'. An extra hypothesis that S relates B and B' is added to simplify our technical lemmas. The elimination rule works in the opposite direction, applying related functions to related arguments and obtaining related results.

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

$$\dfrac{E,\ \begin{array}{c} X \\ \mathit{W} \\ X' \end{array} \vdash \begin{array}{c} b : B \\ \mathit{S} \\ b' : B' \end{array} \qquad \begin{array}{c} X \notin b', B', \mathit{S} \\ X' \notin b, B, \mathit{S} \end{array}}{E \vdash \begin{array}{c} \lambda(X)b : \forall(X)B \\ \forall(\mathit{W})\mathit{S} \\ \lambda(X')b' : \forall(X')B' \end{array}} \qquad \dfrac{E \vdash \begin{array}{c} b : \forall(X)B \\ \forall(\mathit{W})\mathit{S} \\ b' : \forall(X')B' \end{array} \qquad E \vdash \begin{array}{c} C \\ \mathit{T} \\ C' \end{array}}{E \vdash \begin{array}{c} b(C) : B\{X \leftarrow C\} \\ \mathit{S}\{\mathit{W} \leftarrow \mathit{T}\} \\ b'(C') : B'\{X' \leftarrow C'\} \end{array}}$$

These rules follow the same pattern as the F rules:

$$\dfrac{E,\ X \vdash b : B}{E \vdash \lambda(X)b : \forall(X)B} \qquad \dfrac{E \vdash b : \forall(X)B \qquad E \vdash C}{E \vdash b(C) : B\{X \leftarrow C\}}$$

The introduction rule says: Assume that if W is a relation between types X and X', then S relates b of type B and b' of type B'. Then $\forall(\mathit{W})\mathit{S}$, a relation between $\forall$(X)B and $\forall$(X')B', relates the polymorphic terms λ(X)b of type $\forall$(X)B and λ(X')b' of type $\forall$(X')B'. Again, the elimination rule works in the opposite direction: it applies two related polymorphic terms to related types, obtaining related instances.

The system has three rules for variables:

(Rel Val x*R* y)

$$\dfrac{\vdash E',\ \begin{array}{c} x : A \\ \mathit{R} \\ y : B \end{array},\ E''}{E',\ \begin{array}{c} x : A \\ \mathit{R} \\ y : B \end{array},\ E'' \vdash \begin{array}{c} x : A \\ \mathit{R} \\ y : B \end{array}}$$

(Rel Val *R* x)

$$\dfrac{\vdash E',\ \begin{array}{c} x : A \\ \mathit{R} \\ y : B \end{array},\ E''}{E',\ \begin{array}{c} x : A \\ \mathit{R} \\ y : B \end{array},\ E'' \vdash \begin{array}{c} x : A \\ A^* \\ x : A \end{array}}$$

(Rel Val *R* y)

$$\dfrac{\vdash E',\ \begin{array}{c} x : A \\ \mathit{R} \\ y : B \end{array},\ E''}{E',\ \begin{array}{c} x : A \\ \mathit{R} \\ y : B \end{array},\ E'' \vdash \begin{array}{c} y : B \\ B^* \\ y : B \end{array}}$$

The first rule is straightforward. The other two formalize our parametricity condition. For our example of section 1.1, these two rules imply that if a variable f has type $\forall$(X)X$\rightarrow$Bool then f is related to itself by $\forall(\mathit{W})\mathit{W} \rightarrow \mathrm{Bool}^*$. From here we can apply the elimination rules for $\forall$ and $\rightarrow$, and obtain (iii). This kind of reasoning is common in our examples of section 3. (In [Abadi, Cardelli, Curien 1993], we had adopted a different formalization of parametricity; it turned out to be inconsistent, see the appendix.)

The preceding rules, together with the rules of β and η conversion, form the core of the fragment of R that deals with relations built from variables, $\rightarrow$, and quantifiers. This basic system, called R^0, is sufficient to encode F:

$$\text{if F proves } E \vdash a : A \text{ then } R^0 \text{ proves } E \vdash \begin{array}{c} a \,:\, A \\ A^* \\ a \,:\, A \end{array}$$

This is a syntactic version of Reynolds's identity extension property. For closed terms, it can be proved without appeal to parametricity, that is, without using (Rel Val *R* x) or (Rel Val *R* y). We also obtain all F equalities:

$$\text{if F proves } E \vdash a = a' : A \text{ then } R^0 \text{ proves } E \vdash \begin{array}{c} a \,:\, A \\ A^* \\ a' \,:\, A \end{array}$$

But R^0 is not very powerful without some additional methods for constructing relations. In fact, under the encoding just suggested, R^0 is a conservative extension of F.

Until now, the relational constructions have followed closely the ordinary type constructions. In addition we allow relations defined from functions, obtaining a system called R^1:

$$\frac{E \vdash b : A \to B \qquad E \vdash a : A}{E \vdash \begin{array}{c} a \,:\, A \\ \langle b \rangle \\ b(a) \,:\, B \end{array}} \qquad\qquad \frac{E \vdash \begin{array}{c} a \,:\, A \\ \langle b \rangle \\ c \,:\, B \end{array} \qquad E \vdash b : A \to B}{E \vdash \begin{array}{c} b(a) \,:\, B \\ B^* \\ c \,:\, B \end{array}}$$

where $E \vdash a : A$ is an abbreviation for $E \vdash \begin{array}{c} a : A \\ A^* \\ a : A \end{array}$ (and similarly for b).

With these rules, terms can be turned into relations: any function b from A to B can be seen as a relation $\langle b \rangle$ between A and B, intuitively the graph of the function. The rules for functional relations have no analogue in F. Our formalism yields the results typically associated with parametricity only when we include rules for constructing functional relations. Functional relations are often useful for obtaining free theorems; for the example of section 1.1, the relevant functional relation is a constant one, obtained from the function from A to Bool that maps any a in A to true.

One can easily imagine mechanisms for defining relations beyond taking the graphs of functions. We have not yet found examples where these mechanisms are needed.

1.3 Related work

By now there are many papers on semantic aspects of parametricity ([Bainbridge, *et al.* 1990; Hasegawa 1991; Ma, Reynolds 1991; Hasegawa 1992; Mitchell, Scedrov 1992], and others). On the other hand, the syntactic study of parametricity is rather new. Some recent work is related to ours.

Mairson advocated and developed a syntactic approach to parametricity in order to provide careful formal versions of some of Wadler's theorems [Mairson 1991]. Mairson's approach consists in translating a polymorphic language into a second-order logic. Because the second-order logic used is fairly weak, induction arguments become necessary in some of the proofs; our proofs, like Wadler's, do not rely on induction. Mairson treated a system with implicit typing; this stands in contrast with our approach where types and relations are treated explicitly. The resulting formalisms have very different properties.

Cardelli *et al.* have defined $F_{<:}$, an extension of F with subtyping [Cardelli, *et al.* 1991]. Curiously, the rules for $F_{<:}$ capture some aspects of parametricity, but they do not provide a full account of it.

Ma suggested another syntactic approach to parametricity [Ma 1992]. It is based on encoding relations using subtyping. The power of Ma's system seems to be less understood; there is also some difficulty in finding a model for all the desired subtyping rules.

Longo, Milsted, and Soloviev investigated parametricity in a system like F with just one new rule (a special case of one of the rules of $F_{<:}$) [Longo, Milstead, Soloviev 1993]. The system is weaker than R , and leads to different sorts of results.

Finally, Plotkin and Abadi explore an alternative formalization of parametricity closer in spirit to Mairson's [Plotkin, Abadi 1993]. That paper describes a second-order logic with an axiom of parametricity; the logic is not an extension of system F, like R , but rather a logic about system F terms.

2. Formal parametricity

In this section we describe our formalization of parametricity. We aim at a hypothetical system that would be sufficient to prove all the desired parametricity properties of polymorphic programs. Our current approximations are called R^0 and R^1; they are treated in sections 2.1 and 2.2 respectively.

The system R^0 is a rather weak system of pure relations with relational constructions induced by the type constructions of F. A number of technical lemmas can be proved for R^0, and these lead to several interesting metatheorems. For example, a suitable encoding of F in R^0 yields all F typings and F equalities. In addition, R^0 is a conservative extension of F for typing and equality derivations. The abstraction theorem and the identity extension property hold in R^0 but they are not very useful (as the conservativity result indicates) without some additional means for constructing relations. Hence, we extend R^0 with functional relations, obtaining R^1. Relation expressions become dependent on value expressions, and the syntactic properties of the system become slightly more complex. Fortunately, most R^0 metatheorems extend easily to R^1, simply because R^0 derivations are also R^1 derivations. As a typing system, R^1 is still conservative over F, but new equations are provable.

Unless otherwise indicated, the proofs of this section are structural inductions on derivations. The proofs are long but not difficult, if carried out in the order of presentation of the claims. We point out the crucial dependencies.

2.1 Relational interpretation of system F (system R^0)

We use $\vdash^{F}$ for derivations in F, and $\vdash^{R^0}$ (or simply $\vdash$ in this section) for derivations in R^0. Our formalization of system F is listed in the appendix; note the explicit form of the equality judgments ($E \vdash^{F} a = b : A$), which include type and environment information. The complete rules for system R^0 are also listed in the appendix.

In section 2.1.1 we establish the most basic metaproperties of system R^0. In section 2.1.2 we relate typing in F with typing in R^0. In section 2.1.3 we state more structural lemmas for R^0. In section 2.1.4 we show the soundness and completeness of F equality in R^0, that is, we show that F and R^0 prove the same equations. The main result of the section is theorem (Partial relational interpretation of F), which is split across sections 2.1.2 and 2.1.4. Remarkably, this theorem yields as corollaries both the abstraction theorem and the identity extension property.

2.1.1 Basic structural lemmas

Notation

- We write dom(E) for the domain of E, that is, the collection of all the variables introduced by an environment E.
- α-identifications. As usual, we identify terms up to renaming of bound variables. These identifications can be made directly in the syntax, that is, without knowing whether the terms involved are the product of formal derivations in the system. Environments, however, are not identified up to renaming of variables in their domain; environment variables are kept distinct by construction. A more formal approach would use de Bruijn indices for free and bound variables [de Bruijn 1972].

- We use the following metavariables: x,y,z range over value variables; X,Y,Z range over type variables; $\mathcal{W}, \mathcal{X}$ range over relation variables; a,b,c,d range over value terms; A,B,C,D range over type terms; $\mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{U}$ range over relation terms; E ranges over environments.
- We write $\emptyset$ for the empty environment; we often omit this symbol.
- We use J to stand for either $\begin{array}{c} \mathrm{A} \\ \mathcal{R} \\ \mathrm{B} \end{array}$ or $\begin{array}{c} \mathrm{a} : \mathrm{A} \\ \mathcal{R} \\ \mathrm{b} : \mathrm{B} \end{array}$.
- By $J\{\xi \leftarrow \tau\}$ we indicate the substitution of τ for ξ in every component of J, where ξ can be one of x,X,$\mathcal{W}$, and τ can be one of a,A,$\mathcal{R}$. Similarly, $\mathrm{E}\{\xi \leftarrow \tau\}$, with $\xi \notin \mathrm{dom(E)}$, indicates a substitution in every component of the environment E. Note though that $J\{\mathrm{X} \leftarrow \mathcal{R}\}$ may not be well-formed if J contains value terms.
- By $J\begin{Bmatrix} \xi_1 \leftarrow \tau_1 \\ \xi_2 \leftarrow \tau_2 \end{Bmatrix}$ and $J\begin{Bmatrix} \xi_1 \leftarrow \tau_1 \\ \xi_2 \leftarrow \tau_2 \\ \xi_3 \leftarrow \tau_3 \end{Bmatrix}$, where ξ_i are distinct, we indicate simultaneous substitutions per-

formed as above (and similarly for an environment E in place of J, with $\xi_i \notin \mathrm{dom(E)}$).
- For a type A, the relation A* is defined inductively as follows:

$$\begin{array}{lcl} \mathrm{X}^* & \triangleq & \mathcal{X} \\ (\mathrm{A}\rightarrow\mathrm{B})^* & \triangleq & \mathrm{A}^*\rightarrow\mathrm{B}^* \\ (\forall(\mathrm{X})\mathrm{B})^* & \triangleq & \forall(\mathcal{W})(\mathrm{B}^*\{\mathcal{X}\leftarrow\mathcal{W}\}) \end{array}$$

- We use the following abbreviations in order to embed F notation in $\mathcal{R}$:

$$\mathrm{E} \vdash \mathrm{A} \;\triangleq\; \mathrm{E} \vdash \begin{array}{c} \mathrm{A} \\ \mathrm{A}^* \\ \mathrm{A} \end{array} \qquad\qquad \mathrm{E} \vdash \mathrm{a} : \mathrm{A} \;\triangleq\; \mathrm{E} \vdash \begin{array}{c} \mathrm{a} : \mathrm{A} \\ \mathrm{A}^* \\ \mathrm{a} : \mathrm{A} \end{array}$$

$$\vdash \mathrm{E},\ \mathrm{X},\ \mathrm{E'} \;\triangleq\; \vdash \mathrm{E},\ \begin{array}{c} \mathrm{X} \\ \mathcal{X} \\ \mathrm{X'} \end{array},\ \mathrm{E'} \qquad\qquad \mathrm{E},\ \mathrm{X},\ \mathrm{E'} \vdash J \;\triangleq\; \mathrm{E},\ \begin{array}{c} \mathrm{X} \\ \mathcal{X} \\ \mathrm{X'} \end{array},\ \mathrm{E'} \vdash J \qquad \text{where } \mathcal{X}, \mathrm{X'} \text{ are fresh}$$

$$\vdash \mathrm{E},\ \mathrm{x}:\mathrm{A},\ \mathrm{E'} \;\triangleq\; \vdash \mathrm{E},\ \begin{array}{c} \mathrm{x}:\mathrm{A} \\ \mathrm{A}^* \\ \mathrm{x'}:\mathrm{A} \end{array},\ \mathrm{E'} \qquad\qquad \mathrm{E},\ \mathrm{x}:\mathrm{A},\ \mathrm{E'} \vdash J \;\triangleq\; \mathrm{E},\ \begin{array}{c} \mathrm{x}:\mathrm{A} \\ \mathrm{A}^* \\ \mathrm{x'}:\mathrm{A} \end{array},\ \mathrm{E'} \vdash J \qquad \text{where x' is fresh}$$

We start our study of $\mathcal{R}$ with three basic structural lemmas:

Lemma (Renaming)
Assume x',y',X',Y',$\mathcal{W}$' $\notin$ dom(E,E')$\cup${x,y,X,Y,$\mathcal{W}$}. Then:

- $\vdash \mathrm{E}, \begin{array}{c} \mathrm{X} \\ \mathcal{W} \\ \mathrm{Y} \end{array}, \mathrm{E'} \;\Rightarrow\; \vdash \mathrm{E}, \begin{array}{c} \mathrm{X'} \\ \mathcal{W}' \\ \mathrm{Y'} \end{array}, \mathrm{E'}\begin{Bmatrix} \mathrm{X} \leftarrow \mathrm{X'} \\ \mathcal{W} \leftarrow \mathcal{W}' \\ \mathrm{Y} \leftarrow \mathrm{Y'} \end{Bmatrix}$
- $\mathrm{E}, \begin{array}{c} \mathrm{X} \\ \mathcal{W} \\ \mathrm{Y} \end{array}, \mathrm{E'} \vdash J \;\Rightarrow\; \mathrm{E}, \begin{array}{c} \mathrm{X'} \\ \mathcal{W}' \\ \mathrm{Y'} \end{array}, \mathrm{E'}\begin{Bmatrix} \mathrm{X} \leftarrow \mathrm{X'} \\ \mathcal{W} \leftarrow \mathcal{W}' \\ \mathrm{Y} \leftarrow \mathrm{Y'} \end{Bmatrix} \vdash J \begin{Bmatrix} \mathrm{X} \leftarrow \mathrm{X'} \\ \mathcal{W} \leftarrow \mathcal{W}' \\ \mathrm{Y} \leftarrow \mathrm{Y'} \end{Bmatrix}$
- $\vdash \mathrm{E}, \begin{array}{c} \mathrm{x} : \mathrm{A} \\ \mathcal{R} \\ \mathrm{y} : \mathrm{B} \end{array}, \mathrm{E'} \;\Rightarrow\; \vdash \mathrm{E}, \begin{array}{c} \mathrm{x'} : \mathrm{A} \\ \mathcal{R} \\ \mathrm{y'} : \mathrm{B} \end{array}, \mathrm{E'}\begin{Bmatrix} \mathrm{x} \leftarrow \mathrm{x'} \\ \mathrm{y} \leftarrow \mathrm{y'} \end{Bmatrix}$
- $\mathrm{E}, \begin{array}{c} \mathrm{x} : \mathrm{A} \\ \mathcal{R} \\ \mathrm{y} : \mathrm{B} \end{array}, \mathrm{E'} \vdash J \;\Rightarrow\; \mathrm{E}, \begin{array}{c} \mathrm{x'} : \mathrm{A} \\ \mathcal{R} \\ \mathrm{y'} : \mathrm{B} \end{array}, \mathrm{E'}\begin{Bmatrix} \mathrm{x} \leftarrow \mathrm{x'} \\ \mathrm{y} \leftarrow \mathrm{y'} \end{Bmatrix} \vdash J \begin{Bmatrix} \mathrm{x} \leftarrow \mathrm{x'} \\ \mathrm{y} \leftarrow \mathrm{y'} \end{Bmatrix}$

Moreover, the derivations of the conclusions can have the same size as the derivations of the assumptions.

Lemma (Implied judgments)

(1) • $\vdash \mathrm{E}, \mathrm{E'} \;\Rightarrow\; \vdash \mathrm{E}$ • $\mathrm{E}, \mathrm{E'} \vdash J \;\Rightarrow\; \vdash \mathrm{E}$

(2) $\bullet\ \vdash \mathrm{E}, \begin{matrix}\mathrm{X}\\ \mathcal{W}\\ \mathrm{Y}\end{matrix}, \mathrm{E'} \ \Rightarrow\ \mathrm{X}, \mathrm{Y}, \mathcal{W} \notin \mathrm{dom}(\mathrm{E}, \mathrm{E'}),\ \mathrm{X}, \mathrm{Y}, \mathcal{W}$ distinct

$\bullet\ \vdash \mathrm{E}, \begin{matrix}\mathrm{x} : \mathrm{A}\\ \mathcal{R}\\ \mathrm{y} : \mathrm{B}\end{matrix}, \mathrm{E'} \ \Rightarrow\ \mathrm{E} \vdash \begin{matrix}\mathrm{A}\\ \mathcal{R}\\ \mathrm{B}\end{matrix} \ \wedge\ \mathrm{x}, \mathrm{y} \notin \mathrm{dom}(\mathrm{E}, \mathrm{E'}),\ \mathrm{x}, \mathrm{y}$ distinct

Lemma (Weakening)

Assume $\vdash \mathrm{E}, \mathrm{E''}$ and $\mathrm{dom}(\mathrm{E''}) \cap \mathrm{dom}(\mathrm{E'}) = \emptyset$. Then:

$\bullet\ \vdash \mathrm{E}, \mathrm{E'} \ \Rightarrow\ \vdash \mathrm{E}, \mathrm{E''}, \mathrm{E'}$ $\qquad$ $\bullet\ \mathrm{E}, \mathrm{E'} \vdash \mathcal{J} \ \Rightarrow\ \mathrm{E}, \mathrm{E''}, \mathrm{E'} \vdash \mathcal{J}$

2.1.2 From F to $\mathcal{R}^0$ and from $\mathcal{R}^0$ to F (typing)

First, we show the conservativity of $\mathcal{R}^0$ over F for typing. We need a definition for flattening an $\mathcal{R}$ environment E into an F environment $(\mathrm{E})_\mathrm{F}$. The relation part of E is forgotten in $(\mathrm{E})_\mathrm{F}$:

Definition (Environment flattening)

$\bullet\ (\emptyset)_\mathrm{F} = \emptyset$ $\qquad$ $\bullet\ \left(\mathrm{E}, \begin{matrix}\mathrm{X}\\ \mathcal{W}\\ \mathrm{Y}\end{matrix}\right)_\mathrm{F} = (\mathrm{E})_\mathrm{F}, \mathrm{X}, \mathrm{Y}$ $\qquad$ $\bullet\ \left(\mathrm{E}, \begin{matrix}\mathrm{x} : \mathrm{A}\\ \mathcal{R}\\ \mathrm{y} : \mathrm{B}\end{matrix}\right)_\mathrm{F} = (\mathrm{E})_\mathrm{F}, \mathrm{x} : \mathrm{A}, \mathrm{y} : \mathrm{B}$

Theorem (Flattened F derivations from $\mathcal{R}$ derivations) or (Conservativity over F for typing)

(1) $\vdash \mathrm{E} \ \Rightarrow\ \vdash^\mathrm{F} (\mathrm{E})_\mathrm{F}$

(2) $\mathrm{E} \vdash \begin{matrix}\mathrm{A}\\ \mathcal{R}\\ \mathrm{B}\end{matrix} \ \Rightarrow\ (\mathrm{E})_\mathrm{F} \vdash^\mathrm{F} \mathrm{A} \ \wedge\ (\mathrm{E})_\mathrm{F} \vdash^\mathrm{F} \mathrm{B}$

(3) $\mathrm{E} \vdash \begin{matrix}\mathrm{a} : \mathrm{A}\\ \mathcal{R}\\ \mathrm{b} : \mathrm{B}\end{matrix} \ \Rightarrow\ (\mathrm{E})_\mathrm{F} \vdash^\mathrm{F} \mathrm{a} : \mathrm{A} \ \wedge\ (\mathrm{E})_\mathrm{F} \vdash^\mathrm{F} \mathrm{b} : \mathrm{B}$

Conversely, there are several possible encodings of F in $\mathcal{R}^0$. To each type variable X, we associate a fresh type variable X_1 and a fresh relation variable $\mathcal{X}$ between X and X_1. We proceed similarly for value variables; for example, to each x of type A we may associate a fresh x_1 of type A related to x by A*. This enables us to map F environments to $\mathcal{R}^0$ environments. Then, for each use of a type variable X in an F judgment, there will be uses of X, $\mathcal{X}$, or X_1 in a corresponding $\mathcal{R}^0$ judgment. We have some freedom in choosing between X, $\mathcal{X}$, and X_1. We have a similar freedom in the choice of value variables. We can use this freedom to provide several different encodings of F in $\mathcal{R}^0$.

After some technical definitions, we present our most general encoding in theorem (Partial relational interpretation of F). We obtain two simpler encodings as corollaries.

Definition (Decorating variables)
Let Ξ be a set of type and value variable names. The translation:

$[\text{-}]_n^\Xi$

decorates with a numerical subscript *n* every variable not belonging to Ξ but occurring free in an expression. For example:

$$[\lambda(x:\forall(Y)X\to Y)\ y(z)]_1^{\{y\}} = \lambda(x:\forall(Y)X_1\to Y)\ y(z_1)$$

We assume that variable decorations are always chosen so as not to introduce variable clashes.

Definition (Types as relations)
Let Ξ be a set of type and value variables. The translation $[A]_{\mathit{R}}^{\Xi}$ is defined as follows:

$$\begin{array}{lcl}
[X]_{\mathit{R}}^{\Xi} & = & \mathit{X} \quad (X\notin\Xi)\\
[X]_{\mathit{R}}^{\Xi} & = & X \quad (X\in\Xi)\\
[A\to B]_{\mathit{R}}^{\Xi} & = & [A]_{\mathit{R}}^{\Xi}\to[B]_{\mathit{R}}^{\Xi}\\
[\forall(X)B]_{\mathit{R}}^{\Xi} & = & \forall(\mathit{X})\left([B]_{\mathit{R}}^{\Xi\setminus\{X\}}\right)
\end{array}$$

Thus, the translation transforms type quantifiers into relation quantifiers, and free type variables not belonging to Ξ into free relation variables. In particular, if $E\vdash A$ and Ξ=dom(E), then $[A]_{\mathit{R}}^{\Xi}$ is A*.

Definition (Environment decoration)
Let Ξ be a set of type and value variables and let E be an F environment. The translation $[E]_{\begin{smallmatrix}\Xi\\ \mathit{R}\\ 1\end{smallmatrix}}$ is defined as follows:

$$\begin{array}{lcl}
[\varnothing]_{\begin{smallmatrix}\Xi\\ \mathit{R}\\ 1\end{smallmatrix}} & = & \varnothing\\
[E,X]_{\begin{smallmatrix}\Xi\\ \mathit{R}\\ 1\end{smallmatrix}} & = & [E]_{\begin{smallmatrix}\Xi\\ \mathit{R}\\ 1\end{smallmatrix}},\ \begin{array}{c}X\\ \mathit{X}\\ X_1\end{array}\\
[E,\ x:A]_{\begin{smallmatrix}\Xi\\ \mathit{R}\\ 1\end{smallmatrix}} & = & [E]_{\begin{smallmatrix}\Xi\\ \mathit{R}\\ 1\end{smallmatrix}},\ \begin{array}{c}x:\ A\\ {[A]_{\mathit{R}}^{\Xi}}\\ x_1:\ [A]_1^{\Xi}\end{array}
\end{array}$$

Theorem (Partial relational interpretation of F)

(1) $\vdash^{F} E,\ E' \ \Rightarrow\ \vdash^{\mathit{R}\,0} E,\ [E']_{\begin{smallmatrix}\mathrm{dom(E)}\\ \mathit{R}\\ 1\end{smallmatrix}}$

(2) $E,\ E'\vdash^{F} A \ \Rightarrow\ E,\ [E']_{\begin{smallmatrix}\mathrm{dom(E)}\\ \mathit{R}\\ 1\end{smallmatrix}} \vdash^{\mathit{R}\,0} \begin{array}{c}A\\ {[A]_{\mathit{R}}^{\mathrm{dom(E)}}}\\ {[A]_1^{\mathrm{dom(E)}}}\end{array}$

(3) $E,\ E'\vdash^{F} a:A \ \Rightarrow\ E,\ [E']_{\begin{smallmatrix}\mathrm{dom(E)}\\ \mathit{R}\\ 1\end{smallmatrix}} \vdash^{\mathit{R}\,0} \begin{array}{c}a:\ A\\ {[A]_{\mathit{R}}^{\mathrm{dom(E)}}}\\ {[a]_1^{\mathrm{dom(E)}}:\ [A]_1^{\mathrm{dom(E)}}}\end{array}$

Note that the occurrences of E on the right of the implications are abbreviations for *R* environments, as defined in section 2.1.1.

Proof
The proof of this theorem (and of its continuation in section 2.1.4) is by induction on F derivations, for any division of environments into two parts, E and E'. In the proof of (3), the variable cases are settled using (Rel Val x*R* y) for variables in dom(E'), and using (Rel Val *R* x) for variables in dom(E).
□

We emphasize the two special cases where E and E' are empty, respectively:

Corollary (Relational interpretation of F)

(1) $\vdash^{F} E' \;\Rightarrow\; \vdash^{\mathcal{R}^0} [E']^{\varnothing}_{\substack{\mathcal{R}\\1}}$

(2) $E' \vdash^{F} A \;\Rightarrow\; [E']^{\varnothing}_{\substack{\mathcal{R}\\1}} \vdash^{\mathcal{R}^0} \begin{array}{c} A \\ [A]^{\varnothing}_{\mathcal{R}} \\ [A]^{\varnothing}_{1} \end{array}$

(3) $E' \vdash^{F} a : A \;\Rightarrow\; [E']^{\varnothing}_{\substack{\mathcal{R}\\1}} \vdash^{\mathcal{R}^0} \begin{array}{c} a : A \\ [A]^{\varnothing}_{\mathcal{R}} \\ [a]^{\varnothing}_{1} : [A]^{\varnothing}_{1} \end{array}$

Part (3) of this corollary is a syntactic version of Reynolds's abstraction theorem. It can be proved directly, and its proof does not require the use of parametricity (that is, the use of the rules (Rel Val $\mathcal{R}$ x) and (Rel Val $\mathcal{R}$ y)).

Corollary (Soundness of F in $\mathcal{R}$)

(1) $\vdash^{F} E \;\Rightarrow\; \vdash^{\mathcal{R}^0} E$

(2) $E \vdash^{F} A \;\Rightarrow\; E \vdash^{\mathcal{R}^0} A$

(3) $E \vdash^{F} a : A \;\Rightarrow\; E \vdash^{\mathcal{R}^0} a : A$

Part (3) of this corollary is a syntactic version of Reynolds's identity extension property. We refer to it by that name in the sequel.

We close this section with another lemma about flattening. Its proof is very similar to that of the theorem (Partial relational interpretation of F).

Lemma ($\mathcal{R}$ derivations from flattened F derivations)

(1) $\vdash E \;\wedge\; (E)_F \vdash^{F} A \;\Rightarrow\; E \vdash A$

(2) $\vdash E \;\wedge\; (E)_F \vdash^{F} a : A \;\Rightarrow\; E \vdash a : A$

Proof

We prove the statements (1) and (2) as instances of the following more general statements, which are proved as the corresponding statements (1)-(3) of the theorem (Partial relational interpretation of F):

(1') $\vdash E \;\wedge\; \vdash^{F} (E)_F, E' \;\Rightarrow\; \vdash^{\mathcal{R}^0} E, [E']^{\mathrm{dom}(E)}_{\substack{\mathcal{R}\\1}}$

(2') $\vdash E \;\wedge\; (E)_F, E' \vdash^{F} A \;\Rightarrow\; E, [E']^{\mathrm{dom}(E)}_{\substack{\mathcal{R}\\1}} \vdash^{\mathcal{R}^0} \begin{array}{c} A \\ [A]^{\mathrm{dom}(E)}_{\mathcal{R}} \\ [A]^{\mathrm{dom}(E)}_{1} \end{array}$

(3') $\vdash E \;\wedge\; (E)_F, E' \vdash^{F} a : A \;\Rightarrow\; E, [E']^{\mathrm{dom}(E)}_{\substack{\mathcal{R}\\1}} \vdash^{\mathcal{R}^0} \begin{array}{c} a : A \\ [A]^{\mathrm{dom}(E)}_{\mathcal{R}} \\ [a]^{\mathrm{dom}(E)}_{1} : [A]^{\mathrm{dom}(E)}_{1} \end{array}$

□

Before extending some of these results to equality judgments (in section 2.1.4), we complete our collection of structural properties of $\mathcal{R}^0$.

2.1.3 Structural lemmas (continued)

Lemma (Rel Id)

$$\mathrm{E} \vdash \begin{matrix} \mathrm{A} \\ R \\ \mathrm{B} \end{matrix} \quad\Rightarrow\quad \mathrm{E} \vdash \mathrm{A} \;\;\wedge\;\; \mathrm{E} \vdash \mathrm{B}$$

Proof

From $\mathrm{E} \vdash^{R^0} \begin{matrix} \mathrm{A} \\ R \\ \mathrm{B} \end{matrix}$ we derive $(\mathrm{E})_{\mathrm{F}} \vdash^{\mathrm{F}} \mathrm{A}$ and $(\mathrm{E})_{\mathrm{F}} \vdash^{\mathrm{F}} \mathrm{B}$ by theorem (Flattened F derivations from R derivations), and conclude by lemma (R derivations from flattened F derivations).

□

Lemma (Type substitution)

Assume $\mathrm{E} \vdash \begin{matrix} \mathrm{C} \\ U \\ \mathrm{D} \end{matrix}$. Then:

(1) $$\vdash \mathrm{E}, \begin{matrix} \mathrm{X} \\ W \\ \mathrm{Y} \end{matrix}, \mathrm{E'} \quad\Rightarrow\quad \vdash \mathrm{E}, \mathrm{E'}\left\{\begin{matrix} \mathrm{X} \leftarrow \mathrm{C} \\ W \leftarrow U \\ \mathrm{Y} \leftarrow \mathrm{D} \end{matrix}\right\}$$

(2) $$\mathrm{E}, \begin{matrix} \mathrm{X} \\ W \\ \mathrm{Y} \end{matrix}, \mathrm{E'} \vdash J \quad\Rightarrow\quad \mathrm{E}, \mathrm{E'}\left\{\begin{matrix} \mathrm{X} \leftarrow \mathrm{C} \\ W \leftarrow U \\ \mathrm{Y} \leftarrow \mathrm{D} \end{matrix}\right\} \vdash J\left\{\begin{matrix} \mathrm{X} \leftarrow \mathrm{C} \\ W \leftarrow U \\ \mathrm{Y} \leftarrow \mathrm{D} \end{matrix}\right\}$$

Proof

We mention only that the case (Rel W) requires lemma (Weakening), and that the case (Rel WX) requires lemma (Rel Id) in addition.

□

With similar proofs, we obtain:

Lemma (Rel Val Refl)

$$\mathrm{E} \vdash \begin{matrix} \mathrm{a} : \mathrm{A} \\ R \\ \mathrm{b} : \mathrm{B} \end{matrix} \quad\Rightarrow\quad \mathrm{E} \vdash \mathrm{a} : \mathrm{A} \;\;\wedge\;\; \mathrm{E} \vdash \mathrm{b} : \mathrm{B}$$

Lemma (Value substitution)

(1) $$\vdash \mathrm{E}, \begin{matrix} \mathrm{z} : \mathrm{D} \\ U \\ \mathrm{z'} : \mathrm{D'} \end{matrix}, \mathrm{E'} \quad\Rightarrow\quad \vdash \mathrm{E}, \mathrm{E'}$$

(2) $$\mathrm{E}, \begin{matrix} \mathrm{z} : \mathrm{D} \\ U \\ \mathrm{z'} : \mathrm{D'} \end{matrix}, \mathrm{E'} \vdash \begin{matrix} \mathrm{A} \\ R \\ \mathrm{B} \end{matrix} \quad\Rightarrow\quad \mathrm{E}, \mathrm{E'} \vdash \begin{matrix} \mathrm{A} \\ R \\ \mathrm{B} \end{matrix}$$

(3) $$\mathrm{E} \vdash \begin{matrix} \mathrm{d} : \mathrm{D} \\ U \\ \mathrm{d'} : \mathrm{D'} \end{matrix} \;\;\wedge\;\; \mathrm{E}, \begin{matrix} \mathrm{z} : \mathrm{D} \\ U \\ \mathrm{z'} : \mathrm{D'} \end{matrix}, \mathrm{E'} \vdash \begin{matrix} \mathrm{a} : \mathrm{A} \\ R \\ \mathrm{b} : \mathrm{B} \end{matrix} \quad\Rightarrow\quad \mathrm{E}, \mathrm{E'} \vdash \begin{matrix} \mathrm{a} : \mathrm{A} \\ R \\ \mathrm{b} : \mathrm{B} \end{matrix}\left\{\begin{matrix} \mathrm{z} \leftarrow \mathrm{d} \\ \mathrm{z'} \leftarrow \mathrm{d'} \end{matrix}\right\}$$

Lemma (Implied judgments)

(3) $$E \vdash \begin{array}{c} a : A \\ R \\ b : B \end{array} \Rightarrow E \vdash \begin{array}{c} A \\ R \\ B \end{array}$$

We conclude with some derived rules that generalize the R rules for β and η equivalence.

Lemma (Generalized beta/eta)

(Gen Beta)

$$\frac{E, \begin{array}{c} x : A \\ R \\ x' : A' \end{array} \vdash \begin{array}{c} b : B \\ S \\ b' : B' \end{array} \qquad E \vdash \begin{array}{c} a : A \\ R \\ a' : A' \end{array} \qquad \begin{array}{c} x \notin b', S \\ x' \notin b, S \end{array}}{E \vdash \begin{array}{c} (\lambda(x:A)b)(a) : B \\ S \\ b'\{x' \leftarrow a'\} : B' \end{array} \qquad E \vdash \begin{array}{c} b\{x \leftarrow a\} : B \\ S \\ (\lambda(x':A')b')(a') : B' \end{array}}$$

(Gen Beta2)

$$\frac{E, \begin{array}{c} X \\ W \\ X' \end{array} \vdash \begin{array}{c} b : B \\ S \\ b' : B' \end{array} \qquad E \vdash \begin{array}{c} C \\ T \\ C' \end{array} \qquad \begin{array}{c} X \notin b', B', S \\ X' \notin b, B, S \end{array}}{E \vdash \begin{array}{c} (\lambda(X)b)(C) : B\{X \leftarrow C\} \\ S\{W \leftarrow T\} \\ b'\{X' \leftarrow C'\} : B'\{X' \leftarrow C'\} \end{array} \qquad E \vdash \begin{array}{c} b\{X \leftarrow C\} : B\{X \leftarrow C\} \\ S\{W \leftarrow T\} \\ (\lambda(X')b')(C') : B'\{X' \leftarrow C'\} \end{array}}$$

(Gen Eta)

$$\frac{E \vdash \begin{array}{c} b : A \to B \\ R \to S \\ b' : A' \to B' \end{array} \qquad x, x' \notin dom(E)}{E \vdash \begin{array}{c} \lambda(x:A)b(x) : A \to B \\ R \to S \\ b' : A' \to B' \end{array} \qquad E \vdash \begin{array}{c} b : A \to B \\ R \to S \\ \lambda(x':A')b'(x') : A' \to B' \end{array}}$$

(Gen Eta2)

$$\frac{E \vdash \begin{array}{c} b : \forall(X)B \\ \forall(W)S \\ b' : \forall(X')B' \end{array} \qquad X, X' \notin dom(E)}{E \vdash \begin{array}{c} \lambda(X)b(X) : \forall(X)B \\ \forall(W)S \\ b' : \forall(X')B' \end{array} \qquad E \vdash \begin{array}{c} b : \forall(X)B \\ \forall(W)S \\ \lambda(X')b'(X') : \forall(X')B' \end{array}}$$

2.1.4 From F to R^0 and from R^0 to F (continued)

In this section, we complete the material of section 2.1.2 by showing that the same equalities can be derived in F and in R^0. We begin by extending the theorem (Partial relational interpretation of F) to equalities.

Lemma (Environment redecoration)

Let $\vdash^F E, E'$. If $E, E' \vdash^{R^0} J$ and no variables from the middle or bottom of E,E' occur free in J, then $E, [E']_{\begin{array}{c}\scriptstyle dom(E) \\ \scriptstyle R \\ \scriptstyle 1\end{array}} \vdash^{R^0} J$.

Proof

Let $\tilde{}$ denote the renaming of all the variables X and x in dom(E') with fresh $\tilde{X}$ and $\tilde{x}$.

Then $\mathrm{E},\ \tilde{\mathrm{E}}' \vdash^{R\,0} \tilde{J}$, and by weakening $\mathrm{E},\ [\mathrm{E}']^{\mathrm{dom(E)}}_{\substack{R\\1}},\ \tilde{\mathrm{E}}' \vdash^{R\,0} \tilde{J}$.

Moreover, for every X_i in E' we have $\mathrm{E},[\mathrm{E}']^{\mathrm{dom(E)}}_{\substack{R\\1}} \vdash^{R\,0} \begin{array}{c}\mathrm{X}_\mathrm{i}\\ \mathrm{X}_\mathrm{i}\\ \mathrm{X}_\mathrm{i}\end{array}$ by (Rel W X),

and for every $\mathrm{x}_\mathrm{j}{:}\mathrm{A}_\mathrm{j}$ in E' we have $\mathrm{E},[\mathrm{E}']^{\mathrm{dom(E)}}_{\substack{R\\1}} \vdash^{R\,0} \begin{array}{c}\mathrm{x}_\mathrm{j} : \mathrm{A}_\mathrm{j}\\ \mathrm{A}_\mathrm{j}{*}\\ \mathrm{x}_\mathrm{j} : \mathrm{A}_\mathrm{j}\end{array}$ by (Rel Val R x).

By repeated applications of (Type substitution) and (Value substitution) to $\mathrm{E},\ [\mathrm{E}']^{\mathrm{dom(E)}}_{\substack{R\\1}},\ \tilde{\mathrm{E}}' \vdash^{R\,0} \tilde{J}$, eliminating the variables of $\tilde{\mathrm{E}}'$ from left to right, we obtain:

$$\mathrm{E},\ [\mathrm{E}']^{\mathrm{dom(E)}}_{\substack{R\\1}} \vdash^{R\,0} \tilde{J}\left\{\begin{array}{l}\tilde{\mathrm{X}}_\mathrm{i} \leftarrow \mathrm{X}_\mathrm{i}\\ \tilde{X}_\mathrm{i} \leftarrow \mathrm{X}_\mathrm{i}\\ \tilde{\mathrm{X}}_\mathrm{i}' \leftarrow \mathrm{X}_\mathrm{i}\end{array}\right\}\left\{\begin{array}{l}\tilde{\mathrm{x}}_\mathrm{j} \leftarrow \mathrm{x}_\mathrm{j}\\ \tilde{\mathrm{x}}_\mathrm{j}' \leftarrow \mathrm{x}_\mathrm{j}\end{array}\right\}$$

The bottom three sets of substitutions are vacuous by assumption. The top two sets of substitutions transform $\tilde{J}$ back into J.
□

Theorem (Partial relational interpretation of F)

(4) $\qquad \mathrm{E},\ \mathrm{E}' \vdash^{\mathrm{F}} \mathrm{a} = \mathrm{b} : \mathrm{A} \ \Rightarrow$

$$\mathrm{E},[\mathrm{E}']^{\mathrm{dom(E)}}_{\substack{R\\1}} \vdash^{R\,0} \begin{array}{c}\mathrm{a} : \mathrm{A}\\ [\mathrm{A}]^{\mathrm{dom(E)}}_{R}\\ [\mathrm{b}]^{\mathrm{dom(E)}}_{1} : [\mathrm{A}]^{\mathrm{dom(E)}}_{1}\end{array} \quad \text{and} \quad \mathrm{E},[\mathrm{E}']^{\mathrm{dom(E)}}_{\substack{R\\1}} \vdash^{R\,0} \begin{array}{c}\mathrm{b} : \mathrm{A}\\ [\mathrm{A}]^{\mathrm{dom(E)}}_{R}\\ [\mathrm{a}]^{\mathrm{dom(E)}}_{1} : [\mathrm{A}]^{\mathrm{dom(E)}}_{1}\end{array}$$

Proof

The cases (Val Beta), (Val Beta2), (Val Eta), and (Val Eta2) are solved with the (Generalized beta/eta) lemma. We detail the case (Val Eq Trans). If $\mathrm{E},\ \mathrm{E}' \vdash^{\mathrm{F}} \mathrm{a} = \mathrm{b} : \mathrm{A}$ and $\mathrm{E},\ \mathrm{E}' \vdash^{\mathrm{F}} \mathrm{b} = \mathrm{c} : \mathrm{A}$, then, applying (4) to

$\mathrm{E},\ \mathrm{E}' \vdash^{\mathrm{F}} \mathrm{a} = \mathrm{b} : \mathrm{A}$ with the splitting (E,E'),(∅) of E,E', and to

$\mathrm{E},\ \mathrm{E}' \vdash^{\mathrm{F}} \mathrm{b} = \mathrm{c} : \mathrm{A}$ with the splitting (E),(E'),

we get:

$$\mathrm{E},\ \mathrm{E}',\ [\varnothing]^{\mathrm{dom(E,E')}}_{\substack{R\\1}} \vdash^{R\,0} \begin{array}{c}\mathrm{a} : \mathrm{A}\\ [\mathrm{A}]^{\mathrm{dom(E,E')}}_{R}\\ [\mathrm{b}]^{\mathrm{dom(E,E')}}_{1} : [\mathrm{A}]^{\mathrm{dom(E,E')}}_{1}\end{array}$$

and

$$\mathrm{E},\ [\mathrm{E}']^{\mathrm{dom(E)}}_{\substack{R\\1}} \vdash^{R\,0} \begin{array}{c}\mathrm{b} : \mathrm{A}\\ [\mathrm{A}]^{\mathrm{dom(E)}}_{R}\\ [\mathrm{c}]^{\mathrm{dom(E)}}_{1} : [\mathrm{A}]^{\mathrm{dom(E)}}_{1}\end{array}$$

The former can be written more simply $\mathrm{E},\ \mathrm{E}' \vdash^{R\,0} \begin{array}{c}\mathrm{a} : \mathrm{A}\\ \mathrm{A}{*}\\ \mathrm{b} : \mathrm{A}\end{array}$

so, by lemma (Environment redecoration) $\mathrm{E},\ [\mathrm{E}']^{\mathrm{dom(E)}}_{\substack{R\\1}} \vdash^{R\,0} \begin{array}{c}\mathrm{a} : \mathrm{A}\\ \mathrm{A}{*}\\ \mathrm{b} : \mathrm{A}\end{array}$.

The conclusion

$$E,\ [E']^{\mathrm{dom}(E)}_{\substack{\mathcal{R}\\1}} \vdash^{\mathcal{R}^0} \begin{array}{c} a : A \\ [A]^{\mathrm{dom}(E)}_{\mathcal{R}} \\ [c]^{\mathrm{dom}(E)}_1 : [A]^{\mathrm{dom}(E)}_1 \end{array}$$

follows by (Rel Val Saturation Lft). The other judgment relating c and $[a]^{\mathrm{dom}(E)}_1$ is proved similarly.
□

Again, we obtain two interesting special cases:

Corollary (Relational interpretation of F)

(4) $E' \vdash^{F} a = b : A \ \Rightarrow$

$$[E']^{\varnothing}_{\substack{\mathcal{R}\\1}} \vdash^{\mathcal{R}^0} \begin{array}{c} a : A \\ [A]^{\varnothing}_{\mathcal{R}} \\ [b]^{\varnothing}_1 : [A]^{\varnothing}_1 \end{array} \quad \text{and} \quad [E']^{\varnothing}_{\substack{\mathcal{R}\\1}} \vdash^{\mathcal{R}^0} \begin{array}{c} b : A \\ [A]^{\varnothing}_{\mathcal{R}} \\ [a]^{\varnothing}_1 : [A]^{\varnothing}_1 \end{array}$$

Corollary (Soundness of F in *R*)

(4) $E \vdash^{F} a = b : A \ \Rightarrow \ E \vdash^{\mathcal{R}^0} \begin{array}{c} a : A \\ A^* \\ b : A \end{array}$

The final theorem about $\mathcal{R}^0$ is a conservativity result; it states that if two terms are related in $\mathcal{R}^0$ by a type, then they are provably equal in F modulo renamings of free variables. Some definitions are needed to express the necessary renamings:

Definition ($E^{\top}$, $E_{\perp}$)
$E^{\top}$ is the F environment built from the top part of the $\mathcal{R}$ environment E. Note that $\vdash^{\mathcal{R}} E \not\Rightarrow \vdash^{F} E^{\top}$.

$$\varnothing^{\top} = \varnothing \qquad \left(\begin{array}{c} X \\ E, \mathcal{W} \\ Y \end{array}\right)^{\top} = E^{\top}, X \qquad \left(\begin{array}{c} x : A \\ E,\ \ \mathcal{R} \\ y : B \end{array}\right)^{\top} = E^{\top}, x : A$$

$E_{\perp}$ is defined symmetrically from the bottom part of E.

Definition ($E^{\Downarrow}$, $E_{\Uparrow}$)

$\{E^{\Downarrow}\}$ is the substitution that replaces Y by X for each $\begin{array}{c} X \\ \mathcal{W} \\ Y \end{array}$ in E, and

replaces y by x for each $\begin{array}{c} x : A \\ \mathcal{R} \\ y : B \end{array}$ in E.

$\{E_{\Uparrow}\}$ is defined symmetrically.

Theorem (Conservativity over F for equality in $\mathcal{R}^0$)

$$\vdash^{\mathcal{R}^0} E \ \Rightarrow \ \vdash^{F} E^{\top}\{E^{\Downarrow}\} \ \wedge \ \vdash^{F} E_{\perp}\{E_{\Uparrow}\}$$

$$E \vdash^{\mathcal{R}^0} \begin{array}{c} A \\ \mathcal{R} \\ B \end{array} \ \Rightarrow \ E^{\top}\{E^{\Downarrow}\} \vdash^{F} A\{E^{\Downarrow}\} \ \wedge \ E_{\perp}\{E_{\Uparrow}\} \vdash^{F} B\{E_{\Uparrow}\}$$

$$E \vdash^{R^0} \begin{array}{c} a : A \\ A^* \\ b : A \end{array} \Rightarrow E^{\top}\{E^{\Downarrow}\} \vdash^{F} (a = b : A)\{E^{\Downarrow}\} \;\wedge\; E_{\perp}\{E_{\Uparrow}\} \vdash^{F} (a = b : A)\{E_{\Uparrow}\}$$

For example, here is an instance of the third implication of the theorem:

$$\begin{array}{c} X \\ W \\ Y \end{array}, \begin{array}{c} x : X \\ X \\ y : X \end{array} \vdash^{R^0} \begin{array}{c} x : X \\ X \\ y : X \end{array} \Rightarrow \begin{array}{l} (X, x : X)\{Y \leftarrow X,\ y \leftarrow x\} \vdash^{F} (x = y : X)\{Y \leftarrow X,\ y \leftarrow x\} \\ \\ (Y, y : X)\{X \leftarrow Y,\ x \leftarrow y\} \vdash^{F} (x = y : X)\{X \leftarrow Y,\ x \leftarrow y\} \end{array}$$

that is:

$$\begin{array}{c} X \\ W \\ Y \end{array}, \begin{array}{c} x : X \\ X \\ y : X \end{array} \vdash^{R^0} \begin{array}{c} x : X \\ X \\ y : X \end{array} \Rightarrow \begin{array}{l} X, x : X \vdash^{F} x = x : X \\ \\ Y, y : Y \vdash^{F} y = y : Y \end{array}$$

2.2 Functional relations (system R^1)

In this section we use $\vdash^{R^1}$ (or simply $\vdash$) for derivations in R^1. The complete rules for system R^1 are listed in the appendix. Since the rules of R^0 are included in R^1, we have:

Lemma (Transfer)
For every R^0 derivation there exists an R^1 derivation which has the same size and shape and the same conclusion.

The following R^0 results from section 2.1 extend to R^1 by uniformly replacing R^0 derivations by R^1 derivations in the statements: (Renaming), (Weakening), (Flattened F derivations from R derivations), (R derivations from flattened F derivations), (Rel Id), (Type substitution), (Rel Val Refl), (Implied judgments), (Generalized beta/eta), (Partial relational interpretation of F), (Relational interpretation of F), and (Soundness of F in R). The R^1 proofs use either straightforward extensions of the R^0 inductions, or the (Transfer) lemma.

The value substitution lemma reads as follows in R^1:

Lemma (Value substitution)
Assume $E \vdash \begin{array}{c} d : D \\ U \\ d' : D' \end{array}$. Then:

(1) $\vdash E, \begin{array}{c} z : D \\ U \\ z' : D' \end{array}, E' \;\Rightarrow\; \vdash E, E'\left\{\begin{array}{l} z \leftarrow d \\ z' \leftarrow d' \end{array}\right\}$

(2) $E, \begin{array}{c} z : D \\ U \\ z' : D' \end{array}, E' \vdash \begin{array}{c} A \\ R \\ B \end{array} \;\Rightarrow\; E, E'\left\{\begin{array}{l} z \leftarrow d \\ z' \leftarrow d' \end{array}\right\} \vdash \begin{array}{c} A \\ R \\ B \end{array}\left\{\begin{array}{l} z \leftarrow d \\ z' \leftarrow d' \end{array}\right\}$

(3) $E, \begin{array}{c} z : D \\ U \\ z' : D' \end{array}, E' \vdash \begin{array}{c} a : A \\ R \\ b : B \end{array} \;\Rightarrow\; E, E'\left\{\begin{array}{l} z \leftarrow d \\ z' \leftarrow d' \end{array}\right\} \vdash \begin{array}{c} a : A \\ R \\ b : B \end{array}\left\{\begin{array}{l} z \leftarrow d \\ z' \leftarrow d' \end{array}\right\}$

One of the conservativity results for R^0, (Conservativity over F for equality in R^0), does not extend to R^1 . Many examples of new equalities are shown in section 3.

We close this section with a negative result:

Counterexample (to strengthening)

One might expect a strengthening lemma to hold, as it does in F. Such a lemma would claim that if E,x:A $\vdash J$ is provable and x does not occur in J, then E $\vdash J$ is provable as well. As in calculi with empty types [Meyer, *et al.* 1990] this lemma fails in R^1.

As an example we show that $x : \forall(X)X \vdash \begin{array}{c} \mathrm{true} : \mathrm{Bool} \\ \mathrm{Bool}^* \\ \mathrm{false} : \mathrm{Bool} \end{array}$ but the consistency of R^1 disallows $\vdash \begin{array}{c} \mathrm{true} : \mathrm{Bool} \\ \mathrm{Bool}^* \\ \mathrm{false} : \mathrm{Bool} \end{array}$ (see section 4). This result can be attributed to the fact that $\forall$(X)X is provably initial, as stated in section 3.

We start by introducing a functional relation, proving that $x : \forall(X)X \vdash \langle\lambda(y : \begin{array}{c} \mathrm{Bool} \\ \mathrm{Bool} \\ \mathrm{Bool} \end{array})\mathrm{true}\rangle$. Furthermore we have $x : \forall(X)X \vdash \langle\lambda(y : \mathrm{Bool})\mathrm{true}\rangle$ $\begin{array}{c} x(\mathrm{Bool}) : \mathrm{Bool} \\ \\ x(\mathrm{Bool}) : \mathrm{Bool} \end{array}$ by (Rel Val R x) and (Rel Val Appl2), and eliminating the functional relation we obtain $x : \forall(X)X \vdash \begin{array}{c} \mathrm{true} : \mathrm{Bool} \\ \mathrm{Bool}^* \\ x(\mathrm{Bool}) : \mathrm{Bool} \end{array}$. Similarly, we derive $x : \forall(X)X \vdash \begin{array}{c} \mathrm{false} : \mathrm{Bool} \\ \mathrm{Bool}^* \\ x(\mathrm{Bool}) : \mathrm{Bool} \end{array}$. Finally, by (Rel Val Symm) and (Rel Val Saturation Lft), we obtain $x : \forall(X)X \vdash \begin{array}{c} \mathrm{true} : \mathrm{Bool} \\ \mathrm{Bool}^* \\ \mathrm{false} : \mathrm{Bool} \end{array}$.

3. Theorems for free, syntactically

In this section we illustrate the power of R^1 by carrying out formal proofs. The results given below apply to all terms, and not just to closed terms. In some cases, even the results for closed terms are somewhat difficult; Wadler's work [Wadler 1989] includes a few interesting semantic results that can be read as results about closed terms. In order to deal with open terms we do not use structural induction (like Mairson), but rather the rule (Rel Val R x) and the identity extension property (that is, part (3) of corollary (Soundness of F in R), see section 2.1.2). Throughout the section, the η rules are used heavily.

We begin with two simple examples in the first two subsections. Then we develop some general technical tools in sections 3.3 through 3.5; the reader may prefer to skim these sections in the first pass. We formalize commuting squares of functions and the notion of extensional equality of relations. Furthermore, we show that covariant functors commute with functional relations. In section 3.6 we apply these tools to prove properties of the type of the map function. We also obtain a more substantial theorem about initial algebras in section 3.7: the F encoding of initial algebras for covariant functors is indeed initial. (Without parametricity assumptions, the encoding is weakly initial.) Similarly, we treat the encoding of products and coproducts in section 3.8. In section 3.9, we briefly discuss some applications of initiality (mainly to properties of the type Nat). Finally, in section 3.10, we raise a conjecture that connects Reynolds's notion of parametricity with type erasures.

In many statements we make the superscripts explicit in $\vdash^{F}$ and $\vdash^{R^1}$, especially when a statement involves judgments of both systems. Superscripts are often omitted in proofs. A plain $\vdash$ stands for $\vdash^{R^1}$. We use the abbreviations introduced in section 2 and summarized in the appendix.

3.1 A simple example

As a first example we generalize and formalize the reasoning of section 1 about the type $\forall(X)X\rightarrow$Bool.

Proposition (Constant)
The type $\forall(X)X\to A$ (where X is not free in A) is isomorphic to A. That is, given E such that $E \vdash^{F} A$, there exist two terms i and j such that $E \vdash^{F} i: (\forall(X)X\to A)\to A$, $E \vdash^{F} j: A\to(\forall(X)X\to A)$, and:

$$E,\ u:A \vdash^{R^1} \begin{array}{c} i(j(u)):A \\ A^* \\ u:A \end{array}$$

$$E,\ t:\forall(X)X\to A \vdash^{R^1} \begin{array}{c} j(i(t)):\ \forall(X)X\to A \\ (\forall(X)X\to A)^* \\ t:\ \forall(X)X\to A \end{array}$$

where u and t are fresh.

Proof
First we observe that, by the soundness of F in R, $E \vdash^{F} A$ implies $E \vdash A$, hence $\vdash E,u{:}A$. Define:

$$j=\lambda(u{:}A)\lambda(X)\lambda(x{:}X)u$$

For each u, j(u) is a polymorphic constant function. Pick a closed type B and a closed term b such that $\vdash^{F} b{:}B$ (for example, $B = \forall(X)X\to X$, $b = \lambda(X)\lambda(x{:}X)x$). By the soundness of F in R and weakening, we have $E, X, z{:}X \vdash b{:}B$. Now define:

$$i=\lambda(t{:}\ \forall(X)X\to A)t(B)(b)$$

Two applications of the β rule yield: $E,\ u:A \vdash \begin{array}{c} i(j(u)):A \\ A^* \\ u:A \end{array}$.

The second result requires parametricity. We consider the constant function $\lambda(z:X)b$ as a relation. We have $E,X \vdash \begin{array}{c} X \\ \langle\lambda(z:X)b\rangle \\ B \end{array}$ by (Rel FRel), hence $E,\ t:\forall(X)X\to A,\ X \vdash \begin{array}{c} t(X):\ X\to A \\ \langle\lambda(z:X)b\rangle\to A^* \\ t(B):B\to A \end{array}$ by (Rel Val Rx) and (Rel Val Appl2). By functional-relation introduction (more precisely, by (Rel Val FRel Intro), (Rel Val Beta), and (Rel Val Saturation Rht)) we have $E,\ X,\ x:X \vdash \begin{array}{c} x:X \\ \langle\lambda(z:X)b\rangle \\ b:B \end{array}$. By (Rel Val Appl) it follows that $E,\ t:\forall(X)X\to A,\ \begin{array}{c} X \\ \mathcal{X} \\ X_1 \end{array},\ \begin{array}{c} x:X \\ \mathcal{X} \\ x_1:X_1 \end{array} \vdash \begin{array}{c} t(X)(x):\ A \\ A^* \\ t(B)(b):A \end{array}$, where we have partially expanded the environment abbreviations. By the β rules, we can replace t(B)(b) with $j(i(t))(X_1)(x_1)$. We obtain:

$$E,\ t:\forall(X)X\to A,\ \begin{array}{c} X \\ \mathcal{X} \\ X_1 \end{array},\ \begin{array}{c} x:X \\ \mathcal{X} \\ x_1:X_1 \end{array} \vdash \begin{array}{c} t(X)(x):\ A \\ A^* \\ j(i(t))(X_1)(x_1):A \end{array}$$

and the second conclusion follows by (Rel Val Fun), (Rel Val Eta), (Rel Val Fun2), and (Rel Val Eta2).

□

3.2 $\forall(X)X\to X$ contains only the identity function

We show that all terms of type $\forall(X)X\to X$ are equal to the polymorphic identity function $id = \lambda(X)\lambda(x:X)x$, and hence that this type is terminal. For closed terms this result follows easily from strong normalization, but a strong-normalization argument does not extend to open terms.

Proposition (Terminal)

$$E \vdash^{F} f : \forall(X)X \to X \;\Rightarrow\; E \vdash^{R^1} \begin{array}{c} f : \forall(X)X \to X \\ (\forall(X)X \to X)^{*} \\ id : \forall(X)X \to X \end{array}$$

Proof

By the theorem (Soundness of F in R) and by the lemma (Value substitution), it suffices to prove:

$$z : \forall(X)X \to X \vdash \begin{array}{c} z : \forall(X)X \to X \\ (\forall(X)X \to X)^{*} \\ id : \forall(X)X \to X \end{array}$$

Using (Rel FRel) we obtain $z : \forall(X)X \to X,\; X,\; x : X \vdash \begin{array}{c} \forall(X)X \to X \\ \langle\lambda(g : \forall(X)X \to X)x\rangle \\ X \end{array}$.

Hence we derive:

$$z : \forall(X)X \to X,\; X,\; x : X \vdash \begin{array}{c} z(\forall(X)X \to X) : (\forall(X)X \to X) \to (\forall(X)X \to X) \\ \langle\lambda(g : \forall(X)X \to X)x\rangle \to \langle\lambda(g : \forall(X)X \to X)x\rangle \\ z(X) : X \to X \end{array}$$

by (Rel Val R x) and (Rel Val Appl2),

$$z : \forall(X)X \to X,\; X,\; x : X \vdash \begin{array}{c} z : \forall(X)X \to X \\ \langle\lambda(g : \forall(X)X \to X)x\rangle \\ x : X \end{array},$$

by (Rel Val FRel Intro), (Rel Val Beta), and (Rel Val Saturation Rht),

$$z : \forall(X)X \to X,\; X,\; x : X \vdash \begin{array}{c} z(\forall(X)X \to X)(z) : (\forall(X)X \to X) \\ \langle\lambda(g : \forall(X)X \to X)x\rangle \\ z(X)(x) : X \end{array}.$$

by (Rel Val Appl), and

$$z : \forall(X)X \to X,\; X,\; x : X \vdash \begin{array}{c} x : X \\ X \\ z(X)(x) : X \end{array}$$

by (Rel Val FRel Elim). Furthermore, we have:

$$z : \forall(X)X \to X,\; \begin{array}{c} X \\ \mathit{X} \\ X_1 \end{array},\; \begin{array}{c} x : X \\ \mathit{X} \\ x_1 : X_1 \end{array} \vdash \begin{array}{c} z(X)(x) : X \\ \mathit{X} \\ z(X_1)(x_1) : X_1 \end{array}$$

by (Rel Val R x), (Rel W), (Rel Val Appl2), (Rel Val xR y), and (Rel Val Appl); and by (Rel Val Saturation Lft) we derive:

$$z : \forall(X)X \to X,\; \begin{array}{c} X \\ \mathit{X} \\ X_1 \end{array},\; \begin{array}{c} x : X \\ \mathit{X} \\ x_1 : X_1 \end{array} \vdash \begin{array}{c} x : X \\ \mathit{X} \\ z(X_1)(x_1) : X_1 \end{array}$$

After using the β rules to equate x and id(X)(x), the conclusion follows as in proposition (Constant), with in addition an application of (Rel Val Symm).

□

3.3 Commuting squares as assumptions

The following three subsections develop tools that serve to formulate and prove theorems about the type of map (section 3.6) and about initial algebras (section 3.7). We first formalize in R^1 the assumption that a square like the following commutes:

$$\begin{tikzcd} B \arrow[r, "t"] \arrow[d, "k"'] & A \arrow[d, "h"] \\ B' \arrow[r, "t'"] & A' \end{tikzcd}$$

Functional relations can be used to encode such an equational assumption. The commutation of the diagram above can be expressed by the requirement that $\langle k\rangle \to \langle h\rangle$ relates t and t'. This is formalized in the following lemma, where we use ";" to denote the (encoding of) composition, setting t;h = λ(x)h(t(x)).

Lemma (Commuting squares)
Suppose that $E \vdash^F t : B \to A$, $E \vdash^F t' : B' \to A'$, $E \vdash^F k : B \to B'$, and $E \vdash^F h : A \to A'$.
Then $E \vdash^{R^1} \begin{array}{c} t;h : B \to A' \\ (B \to A')^* \\ k;t' : B \to A' \end{array}$ if and only if $E \vdash^{R^1} \begin{array}{c} t : B \to A \\ \langle k\rangle \to \langle h\rangle \\ t' : B' \to A' \end{array}$.

Proof
Let $E \vdash \begin{array}{c} t;h : B \to A' \\ (B \to A')^* \\ k;t' : B \to A' \end{array}$.
We claim:

$$E, \begin{array}{c} x : B \\ \langle k\rangle \\ x' : B' \end{array} \vdash \begin{array}{c} t(x) : A \\ \langle h\rangle \\ t'(x') : A' \end{array}$$

where x and x' are fresh. By (Rel Val Saturation Rht) we may decompose the claim into three parts, all of them easy to check:

$$E, \begin{array}{c} x : B \\ \langle k\rangle \\ x' : B' \end{array} \vdash \begin{array}{c} t(x) : A \\ \langle h\rangle \\ h(t(x)) : A' \end{array} \qquad E, \begin{array}{c} x : B \\ \langle k\rangle \\ x' : B' \end{array} \vdash \begin{array}{c} h(t(x)) : A' \\ A'^* \\ t'(k(x)) : A' \end{array} \qquad E, \begin{array}{c} x : B \\ \langle k\rangle \\ x' : B' \end{array} \vdash \begin{array}{c} t'(k(x)) : A' \\ A'^* \\ t'(x') : A' \end{array}$$

From the claim we derive:

$$E \vdash \begin{array}{c} \lambda(x : B)t(x) : B \to A \\ \langle k\rangle \to \langle h\rangle \\ \lambda(x' : B')t'(x') : B' \to A' \end{array}$$

and then by (Rel Val Eta), (Rel Val Saturation Lft), and (Rel Val Saturation Rht): $E \vdash \begin{array}{c} t : B \to A \\ \langle k\rangle \to \langle h\rangle \\ t' : B' \to A' \end{array}$.

Conversely, suppose $E \vdash \begin{array}{c} t : B \to A \\ \langle k\rangle \to \langle h\rangle \\ t' : B' \to A' \end{array}$, then by weakening, (Rel Val x*R* y), and (Rel Val Appl) :

$$E, \begin{array}{c} x : B \\ \langle k\rangle \\ x' : B' \end{array} \vdash \begin{array}{c} t(x) : A \\ \langle h\rangle \\ t'(x') : A' \end{array}$$

Using $E,\ \begin{array}{c} x : B \\ \langle k\rangle \\ x' : B' \end{array} \vdash \begin{array}{c} t'(k(x)) : A' \\ A'^{*} \\ t'(x') : A' \end{array}$ we have $E,\ \begin{array}{c} x : B \\ \langle k\rangle \\ x' : B' \end{array} \vdash \begin{array}{c} t(x) : A \\ \langle h\rangle \\ t'(k(x)) : A' \end{array}$ by (Rel Val Symm) and (Rel Val Saturation Rht), and by (Rel Val FRel Elim) we get:

$$E,\ \begin{array}{c} x : B \\ \langle k\rangle \\ x' : B' \end{array} \vdash \begin{array}{c} h(t(x)) : A' \\ A'^{*} \\ t'(k(x)) : A' \end{array}$$

By weakening and renaming we also get:

$$E,\ \begin{array}{c} x : B \\ B^{*} \\ y : B \end{array},\ \begin{array}{c} x_1 : B \\ \langle k\rangle \\ x'_1 : B' \end{array} \vdash \begin{array}{c} h(t(x_1)) : A' \\ A'^{*} \\ t'(k(x_1)) : A' \end{array}.$$

Notice that $E,\ \begin{array}{c} x : B \\ B^{*} \\ y : B \end{array} \vdash \begin{array}{c} x : B \\ \langle k\rangle \\ k(x) : B' \end{array}$. We can use this to substitute into the judgment above, replacing x_1 with x and (vacuously) x'_1 with k(x), obtaining:

$$E,\ \begin{array}{c} x : B \\ B^{*} \\ y : B \end{array} \vdash \begin{array}{c} h(t(x)) : A' \\ A'^{*} \\ t'(k(x)) : A' \end{array}$$

By assumption, we can equate x and y, to derive $E,\ \begin{array}{c} x : B \\ B^{*} \\ y : B \end{array} \vdash \begin{array}{c} h(t(x)) : A' \\ A'^{*} \\ t'(k(y)) : A' \end{array}$.

Finally we get $E \vdash \begin{array}{c} t;h : B \to A' \\ (B \to A')^{*} \\ k;t' : B \to A' \end{array}$ using (Rel Val Fun).

□

3.4 Extensional equality

We define a notion of extensional equality between relations. This notion can be formally added to R^1, or it can be left at the metalevel, as we do here. Intuitively, two relations are extensionally equal if they have the same graph, and they are extensionally inverses if the graph of one is the inverse of the graph of the other.

Definition (Extensional equality)
We say that R and S are extensionally equal, and we write:

$$E \vdash R \begin{array}{c} A \\ =_e \\ B \end{array} S$$

if $E,\ \begin{array}{c} x{:}A \\ R \\ y{:}B \end{array} \vdash \begin{array}{c} x{:}A \\ S \\ y{:}B \end{array}$ and $E,\ \begin{array}{c} x{:}A \\ S \\ y{:}B \end{array} \vdash \begin{array}{c} x{:}A \\ R \\ y{:}B \end{array}$.

We say that R and S are extensionally inverses, and we write :

$$E \vdash R \begin{array}{c} A \\ =_e^{op} \\ B \end{array} S$$

if $E,\ \overset{x:A}{\underset{y:B}{R}} \vdash \overset{y:B}{\underset{x:A}{S}}$ and $E,\ \overset{y:B}{\underset{x:A}{S}} \vdash \overset{x:A}{\underset{y:B}{R}}$.

In both definitions we assume that x and y are fresh.

We state a few properties of extensional equality. The proofs of the first two lemmas are omitted.

Lemma (Transitivity of extensional equality)

$$E \vdash R \overset{A}{\underset{A'}{=_e}} R' \ ,\ E \vdash R' \overset{A}{\underset{A'}{=_e}} R'' \ \Rightarrow\ E \vdash R \overset{A}{\underset{A'}{=_e}} R''$$

Lemma (Extensional congruence)

$$E \vdash R \overset{A}{\underset{A'}{=_e}} R' \ ,\ E \vdash S \overset{B}{\underset{B'}{=_e}} S' \ \Rightarrow\ E \vdash R \to S \overset{A \to B}{\underset{A' \to B'}{=_e}} R' \to S'$$

$$E,\ \overset{X}{\underset{X'}{W}} \vdash R \overset{A}{\underset{A'}{=_e}} R' \ \Rightarrow\ E \vdash \forall(W)R \overset{\forall(X)A}{\underset{\forall(X')A'}{=_e}} \forall(W)R' \quad (X \notin A', R, R' \text{ and } X' \notin A, R, R')$$

Lemma (Identity relations)

$$E \vdash A \ \Rightarrow\ E \vdash A^* \overset{A}{\underset{A}{=_e}} \langle\lambda(x:A)x\rangle$$

Proof

In one direction, we have $E,\ \overset{x:A}{\underset{y:A}{A^*}} \vdash y:A$ by (Rel Val R y), and $E,\ \overset{x:A}{\underset{y:A}{A^*}} \vdash \overset{y:A}{\underset{(\lambda(x:A)x)y:A}{\langle\lambda(x:A)x\rangle}}$ by (Rel Val FRel Intro). Hence $E,\ \overset{x:A}{\underset{y:A}{A^*}} \vdash \overset{x:A}{\underset{y:A}{\langle\lambda(x:A)x\rangle}}$ follows by saturation, (Rel Val xR y), and (Rel Val Beta).

For the converse direction we use (Rel Val xR y) and (Rel Val FRel Elim) to obtain:

$$E, \overset{x:A}{\underset{y:A}{\langle\lambda(x:A)x\rangle}} \vdash \overset{(\lambda(x:A)x)x:A}{\underset{y:A}{A^*}}$$

hence, via (Rel Val Beta), we have $E, \overset{x:A}{\underset{y:A}{\langle\lambda(x:A)x\rangle}} \vdash \overset{x:A}{\underset{y:A}{A^*}}$.

□

Lemma (Identity substitution)

$$E,\ X,\ E' \vdash A \ \Rightarrow\ E,\ X,\ E' \vdash A^* \overset{A}{\underset{A}{=_e}} A^*\{X \leftarrow \langle\lambda(x:X)x\rangle\}$$

Proof
By induction on the structure of A, using lemmas (Identity relations) and (Extensional congruence).
□

3.5 A commutation property

The third technical tool concerns covariant types. We say that a type A is covariant in X when X occurs only positively in A. For example, $(X \to Y) \to X$ is covariant in X. Symmetrically, A is contravariant in X when X occurs only negatively in A (as Y in the type above). A type A depending on X (the other free variables being considered as fixed parameters) may be viewed as a map $B \mapsto A\{X{\leftarrow}B\}$ from types to types. When A is covariant in X, it determines a (covariant) functor, which associates with any h:B→B' a term A{X←h} of type A{X←B}→A{X←B'}. When A is contravariant in X, it determines a contravariant functor, which associates with any h:B→B' a term A{X←h} of type A{X←B'}→A{X←B}. We use the following notation:

If $E \vdash^F a : A' \to A$ and $E \vdash^F b : B \to B'$, then $a \to b$ stands for:

$\lambda(x : A \to B)\lambda(y' : A')b(x(a(y')))$ which has type $(A \to B) \to (A' \to B')$

If $E, X \vdash^F a : B \to B'$, then $\forall(X)a$ stands for:

$\lambda(x : \forall(X)B)\lambda(X)\, a(x(X))$ which has type $(\forall(X)B) \to (\forall(X)B')$

Definition (Types as functors)
Suppose that $E, X \vdash^F A$, where A is covariant or contravariant in X, and consider the environment E, Y, Y', h: Y→Y'. We define $A\{X \leftarrow h\}$ as follows, by induction on A:

$$X\{X \leftarrow h\} = h$$
$$Y\{X \leftarrow h\} = \lambda(y : Y)y \qquad (Y \neq X)$$
$$(A_1 \to A_2)\{X \leftarrow h\} = (A_1\{X \leftarrow h\}) \to (A_2\{X \leftarrow h\})$$
$$(\forall(Y)A_1)\{X \leftarrow h\} = \forall(Y)A_1\{X \leftarrow h\}$$

The next lemmas state that the substitution just defined yields well-typed terms and preserves identities. We omit the proof of these lemmas, as well as the statement that A preserves compositions.

Lemma (Functor well-formedness)
If $E, X \vdash^F A$, where A is covariant in X, then, for Y, Y', and h fresh:

$$E, Y, Y', h : Y \to Y' \vdash^F A\{X \leftarrow h\} : A\{X \leftarrow Y\} \to A\{X \leftarrow Y'\}$$

If $E, X \vdash^F A$, where A is contravariant in X, then, for Y, Y', and h fresh:

$$E, Y, Y', h : Y \to Y' \vdash^F A\{X \leftarrow h\} : A\{X \leftarrow Y'\} \to A\{X \leftarrow Y\}$$

Lemma (Functors preserve identity)
If $E, X \vdash^F A$, where A is covariant or contravariant in X, then:

$$E, X \vdash^F A\{X \leftarrow \lambda(x : X)x\} = \lambda(z : A)z : A \to A$$

Typically, in our proofs, we get relations of the form $A^*\{X{\leftarrow}\langle h\rangle\}$ from an application of (Rel Val Appl2), while $\langle A\{X{\leftarrow}h\}\rangle$ may be needed. The following lemma says that covariant functors commute with functional relations, so $A^*\{X{\leftarrow}\langle h\rangle\}$ can be transformed into $\langle A\{X{\leftarrow}h\}\rangle$.

Lemma (Commutation of ⟨-⟩)
Assume $E, X \vdash^F A$, where A is covariant in X, then, for Y, Y', and h fresh:

$$E,\ Y,\ Y',\ h:\ Y \to Y' \vdash^{R\,1} \begin{array}{c} A\{X \leftarrow B\} \\ A^*\{X \leftarrow \langle h\rangle\}\ =_e\ \langle A\{X \leftarrow h\}\rangle \\ A\{X \leftarrow B'\} \end{array}$$

Assume $E, X \vdash^{F} A$, where A is contravariant in X, then, for Y, Y', and h fresh:

$$E,\ Y,\ Y',\ h:\ Y \to Y' \vdash^{R\,1} \begin{array}{c} A\{X \leftarrow B\} \\ A^*\{X \leftarrow \langle h\rangle\}\ =_e^{op}\ \langle A\{X \leftarrow h\}\rangle \\ A\{X \leftarrow B'\} \end{array}$$

Proof

We prove the first claim only, using an idea due to Plotkin. The second one is proved similarly. By theorem (Partial relational interpretation of F), we derive from the first claim of lemma (Functor well-formedness):

$$E,\ \begin{array}{c} Y_1 \\ \mathit{Y} \\ Y_2 \end{array},\ \begin{array}{c} Y'_1 \\ \mathit{Y'} \\ Y'_2 \end{array},\ \begin{array}{c} h_1:\ Y_1 \to Y'_1 \\ \mathit{Y} \to \mathit{Y'} \\ h_2:\ Y_2 \to Y'_2 \end{array}\ \vdash\ \begin{array}{c} A\{X \leftarrow h_1\}:\ A\{X \leftarrow Y_1\} \to A\{X \leftarrow Y'_1\} \\ A^*\{X \leftarrow \mathit{Y}\} \to A^*\{X \leftarrow \mathit{Y'}\} \\ A\{X \leftarrow h_2\}:\ A\{X \leftarrow Y_2\} \to A\{X \leftarrow Y'_2\} \end{array}$$

We use two different substitution instances of this judgment to establish the claim. First, by lemma (Commuting squares), and by weakening and value substitution, with $\langle h\rangle$ for Y, Y' for $\mathit{Y'}$, h for h_1, and $\langle\lambda(y':Y')y'\rangle$ for h_2, we get:

$$E,\ Y,\ Y',\ h:\ Y \to Y' \vdash \begin{array}{c} A\{X \leftarrow h\}:\ A\{X \leftarrow Y\} \to A\{X \leftarrow Y'\} \\ A^*\{X \leftarrow \langle h\rangle\} \to A^*\{X \leftarrow Y'\} \\ A\{X \leftarrow \lambda(y'{:}\ Y')y'\}:\ A\{X \leftarrow Y'\} \to A\{X \leftarrow Y'\} \end{array}$$

By lemma (Functors preserve identity) and the soundness of F equalities in R, we can replace $A\{X \leftarrow \lambda(y'{:}\ Y')y'\}$ with $\lambda(z : A\{X \leftarrow Y'\})z$:

$$E,\ Y,\ Y',\ h:\ Y \to Y' \vdash \begin{array}{c} A\{X \leftarrow h\}:\ A\{X \leftarrow Y\} \to A\{X \leftarrow Y'\} \\ A^*\{X \leftarrow \langle h\rangle\} \to A^*\{X \leftarrow Y'\} \\ \lambda(z : A\{X \leftarrow Y'\})z:\ A\{X \leftarrow Y'\} \to A\{X \leftarrow Y'\} \end{array}$$

By weakening, (Rel Val Appl), and (Rel Val Beta), we have:

$$E,\ Y,\ Y',\ h:\ Y \to Y',\ \begin{array}{c} x:\ A\{X \leftarrow Y\} \\ A^*\{X \leftarrow \langle h\rangle\} \\ x'{:}\ A\{X \leftarrow Y'\} \end{array}\ \vdash\ \begin{array}{c} A\{X \leftarrow h\}(x):\ A\{X \leftarrow Y'\} \\ A^*\{X \leftarrow Y'\} \\ x'{:}\ A\{X \leftarrow Y'\} \end{array}$$

and by functional-relation introduction:

$$E,\ Y,\ Y',\ h:\ Y \to Y',\ \begin{array}{c} x:\ A\{X \leftarrow Y\} \\ A^*\{X \leftarrow \langle h\rangle\} \\ x'{:}\ A\{X \leftarrow Y'\} \end{array}\ \vdash\ \begin{array}{c} x:\ A\{X \leftarrow Y\} \\ \langle A\{X \leftarrow h\}\rangle \\ x'{:}\ A\{X \leftarrow Y'\} \end{array}$$

Our second substitution instance is with Y for Y, $\langle h\rangle$ for $\mathit{Y'}$, $\lambda(y{:}Y)y$ for h_1, and h for h_2:

$$E,\ Y,\ Y',\ h:\ Y \to Y' \vdash \begin{array}{c} A\{X \leftarrow \lambda(y : Y)y\}:\ A\{X \leftarrow Y\} \to A\{X \leftarrow Y\} \\ A^*\{X \leftarrow Y\} \to A^*\{X \leftarrow \langle h\rangle\} \\ A\{X \leftarrow h\}:\ A\{X \leftarrow Y\} \to A\{X \leftarrow Y'\} \end{array}$$

By a similar reasoning, and using (Rel Val R x), we get:

$$E,\ Y,\ Y',\ h:\ Y \to Y',\ \begin{array}{c} x:\ A\{X \leftarrow Y\} \\ \langle A\{X \leftarrow h\}\rangle \\ x'{:}\ A\{X \leftarrow Y'\} \end{array}\ \vdash\ \begin{array}{c} x:\ A\{X \leftarrow Y\} \\ A^*\{X \leftarrow \langle h\rangle\} \\ A\{X \leftarrow h\}(x):\ A\{X \leftarrow Y'\} \end{array}$$

Then the second half of the claim follows by (Rel Val xR y), (Rel Val FRel Elim), and (Rel Val Saturation Rht):

$$\mathrm{E,\ Y,\ Y',\ h: Y \to Y',}\quad \begin{array}{c} \mathrm{x: A\{X \leftarrow Y\}} \\ \langle \mathrm{A\{X \leftarrow h\}} \rangle \\ \mathrm{x': A\{X \leftarrow Y'\}} \end{array} \vdash \begin{array}{c} \mathrm{x: A\{X \leftarrow Y\}} \\ \mathrm{A^*\{X \leftarrow \langle h \rangle\}} \\ \mathrm{x': A\{X \leftarrow Y'\}} \end{array}$$

□

3.6 Properties of map

We first apply the technical tools developed in the last three subsections to the proof of two theorems about map. The statements of these theorems express interesting equations between polymorphic terms that can be interpreted as program transformations. The theorems have been proved semantically for closed terms by Wadler [Wadler 1989]. Mairson has also discussed the second of these theorems, and has argued for the need of structural induction (in his framework). As we have already stressed, our proofs are free of induction.

The F encoding of X-lists is

$$\mathrm{List\{X\}} \triangleq \forall(\mathrm{Y})\mathrm{Y} \to (\mathrm{X} \to \mathrm{Y} \to \mathrm{Y}) \to \mathrm{Y}$$

Then $\forall(\mathrm{Y})\mathrm{Y} \to (\mathrm{h} \to \mathrm{Y} \to \mathrm{Y}) \to \mathrm{Y}$, abbreviated as List{h}, is the encoding of the familiar map function of type $\forall(\mathrm{X})\forall(\mathrm{Y})(\mathrm{X} \to \mathrm{Y}) \to (\mathrm{List\{X\}} \to \mathrm{List\{Y\}})$, instantiated at B, B’, and applied to h. Thus we have:

$$\mathrm{List\{h\} = map(B)(B')(h)} \quad \text{(for h of type } \mathrm{B \to B'}\text{)}$$

One of Wadler’s theorems says: take a term of type $\forall(\mathrm{X})\mathrm{List\{X\}} \to \mathrm{List\{X\}}$, such as reverse, then the following square commutes:

$$\begin{tikzcd}
\mathrm{List\{B\}} \arrow[r, "\mathrm{reverse(B)}"] \arrow[d, "\mathrm{List\{h\}}"'] & \mathrm{List\{B\}} \arrow[d, "\mathrm{List\{h\}}"] \\
\mathrm{List\{B'\}} \arrow[r, "\mathrm{reverse(B')}"] & \mathrm{List\{B'\}}
\end{tikzcd}$$

That is, one may indifferently apply map to a list, and then reverse it, or first reverse it, and then apply map to the reversed list. The property actually has nothing to do with reverse. It applies to any term of the type of reverse. The following proposition is a direct generalization of this example.

Proposition (Commutation for polymorphic functions)
Let A and A’ be two types, such that $\mathrm{E, X} \vdash^{\mathrm{F}} \mathrm{A}$, $\mathrm{E, X} \vdash^{\mathrm{F}} \mathrm{A'}$, and A, A’ are covariant in X. Let $\mathrm{E} \vdash^{\mathrm{F}} \mathrm{t}: \forall(\mathrm{X})(\mathrm{A} \to \mathrm{A'})$ and $\mathrm{E} \vdash^{\mathrm{F}} \mathrm{h: B \to B'}$. Then the following diagram commutes:

$$\begin{tikzcd}
\mathrm{A\{X \leftarrow B\}} \arrow[r, "\mathrm{t(B)}"] \arrow[d, "\mathrm{A\{X \leftarrow h\}}"'] & \mathrm{A'\{X \leftarrow B\}} \arrow[d, "\mathrm{A'\{X \leftarrow h\}}"] \\
\mathrm{A\{X \leftarrow B'\}} \arrow[r, "\mathrm{t(B')}"] & \mathrm{A'\{X \leftarrow B'\}}
\end{tikzcd}$$

that is, formally:

$$\mathrm{E} \vdash^{R^1} \begin{array}{c} \mathrm{(t(B)\ ;\ A'\{X \leftarrow h\}) : A\{X \leftarrow B\} \to A'\{X \leftarrow B'\}} \\ \mathrm{(A\{X \leftarrow B\} \to A'\{X \leftarrow B'\})^*} \\ \mathrm{(A\{X \leftarrow h\}\ ;\ t(B')) : A\{X \leftarrow B\} \to A'\{X \leftarrow B'\}} \end{array}$$

Proof
By lemma (Commuting squares) the claim can be restated as:

$$E \vdash \begin{array}{c} t(B) :\ A\{X \leftarrow B\} \to A'\{X \leftarrow B\} \\ \langle A\{X \leftarrow h\}\rangle \to \langle A'\{X \leftarrow h\}\rangle \\ t(B') :\ A\{X \leftarrow B'\} \to A'\{X \leftarrow B'\} \end{array}$$

After two applications of lemma (Commutation of ⟨-⟩), the claim is reformulated as:

$$E \vdash \begin{array}{c} t(B) : (A \to A')\{X \leftarrow B\} \\ (A \to A')^*\{X \leftarrow \langle h\rangle\} \\ t(B') : (A \to A')\{X \leftarrow B'\} \end{array}$$

and follows by the identity extension property and (Rel Val Appl2).
□

We now proceed to derive a second theorem about map. Wadler has proved that any term m of the type $\forall(X)\forall(Y)(X \to Y) \to (List\{X\} \to List\{Y\})$ of map is the composition (in either order) of map and of a rearrangement function, like reverse. The rearrangement function is retrieved from m by instantiating X and Y to a same type, say X, and then by applying m(X)(X) to the identity on X; the resulting term has type $List\{X\} \to List\{X\}$.

Proposition (Map)
Let E stand for:

$$m : \forall(X)\forall(Y)(X \to Y) \to (List\{X\} \to List\{Y\}),\ X,\ Y,\ f : X \to Y$$

Then the following judgments are provable:

$$E \vdash^{R\,1} \begin{array}{c} m(X)(Y)(f) :\ List\{X\} \to List\{Y\} \\ (List\{X\} \to List\{Y\})^* \\ (m(X)(X)(\lambda(x:X)x)\ ;\ List\{f\}) :\ List\{X\} \to List\{Y\} \end{array}$$

$$E \vdash^{R\,1} \begin{array}{c} m(X)(Y)(f) :\ List\{X\} \to List\{Y\} \\ (List\{X\} \to List\{Y\})^* \\ (List\{f\}\ ;\ m(Y)(Y)(\lambda(y:Y)y)) :\ List\{X\} \to List\{Y\} \end{array}$$

Proof
Consider the following commuting square:

$$\begin{tikzcd} X \arrow[r, "g"] \arrow[d, "a"'] & Z \arrow[d, "b"] \\ X' \arrow[r, "g'"] & Z' \end{tikzcd}$$

Let

$$E_1\ =\ m : A,\ X,\ X',\ Z,\ Z',\ a : X \to X',\ b : Z \to Z',\ \begin{array}{c} g : X \to Z \\ \langle a\rangle \to \langle b\rangle \\ g' : X' \to Z' \end{array}$$

where A stands for $\forall(X)\forall(Y)(X \to Y) \to (List\{X\} \to List\{Y\})$. By (Rel FRel) we have:

$$E_1 \vdash \begin{array}{c} X \\ \langle a\rangle \\ X' \end{array} \qquad E_1 \vdash \begin{array}{c} Z \\ \langle b\rangle \\ Z' \end{array}$$

Hence by (Rel Val *R* x), (Rel Val Appl2), and (Rel Val Appl):

$$E_1 \vdash \begin{array}{c} m(X)(Z)(g) : List\{X\} \to List\{Z\} \\ List\{\langle a\rangle\} \to List\{\langle b\rangle\} \\ m(X')(Z')(g') : List\{X'\} \to List\{Z'\} \end{array}$$

(where List{⟨a⟩} stands for (List{X})*{X←⟨a⟩}), and by lemma (Commutation of ⟨-⟩):

$$E_1 \vdash \begin{array}{c} m(X)(Z)(g) : List\{X\} \to List\{Z\} \\ \langle List\{a\}\rangle \to \langle List\{b\}\rangle \\ m(X')(Z')(g') : List\{X'\} \to List\{Z'\} \end{array} .$$

In diagrammatic form, we have proved:

$$\begin{tikzcd} X \arrow[r, "g"] \arrow[d, "a"'] & Z \arrow[d, "b"] \\ X' \arrow[r, "g'"] & Z' \end{tikzcd} \Rightarrow \begin{tikzcd} List\{X\} \arrow[r, "m(X)(Z)(g)"] \arrow[d, "List\{a\}"'] & List\{Z\} \arrow[d, "List\{b\}"] \\ List\{X'\} \arrow[r, "m(X')(Z')(g')"] & List\{Z'\} \end{tikzcd}$$

Consider now the following substitution instances for X, X’, Z, Z’, a, b, g, and g’:

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "f"'] & Y \arrow[d, "\lambda(y:Y)y"] \\ Y \arrow[r, "\lambda(y:Y)y"] & Y \end{tikzcd} \text{ and } \begin{tikzcd} X \arrow[r, "\lambda(x:X)x"] \arrow[d, "\lambda(x:X)x"'] & X \arrow[d, "f"] \\ X \arrow[r, "f"] & Y \end{tikzcd}$$

The corresponding conclusion squares are:

$$\begin{tikzcd} List\{X\} \arrow[r, "m(X)(Y)(f)"] \arrow[d, "List\{f\}"'] & List\{Y\} \arrow[d, "List\{\lambda(y:Y)y\}"] \\ List\{Y\} \arrow[r, "m(Y)(Y)(\lambda(y:Y)y)"] & List\{Y\} \end{tikzcd}$$

and

$$\begin{tikzcd} List\{X\} \arrow[r, "m(X)(X)(\lambda(x:X)x)"] \arrow[d, "List\{\lambda(x:X)x\}"'] & List\{X\} \arrow[d, "List\{f\}"] \\ List\{X\} \arrow[r, "m(X)(Y)(f)"] & List\{Y\} \end{tikzcd}$$

They yield the two judgments of the statement, using lemma (Functors preserve identity).

□

3.7 Initial algebras

Given a type A covariant in X, an A-algebra is a pair of a type B and of a morphism t:A{X←B}→B. An A-algebra morphism from (B,t) to (B’,t’) is a term h:B→B’ such that t;h = A{X←h};t’. An initial A-algebra is an A-algebra (T,*in*) such that for any other A-algebra (B,t) there exists exactly one A-algebra morphism from (T,*in*) to (B,t). The goal of this subsection is to show that, given A covariant in X, the type

$$T = \forall(X)(A \to X) \to X$$

can be turned into an initial A-algebra. (See also [Wadler 1991].) Hence the initial algebras useful in programming (for example, that of natural numbers, see section 3.9) can be defined properly as polymorphic types. Böhm and Berarducci have used similar types to encode primitive recursion on (possibly heterogeneous) term algebras [Böhm, Berarducci 1985]. They obtain a completeness result that guarantees that the encoding of algebras is correct for closed terms.

We define:

$$\begin{array}{l} \mathit{fold} : \forall(X)(A \to X) \to (T \to X) = \\ \quad \lambda(X)\ \lambda(k : A \to X)\ \lambda(x : T)\ x(X)(k) \end{array}$$

$$\begin{array}{l} \mathit{in} : A\{X \leftarrow T\} \to T = \\ \quad \lambda(y : A\{X \leftarrow T\})\ \lambda(X)\ \lambda(k : A \to X) \\ \quad\quad k(A\{X \leftarrow \mathit{fold}(X)(k)\}(y)) \end{array}$$

Our first lemma states that $\mathit{fold}(X)(k)$ takes an algebra (X,k) to an algebra morphism $\lambda(x{:}T)x(X)(k)$ from (T, in) to (X,k).

Lemma (*in* morphism)
Assume $E, X \vdash^{F} A$ with A covariant in X. Then, if k is fresh:

$$E,\ X,\ k : A \to X \vdash^{R^1} \begin{array}{c} \mathit{in} : A\{X \leftarrow T\} \to T \\ \langle A\{X \leftarrow \mathit{fold}(X)(k)\}\rangle \to \langle \mathit{fold}(X)(k)\rangle \\ k : A \to X \end{array}$$

Proof
By lemma (Commuting squares), the statement is equivalent to the equality of $\mathit{in}\,;\mathit{fold}(X)(k)$ and $A\{X{\leftarrow}\mathit{fold}(X)(k)\}\ ;\ k$, which follows straightforwardly from the definitions of *fold* and *in*, using β rules.
□

The initiality of (T, in) means that if a is a morphism from (T, in) to (X,k), then a must equal $\mathit{fold}(X)(k)$. Before proving the initiality theorem, we establish two further lemmas.

Lemma (Algebra morphisms)
Assume $E, X \vdash^{F} A$ with A covariant in X. Then, if x, Y, Y’, h, t, and t’ are fresh:

$$E,\ x : T, Y, Y’,\ h : Y \to Y’,\ \begin{array}{c} t : A\{X \leftarrow Y\} \to Y \\ \langle A\{X \leftarrow h\}\rangle \to \langle h\rangle \\ t’ : A\{X \leftarrow Y’\} \to Y’ \end{array} \vdash^{R^1} \begin{array}{c} h(x(Y)(t)) : Y’ \\ Y’ \\ x(Y’)(t’) : Y’ \end{array}$$

or, diagrammatically:

$$\begin{tikzcd} A\{X \leftarrow Y\} \arrow[r, "t"] \arrow[d, "A\{X \leftarrow h\}"'] & Y \arrow[d, "h"] & & T \arrow[r, "\mathit{fold}(Y)(t)"] \arrow[d, equal] & Y \arrow[d, "h"] \\ A\{X \leftarrow Y’\} \arrow[r, "t’"] & Y’ \arrow[r, phantom, "\Rightarrow", shift left=20] & & T \arrow[r, "\mathit{fold}(Y’)(t’)"] & Y’ \end{tikzcd}$$

The left square of the diagram expresses that h is a morphism from (Y,t) to (Y’,t’).

Proof
By (Rel Val *R* x) and (Rel Val Appl2), we have:

$$E,\ x : T,\ Y, Y’, h : Y \to Y’ \vdash \begin{array}{c} x(Y) : (A\{X \leftarrow Y\} \to Y) \to Y \\ (A^*\{X \leftarrow \langle h\rangle\} \to \langle h\rangle) \to \langle h\rangle \\ x(Y’) : (A\{X \leftarrow Y’\} \to Y’) \to Y’ \end{array}$$

Then, by (Rel Val Appl), we obtain:

$$E,\ x:T, Y, Y', h:Y\to Y',\ \begin{array}{c} t: A\{X\leftarrow Y\}\to Y \\ A^{*}\{X\leftarrow\langle h\rangle\}\to\langle h\rangle \\ t': A\{X\leftarrow Y'\}\to Y' \end{array} \vdash \begin{array}{c} x(Y)(t): Y \\ \langle h\rangle \\ x(Y')(t'): Y' \end{array}$$

and by (Rel Val FRel Elim):

$$E,\ x:T, Y, Y', h:Y\to Y',\ \begin{array}{c} t: A\{X\leftarrow Y\}\to Y \\ A^{*}\{X\leftarrow\langle h\rangle\}\to\langle h\rangle \\ t': A\{X\leftarrow Y'\}\to Y' \end{array} \vdash \begin{array}{c} h(x(Y)(t)): Y' \\ Y' \\ x(Y')(t'): Y' \end{array}$$

The claim follows by lemma (Commutation of $\langle\text{-}\rangle$).
□

Lemma (x(T)(*in*))
Assume $E, X \vdash^{F} A$ with A covariant in X. Then, if x is fresh:

$$E, x:T \vdash^{R^1} \begin{array}{c} x(T)(\mathit{in}): T \\ T^{*} \\ x: T \end{array}$$

Proof
By lemma (*in* morphism) we obtain the following substitution instance of lemma (Algebra morphisms):

$$E,\ x:T,\ X,\ k: A\to X \vdash \begin{array}{c} \mathit{fold}(X)(k)(x(T)(\mathit{in})): X \\ X \\ x(X)(k): X \end{array}$$

It is obtained with the renaming Y'=X and the substitutions Y=T, h=*fold*(X)(k), t=*in*, t'=k. Hence by the definition of *fold*, and by β rules, we have:

$$E,\ x:T,\ X,\ k: A\to X \vdash \begin{array}{c} x(T)(\mathit{in})(X)(k): X \\ X \\ x(X)(k): X \end{array}$$

and the claim follows by η rules (with manipulations similar to those at the end of proposition (Terminal)).
□

Theorem (Initial algebras)
The algebra (T,*in*) is initial. That is, if $E, X \vdash^{F} A$ with A covariant in X, $E, X \vdash^{F} a : T\to X$, and k is fresh, then:

$$E, X,\ k: A\to X \vdash^{R^1} \begin{array}{c} \mathit{in}: A\{X\leftarrow T\}\to T \\ \langle A\{X\leftarrow a\}\rangle\to\langle a\rangle \\ k: A\to X \end{array} \quad\Rightarrow\quad E, X,\ k: A\to X \vdash^{R^1} \begin{array}{c} a: T\to X \\ (T\to X)^{*} \\ \mathit{fold}(X)(k): T\to X \end{array}$$

Proof
Using the assumption we obtain the following consequence of lemma (Algebra morphisms):

$$E,\ X,\ k: A\to X,\ x:T \ \vdash \begin{array}{c} a(x(T)(\mathit{in})): X \\ X \\ x(X)(k): X \end{array}$$

It is obtained with the renaming Y'=X and the substitutions Y=T, h=a, t=*in*, t'=k. By lemma (x(T)(*in*)) we can equate x(T)(in) and x:

$$\mathrm{E},\ \mathrm{X},\ \mathrm{k} : \mathrm{A} \to \mathrm{X},\ \mathrm{x} : \mathrm{T} \vdash \begin{array}{c} \mathrm{a(x) : X} \\ \mathrm{X} \\ \mathrm{x(X)(k) : X} \end{array}$$

Unfolding *fold*, we obtain:

$$\mathrm{E},\ \mathrm{X},\ \mathrm{k} : \mathrm{A} \to \mathrm{X},\ \mathrm{x} : \mathrm{T} \vdash \begin{array}{c} \mathrm{a(x) : X} \\ \mathrm{X} \\ \mathit{fold}\mathrm{(X)(k)(x) : X} \end{array}$$

Since:

$$\mathrm{E},\ \mathrm{X},\ \mathrm{k} : \mathrm{A} \to \mathrm{X},\ \begin{array}{c} \mathrm{x : T} \\ \mathrm{T^*} \\ \mathrm{x' : T} \end{array} \vdash \begin{array}{c} \mathit{fold}\mathrm{(X)(k)(x) : X} \\ \mathrm{X} \\ \mathit{fold}\mathrm{(X)(k)(x') : X} \end{array}$$

we can conclude using (Rel Val Saturation Lft), (Rel Val Fun), and (Rel Val Eta).
□

A consequence of initiality is that *in* is actually an isomorphism from $\mathrm{A}\{\mathrm{X} \leftarrow \mathrm{T}\}$ to T. Hence, the initial A-algebra is a solution for the fixpoint equation X = A{X}; the two halves of the isomorphism between T and A{X←T} are *in* and *out*, where *out* is defined as follows:

$$\begin{array}{l} \mathit{out} : \mathrm{T} \to \mathrm{A}\{\mathrm{X} \leftarrow \mathrm{T}\} = \\ \quad \mathit{fold}(\mathrm{A}\{\mathrm{X} \leftarrow \mathrm{T}\})(\mathrm{A}\{\mathrm{X} \leftarrow \mathit{in}\}) \end{array}$$

Polymorphic types thus suffice to encode covariant recursive types. In particular, if X does not occur in A, then A and $\forall(\mathrm{X})(\mathrm{A} \to \mathrm{X}) \to \mathrm{X}$ are isomorphic.

3.8 Products and coproducts

In system R^1 the following properties are provable:

(1) $\forall(\mathrm{X})\mathrm{X} \to \mathrm{X}$ is terminal (as already proved),
(2) $\forall(\mathrm{X})(\mathrm{B} \to \mathrm{B'} \to \mathrm{X}) \to \mathrm{X}$ is a product of B and B',
(3) $\forall(\mathrm{X})\mathrm{X}$ is initial,
(4) $\forall(\mathrm{X})(\mathrm{B} \to \mathrm{X}) \to (\mathrm{B'} \to \mathrm{X}) \to \mathrm{X}$ is a coproduct of B and B'.

If the existence of products and coproducts is already assumed, these results can all be seen as instances of the isomorphism between A and $\forall(\mathrm{X})(\mathrm{A} \to \mathrm{X}) \to \mathrm{X}$, for A constant in X. For example, taking A = B×B', and using the isomorphism between B×B'→X and B→B'→X, we get (2). But neither system F nor system R^1 have "pre-existent" finite products and coproducts. Hence each of the properties (1)-(4) has to be proved separately, and independently of the initial algebra theorem. We discuss binary products only.

Proposition (Product)
The type $\forall(\mathrm{X})(\mathrm{B} \to \mathrm{B'} \to \mathrm{X}) \to \mathrm{X}$ is a product of B and B'.

Proof
We adapt a semantic proof communicated to us by Wadler. It is well known that, when $\mathrm{E} \vdash^{\mathrm{F}} \mathrm{b} : \mathrm{B}$, $\mathrm{E} \vdash^{\mathrm{F}} \mathrm{b'} : \mathrm{B'}$, the following laws are provable in F :

$$\mathrm{E} \vdash^{\mathrm{F}} \mathrm{fst(pair(b)(b'))} = \mathrm{b : B} \qquad\qquad \mathrm{E} \vdash^{\mathrm{F}} \mathrm{snd(pair(b)(b'))} = \mathrm{b' : B'}$$

with:

$$\begin{aligned}
&A = \forall(X)(B \to B' \to X) \to X \\
&\text{fst} = \lambda(a : A)a(B)(\lambda(x : B)\lambda(x' : B')x) \\
&\text{snd} = \lambda(a : A)a(B')(\lambda(x : B)\lambda(x' : B')x') \\
&\text{pair} = \lambda(b : B)\lambda(b' : B')\lambda(X)\lambda(k : B \to B' \to X)k(b)(b')
\end{aligned}$$

What remains to be checked is surjective pairing:

$$(\text{SP}) \qquad E,\ a : A \vdash^{R^1} \begin{array}{c} \text{pair}(\text{fst}(a))(\text{snd}(a)) : A \\ A^* \\ a : A \end{array}$$

We follow the same proof pattern as for theorem (Initial algebras). We get the following counterpart of lemma (Algebra morphisms):

$$(1) \qquad E,\ a : A,\ X,\ X',\ h : X \to X',\ \begin{array}{c} k : B \to B' \to X \\ B^* \to B'^* \to \langle h\rangle \\ k' : B \to B' \to X' \end{array} \vdash \begin{array}{c} a(X)(k) :\ X \\ \langle h\rangle \\ a(X')(k') :\ X' \end{array}$$

In (1), much as in lemma (Commuting squares), the assumption $\begin{array}{c} k : B \to B' \to X \\ B^* \to B'^* \to \langle h\rangle \\ k' : B \to B' \to X' \end{array}$ amounts to asserting that k' is λ(b:B)λ(b':B')h(k(b)(b')). By instantiating (1) to X=A, k=pair, and h= $\lambda(a : A)a(X')(k')$, we get:

$$E, a : A, X', k' : B \to B' \to X' \vdash \begin{array}{c} a(A)(\text{pair}) : A \\ \langle \lambda(a : A)a(X')(k')\rangle \\ a(X')(k') : X' \end{array}$$

and from there, the following counterpart of lemma (x(T)(in)) is obtained:

$$(2) \qquad E,\ a : A \vdash \begin{array}{c} a(A)(\text{pair}) : A \\ A^* \\ a : A \end{array}$$

We instantiate (1) again, with X=X'=A, k=k'=pair, and h= $\lambda(a : A)\text{pair}(\text{fst}(a))(\text{snd}(a))$:

$$E, a : A \vdash \begin{array}{c} a(A)(\text{pair}) : A \\ \langle \lambda(a : A)\text{pair}(\text{fst}(a))(\text{snd}(a))\rangle \\ a(A)(\text{pair}) : A \end{array}$$

Combining this with (2), we get:

$$E, a : A \vdash \begin{array}{c} a : A \\ \langle \lambda(a : A)\text{pair}(\text{fst}(a))(\text{snd}(a))\rangle \\ a : A \end{array}$$

and the claim follows by (Rel Val FRel Elim).
□

There is a simpler proof of this theorem if the system R^1 is extended to support ternary relations as well as binary relations. We suggest how such an extension could be defined. The following judgments and rules would be added, among others:

(Rel FRel2)

$$\frac{E \vdash c : A \to B \to C}{E \vdash \begin{array}{c} A, B \\ \langle c\rangle_2 \\ C \end{array}}$$

(Rel Val FRel2 Intro)

$$\frac{E \vdash c : A \to B \to C \qquad E \vdash a : A \qquad E \vdash b : B}{E \vdash \begin{array}{c} a : A, b : B \\ \langle c\rangle_2 \\ c(a)(b) : C \end{array}}$$

In this system, the proof of surjective pairing goes as follows. We have, by (Rel Val $\mathcal{R}$ x) and by a ternary version of (Rel Val Appl2):

$$E,\ a : A,\ X,\ k : B \to B' \to X \vdash \begin{array}{c} a(B) : (B \to B' \to B) \to B\ ,\ a(B') : (B \to B' \to B') \to B' \\ (B^* \to B'^* \to \langle k\rangle_2) \to \langle k\rangle_2 \\ a(X) : (B \to B' \to X) \to X \end{array}$$

On the other hand,

$$E,\ a : A,\ X,\ k : B \to B' \to X \vdash \begin{array}{c} \lambda(x : B)\lambda(x' : B')x : B \to B' \to B\ ,\ \lambda(x : B)\lambda(x' : B')x' : B \to B' \to B' \\ B^* \to B'^* \to \langle k\rangle_2 \\ k : B \to B' \to X \end{array}$$

is an instance of a variant of lemma (Commuting squares), so that we obtain by ternary-relation application:

$$E,\ a : A,\ X,\ k : B \to B' \to X \vdash \begin{array}{c} a(B)(\lambda(x : B)\lambda(x' : B')x) : B\ ,\ a(B')(\lambda(x : B)\lambda(x' : B')x') : B' \\ \langle k\rangle_2 \\ a(X)(k) : X \end{array}$$

and by ternary-relation elimination:

$$E,\ a : A,\ X,\ k : B \to B' \to X \vdash \begin{array}{c} k(a(B)(\lambda(x : B)\lambda(x' : B')x))(a(B')(\lambda(x : B)\lambda(x' : B')x')) : X \\ X \\ a(X)(k) : X \end{array}$$

Then (Rel Val Beta) and (Rel Val Saturation Lft) allow us to replace

$$k(a(B)(\lambda(x : B)\lambda(x' : B')x))(a(B')(\lambda(x : B)\lambda(x' : B')x')) \text{ with } pair(fst(a))(snd(a))(X)(k)$$

and the claim follows as in the proof of propositions (Constant) and (Terminal).

We end this section with an application. Using the properties of products, we obtain a theorem about booleans. In F, the only two closed normal forms of type Bool are:

$$\begin{array}{l} true = \lambda(Z)\lambda(x : Z)\lambda(y : Z)x \\ false = \lambda(Z)\lambda(x : Z)\lambda(y : Z)y \end{array}$$

We prove that any two functions from Bool to the same type A that coincide on true and false are equal. For example, the terms (λ(x:Bool) 3) and (λ(x:Bool) if x then 3 else 3) are provably equal.

Proposition (Bool)
Let $E \vdash A$, $E \vdash b\!: Bool \to A$, $E \vdash b'\!: Bool \to A$. Then:

$$E \vdash \begin{array}{c} b(true) :\ A \\ A^* \\ b'(true) :\ A \end{array} \ \wedge\ E \vdash \begin{array}{c} b(false) :\ A \\ A^* \\ b'(false) :\ A \end{array} \ \Rightarrow\ E \vdash \begin{array}{c} b :\ Bool \to A \\ (Bool \to A)^* \\ b' :\ Bool \to A \end{array}$$

Proof
We only sketch the argument. We exploit the following isomorphisms: Bool is isomorphic to 1+1, (C+C')$\to$A is isomorphic to (C$\to$A)$\times$(C'$\to$A) for any C and C', and 1$\to$A is isomorphic to A. Hence Bool$\to$A is isomorphic to A$\times$A. The two halves of the isomorphism are:

$$\begin{array}{l} i\ =\ \lambda(f : Bool \to A)\lambda(Y)\lambda(g : A \to A \to Y) \\ \qquad g(f(true))(f(false)) \\ j\ =\ \lambda(h : \forall(Y)(A \to A \to Y) \to Y)\lambda(x : Bool) \\ \qquad h(A)(x(A)) \end{array}$$

One then observes that i(b) and i(b’) are equal, since the argument f occurs only in the contexts f(true) and f(false) in i. Finally, the equality of i(b) and i(b’) entails the equality of b and b’, since b is equal to j(i(b)) and b’ is equal to j(i(b’)).
□

3.9 Some applications of initiality

We briefly mention two other consequences of the general theorems about initiality and products.

- The type Nat = $\forall(X)(X \to X) \to X \to X$ of Church integers is the initial A-algebra for A=1+X, hence Nat and 1+Nat are provably isomorphic in R.
- The type List{Y} = $\forall(X)X \to (Y \to X \to X) \to X$ of lists is the initial A-algebra for A=1+(Y×X), covariant in variable X. Hence List{Y} and 1+(Y×List{Y}) are provably isomorphic.

We concentrate on the type Nat for the rest of this section. If n has type Nat, we can prove the following naturality condition, similar to the statement of lemma (Algebra morphisms):

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "H"'] & A \arrow[d, "H"] & & A \arrow[r, "n(A)(f)"] \arrow[d, "H"'] & A \arrow[d, "H"] \\
B \arrow[r, "F"] & B \arrow[r, phantom, "\Rightarrow" near start, yshift=20pt] & & B \arrow[r, "n(B)(F)"] & B
\end{tikzcd}$$

This implication has several interesting instantiations. Recall the classical encodings of arithmetical operations in F:

$$\begin{array}{l}\mathit{succ} : \text{Nat} \to \text{Nat} = \\ \quad \lambda(\text{n : Nat})\lambda(X)\ \lambda(\text{f} : X \to X)\ \lambda(\text{x} : X)\ \text{f}(\text{n}(X)(\text{f})(\text{x}))\end{array}$$

$$\begin{array}{l}\mathit{zero} : \text{Nat} = \\ \quad \lambda(X)\ \lambda(\text{f} : X \to X)\lambda(\text{x} : X)\text{x}\end{array}$$

$$\begin{array}{l}\mathit{add} : \text{Nat} \to \text{Nat} \to \text{Nat} = \\ \quad \lambda(\text{m : Nat})\lambda(\text{n : Nat})\text{m}(\text{Nat})(\mathit{succ})(\text{n})\end{array}$$

$$\begin{array}{l}\mathit{mult} : \text{Nat} \to \text{Nat} \to \text{Nat} = \\ \quad \lambda(\text{m : Nat})\lambda(\text{n : Nat})\text{m}(\text{Nat})(\mathit{add}(\text{n}))(\mathit{zero})\end{array}$$

$$\begin{array}{l}\mathit{exp} : \text{Nat} \to \text{Nat} \to \text{Nat} = \\ \quad \lambda(\text{m : Nat})\lambda(\text{n : Nat})\text{m}(\text{Nat})(\mathit{mult}(\text{n}))(\mathit{succ}(\mathit{zero}))\end{array}$$

In R^1 we can prove:

$$\text{n : Nat} \vdash \begin{array}{c}\text{n : Nat} \\ \text{Nat}\,* \\ \text{n}(\text{Nat})(\mathit{succ})(\mathit{zero}) : \text{Nat}\end{array}$$

$$\text{m : Nat, n : Nat} \vdash \begin{array}{c}\mathit{add}(\text{m})(\text{n}) : \text{Nat} \\ \text{Nat}\,* \\ \lambda(X)\lambda(\text{f} : X \to X)\lambda(\text{x} : X)\text{m}(X)(\text{f})(\text{n}(X)(\text{f})(\text{x})) : \text{Nat}\end{array}$$

$$\text{m : Nat, n : Nat} \vdash \begin{array}{c}\mathit{mult}(\text{m})(\text{n}) : \text{Nat} \\ \text{Nat}\,* \\ \lambda(X)\lambda(\text{f} : X \to X)\text{m}(X)(\text{n}(X)(\text{f})) : \text{Nat}\end{array}$$

$$\mathrm{m : Nat,\ n : Nat} \vdash \begin{array}{c} \mathit{exp}(\mathrm{m})(\mathrm{n}) : \mathrm{Nat} \\ \mathrm{Nat}^{*} \\ \lambda(\mathrm{X})\mathrm{m}(\mathrm{X} \to \mathrm{X})(\mathrm{n}(\mathrm{X})) : \mathrm{Nat} \end{array}$$

Paulin-Mohring has pointed out to us that these equalities justify optimizations found in various higher-order type systems.

3.10 On erasures

We end section 3 with a collection of examples of a somewhat different flavor. They are all examples of a general “erasure conjecture”. Roughly, the conjecture states that two F terms having the same type in the same environment and having the same erasure are provably equal in R .

The erasure of an F term is the untyped term obtained by erasing all its type information. Formally:

erase(x) = x
erase(a(b)) = erase(a)(erase(b))
erase(λ(x:A)a) = λ(x) erase(a)
erase(a(A)) = erase(a)
erase(λ(X)a) = erase(a)

The precise formulation of the conjecture is:

Conjecture
If $\mathrm{E} \vdash^{\mathrm{F}} \mathrm{a} : \mathrm{A}$, $\mathrm{E} \vdash^{\mathrm{F}} \mathrm{b} : \mathrm{A}$, and erase(a) = erase(b), then:

$$\mathrm{E} \vdash^{R^1} \begin{array}{c} \mathrm{a} : \mathrm{A} \\ \mathrm{A}^{*} \\ \mathrm{b} : \mathrm{A} \end{array}$$

If the conjecture holds, it gives precise evidence that Reynolds's notion of parametricity, which our formal system captures in syntax, reflects the intuition that types do not matter in computations of polymorphic programs.

Here we neither prove nor disprove the conjecture, but simply verify some instances. The first instance is the R analogue of Axiom (C) considered in [Longo, Milstead, Soloviev 1993].

Instance 1
Let $\mathrm{E} \vdash^{\mathrm{F}} \mathrm{a} : \forall(\mathrm{X})\mathrm{A}$, where $\mathrm{X} \notin \mathrm{A}$, and let $\mathrm{E} \vdash^{\mathrm{F}} \mathrm{B}$ and $\mathrm{E} \vdash^{\mathrm{F}} \mathrm{C}$. Then:

$$\mathrm{E} \vdash^{R^1} \begin{array}{c} \mathrm{a(B)} : \mathrm{A} \\ \mathrm{A}^{*} \\ \mathrm{a(C)} : \mathrm{A} \end{array}$$

Proof
We show how to prove:

$$\mathrm{E} \vdash \begin{array}{c} \mathrm{a}(\forall(\mathrm{X})\mathrm{X}) : \mathrm{A} \\ \mathrm{A}^{*} \\ \mathrm{a(B)} : \mathrm{A} \end{array} \quad \text{and} \quad \mathrm{E} \vdash \begin{array}{c} \mathrm{a}(\forall(\mathrm{X})\mathrm{X}) : \mathrm{A} \\ \mathrm{A}^{*} \\ \mathrm{a(C)} : \mathrm{A} \end{array}$$

The desired result follows from (Rel Val Symm) and (Rel Val Saturation Lft). We derive the first judgment; the other derivation is similar. By the identity extension property, we have $\mathrm{E} \vdash^{R^1} \mathrm{a} : \forall(\mathrm{X})\mathrm{A}$. Moreover, (Rel FRel) yields:

$$\mathrm{E} \vdash \begin{array}{c} \forall(\mathrm{X})\mathrm{X} \\ \langle \lambda(\mathrm{x} : \forall(\mathrm{X})\mathrm{X})\mathrm{x}(\mathrm{B}) \rangle \\ \mathrm{B} \end{array}$$

We conclude using (Rel Val Appl2).
□

Instance 2

$$x:\ \forall(X)X \vdash^{R\,1} \begin{array}{c} x(\forall(X)X):\ \forall(X)X \\ (\forall(X)X)^{*} \\ x:\ \forall(X)X \end{array}$$

Proof
We start by constructing a functional relation:

$$X \vdash \begin{array}{c} \forall(Y)Y \\ \langle\lambda(x:\forall(Y)Y)x(X)\rangle \\ X \end{array}$$

By applying (Rel Val R x) and (Rel Val Appl2) we get:

$$x:\forall(Y)Y,\ X \vdash \begin{array}{c} x(\forall(Y)Y):\ \forall(Y)Y \\ \langle\lambda(x:\forall(Y)Y)x(X)\rangle \\ x(X):\ X \end{array}$$

and (Rel Val FRel Elim) leads to:

$$x:\forall(Y)Y,\ X \vdash \begin{array}{c} x(\forall(Y)Y)(X):\ X \\ X \\ x(X):\ X \end{array}$$

The result then follows as in propositions (Constant) and (Terminal), using (Rel Val Eta2).
□

A simple variant of this proof yields:

Instance 3
Assume that $E \vdash^{F} a{:}A$, with $X \notin A$, and x fresh.

$$E,\ x:\ \forall(X)A \to X \vdash^{R\,1} \begin{array}{c} x(\forall(X)X)(a):\ \forall(X)X \\ (\forall(X)X)^{*} \\ \lambda(X)\ x(X)(a):\ \forall(X)X \end{array}$$

The final instance is based on two different ways of assigning the type $(\forall(X)X \to X) \to (\forall(X)X \to X)$ to the untyped term $\lambda(x)\ x(x)$:

Instance 4

$$x:\ \forall(X)X \to X \vdash^{R\,1} \begin{array}{c} x(\forall(X)X \to X)(x):\ \forall(X)X \to X \\ (\forall(X)X \to X)^{*} \\ \lambda(X)\ x(X \to X)\big(x(X)\big):\ \forall(X)X \to X \end{array}$$

Of course R yields far more equations than the ones arising from the conjecture. For example f(A)(a) and f(B)(b) are equal for any f: $\forall(X)X \to \text{Bool}$, since $\forall(X)X \to \text{Bool}$ contains only constant functions (see section 3.1). Here a and b can be any terms, of types A and B, respectively. In particular the terms f(A)(a) and f(B)(b) need not have the same erasure.

4. Conclusions

After working with R for some time, we feel that it is a useful system, with reasonable syntactic properties. In particular we are able to prove theorems and metatheorems in full generality for open terms. However, the power of R, in both syntactic and semantic terms, deserves further exploration.

In the realm of syntax, we are particularly interested in the conjecture discussed in section 3.10 that if two F terms have the same erasure and the same type then they are provably equal in R.

As for semantics, we intend to develop a model of R based on the per model of [Bainbridge, *et al.* 1990]. In the standard per model, universal quantification over types is interpreted with an intersection over pers; in contrast, in the per model of [Bainbridge, *et al.* 1990], universal quantification over types is interpreted with an intersection over saturated relations. This modification of the per model leads to a simple proof of soundness for the rules (Rel Val R x) and (Rel Val R y), and for all the other rules of R. On the other hand, the work of Hasegawa [Hasegawa 1991] and Hyland, Robinson, and Rosolini [Hyland, Robinson, Rosolini 1990] suggest that the standard per model itself, or closely related ones, may validate those rules.

As mentioned in the introduction, system $F_{<:}$ [Cardelli, *et al.* 1991] captures some aspects of parametricity. An extension of R with subtyping may yield an encoding of $F_{<:}$ and provide a basis for studying parametricity in languages with subtyping. An analogous extension of a logic for parametric polymorphism is carried out in [Plotkin, Abadi, Cardelli 1993].

Acknowledgments

We would like to thank Roberto Bellucci, Ryu Hasegawa, Christine Paulin-Mohring, Gordon Plotkin, and Phil Wadler for helpful discussions.

References

[Abadi, Cardelli, Curien 1993] M. Abadi, L. Cardelli, and P.-L. Curien. **Formal Parametric Polymorphism**. *Proc. 20th Annual ACM Symposium on Principles of Programming Languages*.

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

[Böhm, Berarducci 1985] C. Böhm and A. Berarducci, **Automatic synthesis of typed λ-programs on term algebras**. *Theoretical Computer Science* **39**, 135-154.

[Cardelli, *et al.* 1991] L. Cardelli, J.C. Mitchell, S. Martini, and A. Scedrov. **An extension of system F with subtyping**. *Proc. Theoretical Aspects of Computer Software.* Lecture Notes in Computer Science 526. Springer-Verlag.

[de Bruijn 1972] N.G. de Bruijn, **Lambda-calculus notation with nameless dummies**. *Indag. Math.* **34**(5), 381-392.

[Girard, Lafont, Taylor 1989] J.-Y. Girard, Y. Lafont, and P. Taylor, **Proofs and types**. Cambridge University Press.

[Hasegawa 1991] R. Hasegawa. **Parametricity of extensionally collapsed term models of polymorphism and their categorical properties**. *Proc. Theoretical Aspects of Computer Software*. Lecture Notes in Computer Science 526. Springer-Verlag.

[Hasegawa 1992] R. Hasegawa, **Categorical data types in parametric polymorphism**. Manuscript.

[Hyland, Robinson, Rosolini 1990] J.M.E. Hyland, E.P. Robinson, and G. Rosolini. **Algebraic types in PER models**. *Proc. Mathematical Foundations of Programming Semantics*. Lecture Notes in Computer Science 442. Springer-Verlag.

[Longo, Milstead, Soloviev 1993] G. Longo, K. Milstead, and S. Soloviev, **The genericity theorem and the notion of parametricity in the polymorphic λ-calculus**. In *Böhm Festschrift*. Cambridge University Press.

[Longo, Moggi 1991] G. Longo and E. Moggi, **Constructive natural deduction and its 'ω-set' interpretation**. *Mathematical Structures in Computer Science* **1**(2).

[Ma 1992] Q.-Y. Ma. **Parametricity as subtyping**. *Proc. 19th Annual ACM Symposium on Principles of Programming Languages*.

[Ma, Reynolds 1991] Q.-Y. Ma and J. Reynolds. **Types, abstraction, and parametric polymorphism, part 2**. *Proc. Mathematical Foundations of Programming Semantics*. Springer-Verlag.

[Mairson 1991] H. Mairson. **Outline of a proof theory of parametricity**. *Proc. 5th International Symposium on Functional Programming Languages and Computer Architecture*. Springer-Verlag.

[Meyer, *et al.* 1990] A.R. Meyer, J.C. Mitchell, E. Moggi, and R. Statman, **Empty types in polymorphic lambda calculus (preliminary report)**. In *Logical foundations of functional programming,* G. Huet, ed. Addison-Wesley. 273-314.

[Milner, Tofte, Harper 1989] R. Milner, M. Tofte, and R. Harper, **The definition of Standard ML**. MIT Press.

[Mitchell, Scedrov 1992] J.C. Mitchell and A. Scedrov, **Notes on sconing and relators**. Manuscript.

[Plotkin, Abadi 1993] G.D. Plotkin and M. Abadi. **A logic for parametric polymorphism**. *Proc. International Conference on Typed Lambda Calculi and Applications*. Springer-Verlag.

[Plotkin, Abadi, Cardelli 1993] G.D. Plotkin, M. Abadi, and L. Cardelli, **Subtyping and parametricity**. Manuscript.

[Reynolds 1983] J.C. Reynolds, **Types, abstraction, and parametric polymorphism**. In *Information Processing,* R.E.A. Mason, ed. North Holland. 513-523.

[Strachey 1967] C. Strachey, **Fundamental concepts in programming languages**. Lecture notes for the International Summer School in Computer Programming, Copenhagen, August 1967.

[Wadler 1989] P. Wadler. **Theorems for free**! *Proc. 4th International Symposium on Functional Programming Languages and Computer Architecture*. Springer-Verlag.

[Wadler 1991] P. Wadler, **Recursive types for free**! Manuscript.

Appendix

A.1 System F

Environments

(Env $\emptyset$)
$$\frac{}{\vdash \emptyset}$$

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

(Env x)
$$\frac{E \vdash A \qquad x \notin dom(E)}{\vdash E,\ x : A}$$

Types

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

(Type Arrow)
$$\frac{E \vdash A \qquad E \vdash B}{E \vdash A \rightarrow B}$$

(Type Forall)
$$\frac{E,\ X \vdash B}{E \vdash \forall(X)B}$$

Values

(Val x)
$$\frac{\vdash E',\ x : A,\ E''}{E',\ x : A,\ E'' \vdash x : A}$$

(Val Fun)
$$\frac{E,\ x : A \vdash b : B}{E \vdash \lambda(x{:}A)b : A \rightarrow B}$$

(Val Fun2)
$$\frac{E,\ X \vdash b : B}{E \vdash \lambda(X)b : \forall(X)B}$$

(Val Appl)
$$\frac{E \vdash b : A \rightarrow B \qquad E \vdash a : A}{E \vdash b(a) : B}$$

(Val Appl2)
$$\frac{E \vdash b : \forall(X)B \qquad E \vdash C}{E \vdash b(C) : B\{X \leftarrow C\}}$$

Value equality

(Val Eq Symm)
$$\frac{E \vdash a = b : A}{E \vdash b = a : A}$$

(Val Eq Trans)
$$\frac{E \vdash a = b : A \qquad E \vdash b = c : A}{E \vdash a = c : A}$$

(Val Eq x)
$$\frac{\vdash E',\ x : A,\ E''}{E',\ x : A,\ E'' \vdash x = x : A}$$

(Val Eq Fun)
$$\frac{E,\ x : A \vdash b = b' : B}{E \vdash \lambda(x{:}A)b = \lambda(x{:}A)b' : A \rightarrow B}$$

(Val Eq Appl)
$$\frac{E \vdash b = b' : A \rightarrow B \qquad E \vdash a = a' : A}{E \vdash b(a) = b'(a') : B}$$

(Val Eq Fun2)
$$\frac{E,\ X \vdash b = b' : B}{E \vdash \lambda(X)b = \lambda(X)b' : \forall(X)B}$$

(Val Eq Appl2)
$$\frac{E \vdash b = b' : \forall(X)B \qquad E \vdash C}{E \vdash b(C) = b'(C) : B\{X \leftarrow C\}}$$

(Val Beta)
$$\frac{E,\ x : A \vdash b = b' : B \qquad E \vdash a = a' : A}{E \vdash (\lambda(x{:}A)b)(a) = b'\{x \leftarrow a'\} : B}$$

(Val Beta2)
$$\frac{E,\ X \vdash b = b' : B \qquad E \vdash A}{E \vdash (\lambda(X)b)(A) = b'\{X \leftarrow A\} : B\{X \leftarrow A\}}$$

(Val Eta)
$$\frac{E \vdash b = b' : A \rightarrow B \qquad x \notin dom(E)}{E \vdash \lambda(x{:}A)b(x) = b' : A \rightarrow B}$$

(Val Eta2)
$$\frac{E \vdash b = b' : \forall(X)B \qquad X \notin dom(E)}{E \vdash \lambda(X)b(X) = b' : \forall(X)B}$$

A.2 System R ^{1}

Notation

- We use the following metavariables: x,y,z range over value variables; X,Y,Z range over type variables; W ranges over relation variables; a,b,c,d range over value terms; A,B,C,D range over type terms; R, S, T, U range over relation terms; E ranges over environments.
- We use the abbreviations:

$$E \vdash A \;\triangleq\; E \vdash \begin{array}{c}A\\ A^*\\ A\end{array} \qquad\qquad E \vdash a : A \;\triangleq\; E \vdash \begin{array}{c}a : A\\ A^*\\ a : A\end{array}$$

$$\vdash E,\ X,\ E' \;\triangleq\; \vdash E,\ \begin{array}{c}X\\ \mathit{X}\\ X'\end{array},\ E' \qquad\qquad E,\ X,\ E' \vdash J \;\triangleq\; E,\ \begin{array}{c}X\\ \mathit{X}\\ X'\end{array},\ E' \vdash J \quad \text{where } \mathit{X}, X' \text{ are fresh}$$

$$\vdash E,\ x : A,\ E' \;\triangleq\; \vdash E,\ \begin{array}{c}x : A\\ A^*\\ x' : A\end{array},\ E' \qquad\qquad E,\ x : A,\ E' \vdash J \;\triangleq\; E,\ \begin{array}{c}x : A\\ A^*\\ x' : A\end{array},\ E' \vdash J \quad \text{where } x' \text{ is fresh}$$

Environments

(Env $\emptyset$)

$$\frac{}{\vdash \emptyset}$$

(Env XWY)

$$\frac{\vdash E \qquad \begin{array}{l}X, W, Y \notin dom(E)\\ X, W, Y \text{ distinct}\end{array}}{\vdash E,\ \begin{array}{c}X\\ W\\ Y\end{array}}$$

(Env xR y)

$$\frac{E \vdash \begin{array}{c}A\\ R\\ B\end{array} \qquad \begin{array}{l}x, y \notin dom(E)\\ x, y \text{ distinct}\end{array}}{\vdash E,\ \begin{array}{c}x : A\\ R\\ y : B\end{array}}$$

Related types

(Rel W)

$$\frac{\vdash E',\ \begin{array}{c}X\\ W\\ Y\end{array},\ E''}{E',\ \begin{array}{c}X\\ W\\ Y\end{array},\ E'' \vdash \begin{array}{c}X\\ W\\ Y\end{array}}$$

(Rel WX)

$$\frac{\vdash E',\ \begin{array}{c}X\\ W\\ Y\end{array},\ E''}{E',\ \begin{array}{c}X\\ W\\ Y\end{array},\ E'' \vdash X}$$

(Rel WY)

$$\frac{\vdash E',\ \begin{array}{c}X\\ W\\ Y\end{array},\ E''}{E',\ \begin{array}{c}X\\ W\\ Y\end{array},\ E'' \vdash Y}$$

(Rel Arrow)

$$\frac{E \vdash \begin{array}{c}A\\ R\\ A'\end{array} \qquad E \vdash \begin{array}{c}B\\ S\\ B'\end{array}}{E \vdash \begin{array}{c}A \to B\\ R \to S\\ A' \to B'\end{array}}$$

(Rel Forall)

$$\frac{E,\ \begin{array}{c}X\\ W\\ X'\end{array} \vdash \begin{array}{c}B\\ S\\ B'\end{array} \qquad \begin{array}{l}X \notin B', S\\ X' \notin B, S\end{array}}{E \vdash \begin{array}{c}\forall(X)B\\ \forall(W)S\\ \forall(X')B'\end{array}}$$

(Rel FRel)

$$\frac{E \vdash A \to B \qquad E \vdash b : A \to B}{E \vdash \begin{array}{c}A\\ \langle b\rangle\\ B\end{array}}$$

Related values

(Rel Val Symm)

$$\frac{E \vdash \begin{array}{c} a : A \\ A^* \\ b : A \end{array}}{E \vdash \begin{array}{c} b : A \\ A^* \\ a : A \end{array}}$$

(Rel Val Saturation Lft)

$$\frac{E \vdash \begin{array}{c} a : A \\ A^* \\ b : A \end{array} \qquad E \vdash \begin{array}{c} b : A \\ \mathit{R} \\ c : B \end{array}}{E \vdash \begin{array}{c} a : A \\ \mathit{R} \\ c : B \end{array}}$$

(Rel Val Saturation Rht)

$$\frac{E \vdash \begin{array}{c} b : A \\ \mathit{R} \\ c : B \end{array} \qquad E \vdash \begin{array}{c} c : B \\ B^* \\ d : B \end{array}}{E \vdash \begin{array}{c} b : A \\ \mathit{R} \\ d : B \end{array}}$$

(Rel Val x*R* y)

$$\frac{\vdash E', \begin{array}{c} x : A \\ \mathit{R} \\ y : B \end{array}, E''}{E', \begin{array}{c} x : A \\ \mathit{R} \\ y : B \end{array}, E'' \vdash \begin{array}{c} x : A \\ \mathit{R} \\ y : B \end{array}}$$

(Rel Val *R* x)

$$\frac{\vdash E', \begin{array}{c} x : A \\ \mathit{R} \\ y : B \end{array}, E''}{E', \begin{array}{c} x : A \\ \mathit{R} \\ y : B \end{array}, E'' \vdash x : A}$$

(Rel Val *R* y)

$$\frac{\vdash E', \begin{array}{c} x : A \\ \mathit{R} \\ y : B \end{array}, E''}{E', \begin{array}{c} x : A \\ \mathit{R} \\ y : B \end{array}, E'' \vdash y : B}$$

(Rel Val Fun)

$$\frac{E, \begin{array}{c} x : A \\ \mathit{R} \\ x' : A' \end{array} \vdash \begin{array}{c} b : B \\ \mathit{S} \\ b' : B' \end{array} \qquad E \vdash \begin{array}{c} B \\ \mathit{S} \\ B' \end{array} \qquad \begin{array}{c} x \notin b' \\ x' \notin b \end{array}}{E \vdash \begin{array}{c} \lambda(x : A)b : A \to B \\ \mathit{R} \to \mathit{S} \\ \lambda(x' : A')b' : A' \to B' \end{array}}$$

(Rel Val Appl)

$$\frac{E \vdash \begin{array}{c} b : A \to B \\ \mathit{R} \to \mathit{S} \\ b' : A' \to B' \end{array} \qquad E \vdash \begin{array}{c} a : A \\ \mathit{R} \\ a' : A' \end{array}}{E \vdash \begin{array}{c} b(a) : B \\ \mathit{S} \\ b'(a') : B' \end{array}}$$

(Rel Val Fun2)

$$\frac{E, \begin{array}{c} X \\ \mathit{W} \\ X' \end{array} \vdash \begin{array}{c} b : B \\ \mathit{S} \\ b' : B' \end{array} \qquad \begin{array}{c} X \notin b', B', \mathit{S} \\ X' \notin b, B, \mathit{S} \end{array}}{E \vdash \begin{array}{c} \lambda(X)b : \forall(X)B \\ \forall(\mathit{W})\mathit{S} \\ \lambda(X')b' : \forall(X')B' \end{array}}$$

(Rel Val Appl2)

$$\frac{E \vdash \begin{array}{c} b : \forall(X)B \\ \forall(\mathit{W})\mathit{S} \\ b' : \forall(X')B' \end{array} \qquad E \vdash \begin{array}{c} C \\ \mathit{T} \\ C' \end{array}}{E \vdash \begin{array}{c} b(C) : B\{X \leftarrow C\} \\ \mathit{S}\{\mathit{W} \leftarrow \mathit{T}\} \\ b'(C') : B'\{X' \leftarrow C'\} \end{array}}$$

(Rel Val FRel Intro)

$$\frac{E \vdash b : A \to B \qquad E \vdash a : A}{E \vdash \begin{array}{c} a : A \\ \langle b \rangle \\ b(a) : B \end{array}}$$

(Rel Val FRel Elim)

$$\frac{E \vdash \begin{array}{c} a : A \\ \langle b \rangle \\ c : B \end{array} \qquad E \vdash b : A \to B}{E \vdash \begin{array}{c} b(a) : B \\ B^* \\ c : B \end{array}}$$

(Rel Val Beta)

$$\frac{E, x : A \vdash b : B \qquad E \vdash a : A}{E \vdash \begin{array}{c} (\lambda(x : A)b)(a) : B \\ B^* \\ b\{x \leftarrow a\} : B \end{array}}$$

(Rel Val Beta2)

$$\frac{E, X \vdash b : B \qquad E \vdash A}{E \vdash \begin{array}{c} (\lambda(X)b)(A) : B\{X \leftarrow A\} \\ B^*\{X \leftarrow A^*\} \\ b\{X \leftarrow A\} : B\{X \leftarrow A\} \end{array}}$$

(Rel Val Eta)

$$\frac{E \vdash b : A \to B \qquad x \notin dom(E)}{E \vdash \begin{array}{c} \lambda(x:A)b(x) : A \to B \\ (A \to B)* \\ b : A \to B \end{array}}$$

(Rel Val Eta2)

$$\frac{E \vdash b : \forall(X)B \qquad X \notin dom(E)}{E \vdash \begin{array}{c} \lambda(X)b(X) : \forall(X)B \\ (\forall(X)B)* \\ b : \forall(X)B \end{array}}$$

System R^0

System R^0 is obtained by removing functional relations and the corresponding rules (Rel FRel), (Rel Val FRel Intro), and (Rel Val FRel Elim) from system R^1.

A.3 Hasegawa’s Paradox

Consider the system obtained from R^1 by allowing quantification over type variables in relations, and by adding a notion of relation equality, with the rules:

(Rel Eq Forall XW)

$$\frac{E, \begin{array}{c} X \\ W \\ X' \end{array} \vdash \begin{array}{c} B \\ S = S' \\ B' \end{array} \qquad \begin{array}{l} X \notin B', S, S' \\ X' \notin B, S, S' \\ Z \notin dom(E) \end{array}}{E \vdash \begin{array}{c} \forall(X)B \\ \forall(Z)S\{W \leftarrow Z\} = \forall(W)S' \\ \forall(X')B' \end{array}}$$

(Rel Val Rel Eq)

$$\frac{E \vdash \begin{array}{c} a : A \\ R \\ b : B \end{array} \qquad E \vdash \begin{array}{c} A \\ R = S \\ B \end{array}}{E \vdash \begin{array}{c} a : A \\ S \\ b : B \end{array}}$$

and further rules for formation of relations, introduction and elimination of quantifiers, and congruence rules. This is the system presented in [Abadi, Cardelli, Curien 1993]. Hasegawa has shown that this system is inconsistent, as follows.

Consider the environment:

$$E = X, \begin{array}{c} y : Bot \to X \\ \langle f \rangle \to X \\ y' : Bool \to X \end{array}, \begin{array}{c} x : X \to Bot \\ X \to \langle f \rangle \\ x' : X \to Bool \end{array}$$

where $Bot = \forall(X)X$ and $f = \lambda(z:Bot)z(Bool) : Bot \to Bool$. By (Rel Val R x) and (Rel Val R y), we have $E \vdash y' : Bool \to X$ and $E \vdash x : X \to Bot$, hence $E \vdash x(y'(true)) : Bot$. By the initiality of Bot (section 3.8), we have:

$$z : Bot \vdash \begin{array}{c} true : Bool \\ Bool* \\ false : Bool \end{array} \qquad \text{so we obtain:} \qquad E \vdash \begin{array}{c} true : Bool \\ Bool* \\ false : Bool \end{array}$$

Hence, abstracting, we obtain:

$$\vdash \begin{array}{c} \lambda(X)\lambda(y:Bot \to X)\lambda(x:X \to Bot)true : \forall(X)(Bot \to X) \to (X \to Bot) \to Bool \\ \forall(X)(\langle f \rangle \to X) \to (X \to \langle f \rangle) \to Bool* \\ \lambda(X)\lambda(y':Bool \to X)\lambda(x':X \to Bool)false : \forall(X)(Bool \to X) \to (X \to Bool) \to Bool \end{array}$$

Now (Rel Eq Forall XW) and (Rel Val Rel Eq) yield:

$$\vdash \begin{array}{c} \lambda(X)\lambda(y:Bot \to X)\lambda(x:X \to Bot)true : \forall(X)(Bot \to X) \to (X \to Bot) \to Bool \\ \forall(W)(\langle f \rangle \to W) \to (W \to \langle f \rangle) \to Bool* \\ \lambda(X)\lambda(y':Bool \to X)\lambda(x':X \to Bool)false : \forall(X)(Bool \to X) \to (X \to Bool) \to Bool \end{array}$$

On the other hand, we have:

$$\vdash \begin{array}{c} Bot \\ \langle f \rangle \\ Bool \end{array} \qquad \vdash \begin{array}{c} \lambda(z:Bot)z : Bot \to Bot \\ \langle f \rangle \to \langle f \rangle \\ \lambda(z':Bool)z' : Bool \to Bool \end{array}$$

Finally, by (Rel Val Appl2) and (Rel Val Appl), we reach the inconsistency:

$$\vdash \begin{array}{c} true : Bool \\ Bool* \\ false : Bool \end{array}$$

We blame this inconsistency on (Rel Eq Forall XW), which equates type quantifiers and relation quantifiers in arbitrary relation expressions. The rules in appendix A.2 keep the two quantifiers separate.