A Game Semantics for Generic Polymorphism

Samson Abramsky[1⋆] and Radha Jagadeesan[2⋆⋆]

[1] Oxford University Computing Laboratory
samson@comlab.ox.ac.uk
[2] DePaul University
rjagadeesan@cs.depaul.edu

Abstract. Genericity is the idea that the same program can work at many different data types. Longo, Milsted and Soloviev proposed to capture the inability of generic programs to probe the structure of their instances by the following equational principle: if two generic programs, viewed as terms of type $\forall X.\, A[X]$, are equal at any given instance $A[T]$, then they are equal at all instances. They proved that this rule is admissible in a certain extension of System F, but finding a semantically motivated model satisfying this principle remained an open problem.
In the present paper, we construct a categorical model of polymorphism, based on game semantics, which contains a large collection of generic types. This model builds on two novel constructions:

- A direct interpretation of variable types as games, with a natural notion of substitution of games. This allows moves in games $A[T]$ to be decomposed into the generic part from A, and the part pertaining to the instance T. This leads to a simple and natural notion of generic strategy.
- A "relative polymorphic product" $\Pi_i(A, B)$ which expresses quantification over the type variable X_i in the variable type A with respect to a "universe" which is explicitly given as an additional parameter B. We then solve a recursive equation involving this relative product to obtain a universe in a suitably "absolute" sense.

Full Completeness for ML types (universal closures of quantifier-free types) can be proved for this model.

1 Introduction

We begin with an illuminating quotation from Gérard Berry [9]:

> Although it is not always made explicit, the *Write Things Once* or WTO principle is clearly the basis for loops, procedures, higher-order functions, object-oriented programming and inheritance, concurrency *vs.* choice between interleavings, etc.

⋆ Samson Abramsky was supported in part by UK EPSRC.
⋆⋆ Radha Jagadeesan was supported in part by NSF CCR-020244901.

A.D. Gordon (Ed.): FOSSACS 2003, LNCS 2620, pp. 1–22, 2003.

In short, much of the search for high-level structure in programming can be seen as the search for concepts which allow commonality to be expressed. An important facet of this quest concerns *genericity*: the idea that the same program can work at many different data types.

For illustration, consider the abstraction step involved in passing from list-processing programs which work on data types $\mathsf{List}[T]$ for specific types T, to programs which work generically on $\mathsf{List}[X]$. Since lists can be so clearly visualized, it is easy to see what this should mean (see Figure 1). A generic program cannot probe the internal structure of the list elements. Thus e.g. list concatenation and reversal are generic, while summing a list is not. However, when we go beyond lists and other concrete data structures, to higher-order types and beyond, what genericity or type-independence should mean becomes much less clear.

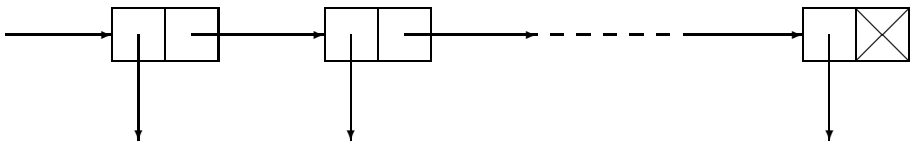

Fig. 1. 'Generic' list structure

One very influential proposal for a general understanding of the *uniformity* which generic programs should exhibit with respect to the type instances has been John Reynolds' notion of *relational parametricity* [21], which requires that relations between instances be preserved in a suitable sense by generic programs. This has led to numerous further developments, e.g. [17,1,19].

Relational parametricity is a beautiful and important notion. However, in our view it is not the whole story. In particular:

- It is a "pointwise" notion, which gets at genericity indirectly, via a notion of uniformity applied to the family of instantiations of the program, rather than directly capturing the idea of a program written at the generic level, which necessarily cannot probe the structure of an instance.
- It is closely linked to strong extensionality principles, as shown e.g. in [1,19], whereas the intuition of generic programs not probing the structure of instances is *prima facie* an intensional notion—a constraint on the behaviour of processes.

An interestingly different analysis of genericity with different formal consequences was proposed by Giuseppe Longo, Kathleen Milsted and Sergei Soloviev [15,16]. Their idea was to capture the inability of generic programs to probe the structure of their instances by the following equational principle: if two generic programs,

viewed as terms t, u of type $A[X]$, are equal at *any* given instance T, then they are equal at *all* instances:

$$\exists T.\, t\{T\} = u\{T\} : A[T] \implies \forall U.\, t\{U\} = u\{U\} : A[U].$$

This principle can be stated even more strongly when second-order polymorphic quantification over type variables is used. For $t, u : \forall X.\, A$:

$$\frac{t\{T\} = u\{T\} : A[T]}{t = u : \forall X.\, A}.$$

We call this the *Genericity Rule*. In one of the most striking syntactic results obtained for System F (*i.e.* the polymorphic second-order λ-calculus [10,20]), Longo, Milsted and Soloviev proved in [15] that the Genericity Rule is admissible in the system obtained by extending System F with the following axiom scheme:

$$(C) \qquad t\{B\} = t\{C\} : A \qquad (t : \forall X.\, A,\ X \notin \mathrm{FV}(A)).$$

While many of the known semantic models of System F satisfy axiom (C), *there is no known naturally occurring model which satisfies the Genericity principle* (*i.e.* in which the rule of Genericity is valid). In fact, in the strong form given above, the Genericity rule is actually *incompatible* with well-pointedness and parametricity, as observed by Longo. Thus if we take the standard polymorphic terms representing the Boolean values

$$\Lambda X.\, \lambda x{:}X.\, \lambda y{:}X.\, x, \quad \Lambda X.\, \lambda x{:}X.\, \lambda y{:}X.\, y \quad : \quad \forall X.\, X \to X \to X$$

then if the type $\forall X.\, X \to X$ has only one inhabitant — as will be the case in a parametric model — then by well-pointedness the Boolean values will be equated at this instance, while they cannot be equated in general on pain of inconsistency.

However, we can state a more refined version. Say that a type T is *a generic instance* if for all types $A[X]$:

$$t\{T\} = u\{T\} : A[T] \implies t = u : \forall X.\, A.$$

This leads to the following problem posed by Longo in [16], and still, to the best of our knowledge, open:

> Open Problem 2. Construct, at least, some (categorical) models that contain a collection of "generic" types. . . . If our intuition about constructivity is correct, infinite objects in categories of (effective) sets should satisfy this property.

In the present paper, we present a solution to this problem by constructing a categorical model of polymorphism which contains a large collection of generic types. The model is based on game semantics; more precisely, it extends the "AJM games" of [5] to provide a model for generic polymorphism. Moreover, Longo's intuition as expressed above is confirmed in the following sense: our main

sufficient condition for games (as denotations of types) to be generic instances is that they have plays of arbitrary length. This can be seen as an intensional version of Longo's intuition about infinite objects.

In addition to providing a solution to this problem, the present paper also makes the following contributions.

- We interpret variable types in a simple and direct way, with a natural notion of *substitution of games into variable games*. The crucial aspect of this idea is that it allows moves in games $A[T]$ to be decomposed into the generic part from A, and the part pertaining to the instance T. This in turn allows the evident content of genericity in the case of concrete data structures such as lists to be carried over to arbitrary higher-order and polymorphic types. In particular, we obtain a simple and natural notion of *generic strategy*. This extends the notion of history-free strategy from [5], which is determined by a function on moves, to that of a generic strategy, which is determined by a function on *the generic part of the move only*, and simply acts as the identity on the part pertaining to the instance. This captures the intuitive idea of a generic program, existing "in advance" of its instances, in a rather direct way.
- We solve the size problem inherent in modelling System F in a somewhat novel way. We define a "relative polymorphic product" $\Pi_i(A, B)$ which expresses quantification over the type variable X_i in the variable type A with respect to a "universe" which is explicitly given as an additional parameter B. We then solve a recursive equation involving this relative product to obtain a universe in a suitably "absolute" sense: a game $\mathcal{U}$ with the requisite closure properties to provide a model for System F.

It is also possible to prove a Full Completeness theorem for the ML types (*i.e.* the universal closures of quantifier-free types). For this, further technical details, and proofs of the results, we refer to the full version of this paper [4].

2 Background

2.1 Syntax of System F

We briefly review the syntax of System F. For further background information we refer to [11].

Types (Formulas)

$$A \quad ;;= \quad X \mid A \to B \mid \forall X.\, A$$

Typing Judgements Terms in context have the form

$$x_1 : A_1, \ldots, x_k : A_k \vdash t : A$$

Assumption

$$\frac{}{\Gamma, x : t \vdash x : T}$$

Implication

$$\frac{\Gamma, x : U \vdash t : T}{\Gamma \vdash \lambda x{:}U.\, t : U \to T}\ (\to - I) \qquad \frac{\Gamma \vdash t : U \to T \qquad \Gamma \vdash u : U}{\Gamma \vdash tu : T}\ (\to - E)$$

Second-order Quantification

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \Lambda X.\, t : \forall X.\, A}\ (\forall - I) \qquad \frac{\Gamma \vdash t : \forall X.\, A}{\Gamma \vdash t\{B\} : A[B/X]}\ (\forall - E)$$

The $(\forall - I)$ rule is subject to the usual eigenvariable condition, that X does not occur free in Γ.
The following isomorphism is definable in System F:

$$\forall X.\, A \to B \ \cong\ A \to \forall X.\, B \qquad (X \notin \mathrm{FV}(A)).$$

This allows us to use the following normal form for types:

$$\forall \boldsymbol{X}.\, T_1 \to \cdots \to T_k \to X \qquad (k \geq 0)$$

where each T_i is inductively of the same form.

2.2 Notation

We write ω for the set of natural numbers.

We shall use vector notation, writing $\boldsymbol{A}$ for a list $A_1, \ldots, A_k$.

If X is a set, X^* is the set of finite sequences (words, strings) over X. We use s, t, u, v to denote sequences, and a, b, c, d, m, n to denote elements of these sequences. Concatenation of sequences is indicated by juxtaposition, and we don't distinguish notationally between an element and the corresponding unit sequence. Thus as denotes the sequence with first element a and tail s. However, we will sometimes write $a \cdot s$ or $s \cdot a$ to give the name a to the first or last element of a sequence.

If $f : X \longrightarrow Y$ then $f^* : X^* \longrightarrow Y^*$ is the unique monoid homomorphism extending f. We write $|s|$ for the length of a finite sequence, and s_i for the ith element of s, $1 \leq i \leq |s|$. We write $\mathsf{numoccs}(a, s)$ for the number of occurrences of a in the sequence s.

We write $s \sqsubseteq t$ if s is a prefix of t, *i.e.* $t = su$ for some u. We write $s \sqsubseteq^{\mathrm{even}} t$ if s is an even-length prefix of t. $\mathsf{Pref}(S)$ is the set of prefixes of elements of $S \subseteq X^*$. S is *prefix-closed* if $S = \mathsf{Pref}(S)$.

3 Variable Games and Substitution

3.1 A Universe of Moves

We fix an algebraic signature consisting of the following set of *unary* operations:

$$\mathtt{p},\ \mathtt{q},\ \{\mathtt{l}_i \mid i \in \omega\},\ \mathtt{r}.$$

We take $\mathcal{M}$ to be the algebra over this signature freely generated by ω. Explicitly, $\mathcal{M}$ has the following "concrete syntax":

$$m \quad ::= \quad i\ (i \in \omega) \quad | \quad \mathtt{p}(m) \quad | \quad \mathtt{q}(m) \quad | \quad \mathtt{l}_i(m)\ \ (i \in \omega) \quad | \quad \mathtt{r}(m).$$

For any algebra $(A, \mathtt{p}^A, \mathtt{q}^A, \{\mathtt{l}_i^A \mid i \in \omega\}, \mathtt{r}^A)$ and map $f : \omega \longrightarrow A$, there is a unique homomorphism $f^\dagger : \mathcal{M} \longrightarrow A$ extending f, defined by:

$$f^\dagger(i) = f(i), \qquad f^\dagger(\phi(m)) = \phi^A(f^\dagger(m)) \quad (\phi \in \{\mathtt{p}, \mathtt{q}, \mathtt{r}\} \cup \{\mathtt{l}_i \mid i \in \omega\}).$$

We now define a number of maps on $\mathcal{M}$ by this means.

- The *labelling map* $\lambda : \mathcal{M} \longrightarrow \{P, O\}$. The polarity algebra on the carrier $\{P, O\}$ interprets $\mathtt{p}$, $\mathtt{q}$, $\mathtt{r}$ as the identity, and each $\mathtt{l}_i$ as the involution $\bar{(\ )}$, where $\bar{P} = O$, $\bar{O} = P$. The map on the generators is the constant map sending each i to O.
- The map $\rho : \mathcal{M} \longrightarrow \omega$ sends each move to the unique generator occurring in it. All the unary operations are interpreted as the identity, and the map on generators is the identity.
- The substitution map. For each move $m' \in \mathcal{M}$, there is a map

  $$h_{m'} : \mathcal{M} \longrightarrow \mathcal{M}$$

  induced by the constant map on ω which sends each i to m'. We write $m[m']$ for $h_{m'}(m)$.
- An alternative form of substitution is written $m[m'/i]$. This is induced by the map which send i to m', and is the identity on all $j \neq i$.

Proposition 1. *Substitution interacts with λ and ρ as follows.*

1. $\lambda(m[m']) = \begin{cases} \overline{\lambda(m')} \text{ if } \lambda(m) = P \\ \lambda(m') \text{ if } \lambda(m) = O \end{cases}$
2. $\rho(m[m']) = \rho(m')$.

We extend the notions of substitution pointwise to sequences and sets of sequences of moves in the evident fashion.
We say that $m_1, m_2 \in \mathcal{M}$ are *unifiable* if for some $m_3, m_4 \in \mathcal{M}$, $m_1[m_3] = m_2[m_4]$. A set $S \subseteq \mathcal{M}$ is *unambiguous* if whenever $m_1, m_2 \in S$ are unifiable, $m_1 = m_2$.

Given a subset $S \subseteq \mathcal{M}$ and $i \in \omega$, we write

$$S^i = \{m \in S \mid \rho(m) = i\}.$$

We define a notion of projection of a sequence of moves s onto a move m inductively as follows:

$$\begin{array}{lll} \varepsilon \upharpoonright m & = \varepsilon & \\ m[m'] \cdot s \upharpoonright m & = m' \cdot (s \upharpoonright m) & \\ m' \cdot s \upharpoonright m & = s \upharpoonright m, & \forall m''.\, m' \neq m[m'']. \end{array}$$

Dually, given an unambiguous set of moves S, and a sequence of moves s in which every move has the form $m[m']$ for some $m \in S$ (necessarily unique since S is unambiguous), we define a projection $s \upharpoonright S$ inductively as follows:

$$\begin{array}{lll} \varepsilon \upharpoonright S & = \varepsilon & \\ m[m'] \cdot s \upharpoonright S & = m \cdot (s \upharpoonright S) & (m \in S \ \wedge \ \rho(m) > 0) \\ m[m'] \cdot s \upharpoonright S & = m[m'] \cdot (s \upharpoonright S) & (m \in S^0) \end{array}$$

3.2 Variable Games

A *variable game* is a structure

$$A = (\mathcal{O}_A, P_A, \approx_A)$$

where:

- $\mathcal{O}_A \subseteq \mathcal{M}$ is an unambiguous set of moves: the *occurrences* of A. We then define:
  - $\lambda_A = \lambda \upharpoonright \mathcal{O}_A$.
  - $\rho_A = \rho \upharpoonright \mathcal{O}_A$.
  - $M_A = \{m[m'] \mid m \in \mathcal{O}_A^0 \ \wedge \ m' \in \mathcal{M}\} \ \cup \ \bigcup_{j>0} \mathcal{O}_A^j$.
- P_A is a non-empty prefix-closed subset of M_A^* satisfying the following form of *alternation condition*: the odd-numbered moves in a play are *moves by* O, while the even-numbered moves are *by* P. Here we regard the first, third, fifth, ... occurrences of a move m in a sequence as being by $\lambda_A(m)$, while the second, fourth, sixth ... occurrences are by the other player.
- $\approx_A$ is an equivalence relation on P_A such that:

$$\begin{array}{ll} (\mathbf{e1}) & s \approx_A t \implies s \longleftrightarrow t \\ (\mathbf{e2}) & ss' \approx_A tt' \ \wedge \ |s| = |t| \implies s \approx_A t \\ (\mathbf{e3}) & s \approx_A t \ \wedge \ sa \in P_A \implies \exists b.\, sa \approx_A tb. \end{array}$$

  Here $s \longleftrightarrow t$ holds if

$$s = \langle m_1, \ldots, m_k \rangle, \qquad t = \langle m'_1, \ldots, m'_k \rangle$$

  and the correspondence $m_i \longleftrightarrow m'_i$ is bijective and preserves λ_A and ρ_A. We write

$$\pi : s \longleftrightarrow t$$

  to give the name π to the bijective correspondence $m_i \longleftrightarrow m'_i$.

A move $m \in \mathcal{O}_A^i$, $i > 0$, is an *occurrence* of the type variable X_i, while $m \in \mathcal{O}_A^0$ is a *bound occurrence*.

The set of variable games is denoted by $\mathcal{G}(\omega)$. The set of those games A for which the range of ρ_A is included in $\{0, \ldots, k\}$ is denoted by $\mathcal{G}(k)$. Note that if $k \leq l$, then

$$\mathcal{G}(k) \subseteq \mathcal{G}(l) \subseteq \mathcal{G}(\omega).$$

$\mathcal{G}(0)$ is the set of *closed games*.

Comparison with AJM games The above definition of game differs from that in [5] in several minor respects.

1. The notion of bracketing condition, requiring a classification of moves as *questions* or *answers*, has been omitted. This is because we are dealing here with pure type theories, with no notion of "ground data types".
2. The alternation condition has been modified: we still have strict OP-alternation of moves, but now successive occurrences of moves within a sequence are regarded as themselves having alternating polarities. Since in the PCF games in [5] moves in fact only occur once in any play, they do fall within the present formulation. The reason for the revised formulation is that moves in variable games are to be seen as *occurrences* of type variables, which can be expanded into plays at an instance.
3. We have replaced the condition (**e1**) from [5] with a slightly stronger condition, which is in fact satisfied by the games in [5].

3.3 Constructions on Games

Since variable games are essentially just AJM games with some additional structure on moves, the cartesian closed structure on AJM games can be lifted straighforwardly to variable games.

Unit Type The unit type **1** is the empty game.

$$\mathbf{1} \;=\; (\varnothing, \{\varepsilon\}, \{(\varepsilon, \varepsilon)\}).$$

Product The product $A\&B$ is the disjoint union of games.

$$\mathcal{O}_{A\&B} = \{\mathtt{p}(m) \mid m \in \mathcal{O}_A\} \cup \{\mathtt{q}(m) \mid m \in \mathcal{O}_B\}$$

$$P_{A\&B} = \{\mathtt{p}^*(s) \mid s \in P_A\} \cup \{\mathtt{q}^*(t) \mid t \in P_B\}$$

$$\mathtt{p}^*(s) \approx_{A\&B} \mathtt{p}^*(t) \;\equiv\; s \approx_A t \qquad \mathtt{q}^*(s) \approx_{A\&B} \mathtt{q}^*(t) \;\equiv\; s \approx_B t.$$

Function Space The function space $A \Rightarrow B$ is defined as follows.

$$\mathcal{O}_{A\Rightarrow B} \;=\; \{\mathtt{l}_i(m) \mid i \in \omega \;\wedge\; m \in \mathcal{O}_A\} \;\cup\; \{\mathtt{r}(m) \mid m \in \mathcal{O}_B\}.$$

$P_{A\Rightarrow B}$ is defined to be the set of all sequences in $M^*_{A\Rightarrow B}$ satisfying the alternation condition, and such that:

- $\forall i \in \omega.\, s{\restriction}\mathtt{l}_i(1) \in P_A$.
- $s{\restriction}\mathtt{r}(1) \in P_B$.

Let $S = \{\mathtt{l}_i(1) \mid i \in \omega\} \cup \{\mathtt{r}(1)\}$. Note that S is unambiguous. Given a permutation α on ω, we define

$$\breve{\alpha}(\mathtt{l}_i(1)) = \mathtt{l}_{\alpha(i)}(1), \qquad \breve{\alpha}(\mathtt{r}(1)) = \mathtt{r}(1).$$

The equivalence relation $s \approx_{A\Rightarrow B} t$ is defined by the condition

$$\exists \alpha \in S(\omega).\, \breve{\alpha}^*(s{\restriction}S) = t{\restriction}S \;\wedge\; s{\restriction}\mathtt{r}(1) \approx_B t{\restriction}\mathtt{r}(1) \;\wedge\; \forall i \in \omega.\, s{\restriction}\mathtt{l}_i(1) \approx_A t{\restriction}\mathtt{l}_{\alpha(i)}(1)).$$

This is essentially identical to the definition in [5]. The only difference is that we use the revised version of the alternation condition in defining the positions, and that we define $A \Rightarrow B$ directly, rather than via the linear connectives $\multimap$ and !.

3.4 Substitution

Given $A \in \mathcal{G}(k)$, and $\boldsymbol{B} = B_1, \ldots, B_k \in \mathcal{G}(l)$, we define $A[\boldsymbol{B}] \in \mathcal{G}(l)$ as follows.

$$\mathcal{O}_{A[\boldsymbol{B}]} \;=\; \mathcal{O}^0_A \;\cup\; \bigcup_{i=1}^{k} \{m[m'] \mid m \in \mathcal{O}^i_A \;\wedge\; m' \in \mathcal{O}_{A_i}\}.$$

$$P_{A[\boldsymbol{B}]} = \{s \in M^*_{A[\boldsymbol{B}]} \mid s{\restriction}A \in P_A \;\wedge\; \forall i : 1 \le i \le k.\, \forall m \in \mathcal{O}^i_A.\, s{\restriction}m \in P_{B_i}\}$$

$$\begin{aligned} s \approx_{A[\boldsymbol{B}]} t \;\equiv\; & s{\restriction}A \approx_A t{\restriction}A \\ & \wedge \\ & \pi : s{\restriction}A \longleftrightarrow t{\restriction}A \implies \forall i : 1 \le i \le k.\, \forall m \in \mathcal{O}^i_A.\, s{\restriction}m \approx_{B_i} t{\restriction}\pi(m). \end{aligned}$$

Here by convenient abuse of notation we write $s{\restriction}A$ for $s{\restriction}\mathcal{O}_A$.

Note that the above definitions would still make sense if we took $k = \omega$ and/or $l = \omega$, so that, for example, there is a well-defined operation

$$\mathcal{G}(\omega) \times \mathcal{G}(\omega)^\omega \longrightarrow \mathcal{G}(\omega).$$

In practice, the finitary versions will be more useful for our purposes here, as they correspond to the finitary syntax of System F.

3.5 Properties of Substitution

Proposition 2. *If $A \in \mathcal{G}(k)$, $B_1, \ldots, B_k \in \mathcal{G}(l)$, and $C_1, \ldots, C_l \in \mathcal{G}(m)$, then:*

$$A[B_1[\boldsymbol{C}], \ldots, B_k[\boldsymbol{C}]] \;=\; (A[B_1, \ldots, B_k])[\boldsymbol{C}].$$

For each $i > 0$ we define the variable game X_i as follows.

$$\begin{array}{ll} \mathcal{O}_{X_i} & = \{i\} \\ P_{X_i} & = M^*_{X_i} \\ s \approx_{X_i} t & \equiv s = t \end{array}$$

Proposition 3. *1. For all $B_1, \ldots B_k \in \mathcal{G}(\omega)$, $i \leq k$: $X_i[B_1, \ldots B_k] \;=\; B_i$.*
2. For all $A \in \mathcal{G}(k)$: $A[X_1, \ldots, X_k] \;=\; A$.

We can define a useful variant of substitution by:

$$A[B/X_i] \;\;=\;\; A[X_1, \ldots, X_{i-1}, B, X_{i+1}, \ldots, X_k]$$

for $A \in \mathcal{G}(k)$, $1 \leq i \leq k$.

Proposition 4. *The cartesian closed structure commutes with substitution:*

$$\begin{array}{l} 1.\ (A \Rightarrow B)[\boldsymbol{C}] = A[\boldsymbol{C}] \Rightarrow B[\boldsymbol{C}] \\ 2.\ (A \,\&\, B)[\boldsymbol{C}] \;=\; A[\boldsymbol{C}] \,\&\, B[\boldsymbol{C}]. \end{array}$$

Combining Propositions 3 and 4, we obtain:

Proposition 5. *The cartesian closed constructions can be obtained by substitution from their generic forms:*

$$\begin{array}{l} 1.\ A \Rightarrow B = X_1 \Rightarrow X_2[A, B] \\ 2.\ A \,\&\, B \;=\; X_1 \,\&\, X_2[A, B]. \end{array}$$

4 Constructing a Universe for Polymorphism

4.1 The Inclusion Order

We define $A \trianglelefteq B$ by:

- $\mathcal{O}_A \subseteq \mathcal{O}_B$
- $P_A \subseteq P_B$
- $s \approx_A t \iff s \in P_A \;\wedge\; s \approx_B t$

The inclusion order is useful in the following context. Suppose we fix a "big game" $\mathcal{U}$ to serve as a "universe". Define a *sub-game* of $\mathcal{U}$ to be a game of the form

$$A = (\mathcal{O}_{\mathcal{U}}, P_A, \approx_{\mathcal{U}} \cap P_A^2),$$

where $P_A \subseteq P_{\mathcal{U}}$, and

$$s \in P_A \ \wedge \ s \approx_{\mathcal{U}} t \ \implies \ t \in P_A.$$

Thus sub-games of $\mathcal{U}$ are completely determined by their sets of positions. We write $\mathsf{Sub}(\mathcal{U})$ for the set of sub-games of $\mathcal{U}$. Note that, for $A, B \in \mathsf{Sub}(\mathcal{U})$:

$$A \trianglelefteq B \iff P_A \subseteq P_B.$$

Proposition 6. *1. $\mathsf{Sub}(\mathcal{U})$ is a complete lattice, with meets and joins given by intersections and unions respectively.*
2. If $S \subseteq P_{\mathcal{U}}$, then the least sub-game $A \in \mathsf{Sub}(\mathcal{U})$ such that $S \subseteq P_A$ is defined by

$$P_A = \{u \mid \exists s \in S.\, \exists t.\, t \sqsubseteq s \ \wedge \ u \approx_{\mathcal{U}} t\}.$$

It is straightforward to verify that function space and product are monotonic with respect to the inclusion order. This leads to the following point, which will be important for our model construction.

Proposition 7. *Suppose that $\mathcal{U}$ is such that*

$$\mathcal{U} \Rightarrow \mathcal{U} \trianglelefteq \mathcal{U}, \qquad \mathcal{U} \,\&\, \mathcal{U} \trianglelefteq \mathcal{U}, \qquad \mathbf{1} \trianglelefteq \mathcal{U}.$$

Then $\mathsf{Sub}(\mathcal{U})$ is closed under these constructions.

Adjoints of substitution Let A be a variable game, and $s \in P_{A[\mathcal{U}/X_i]}$. We can use the substitution structure to compute the *least* instance B (with respect to $\trianglelefteq$) such that $s \in P_{A[B/X_i]}$. We define

$$A_i^*(s) = \{t \mid \exists u.\, \exists m \in \mathcal{O}_A^i.\, t \approx u \ \wedge \ u \sqsubseteq s\restriction m\}$$

Proposition 8. *With notation as in the preceding paragraph, let $B = A_i^*(s)$.*

1. $s \in P_{A[B/X_i]}$.
2. $s \in P_{A[C/X_i]} \implies B \trianglelefteq C$.

4.2 The Relative Polymorphic Product

Given $A, B \in \mathcal{G}(\omega)$ and $i > 0$, we define the relative polymorphic product $\Pi_i(A, B)$ (the "second-order quantification over X_i in the variable type A relative to the universe B") as follows.

$$\mathcal{O}_{\Pi_i(A,B)} = \mathcal{O}_A[0/i] = \{m[0/i] \mid m \in \mathcal{O}_A\}.$$

$$P_{\Pi_i(A,B)} = \{s \in P_{A[B/X_i]} \mid \forall t \cdot a \sqsubseteq^{\text{even}} s.\ A_i^*(t \cdot a) = A_i^*(t)\}$$

$$s \approx_{\Pi_i(A,B)} t \iff s \approx_{A[B/X_i]} t.$$

To understand the definition of $P_{\Pi_i(A,B)}$, it is helpful to consider the following alternative, inductive definition (*cf.* [2]):

$$\begin{aligned} P_{\Pi_i(A,B)} = \quad & \{\epsilon\} \\ & \cup \{sa \mid s \in P^{\mathrm{even}}_{\Pi_i(A,B)} \ \wedge \ \exists C \in \mathsf{Sub}(B).\, sa \in P_{A[C]}\} \\ & \cup \{sab \mid sa \in P^{\mathrm{odd}}_{\Pi_i(A,B)} \ \wedge \ \forall C \in \mathsf{Sub}(B).\, sa \in P_{A[C]} \Rightarrow sab \in P_{A[C]}\} \end{aligned}$$

The first clause in the definition of $P_{\Pi(F)}$ is the basis of the induction. The second clause refers to positions in which it is Opponent's turn to move. It says that Opponent may play in any way which is valid in *some* instance. The final clause refers to positions in which it is Player's turn to move. It says that Player can only move in a fashion which is valid in *every* possible instance. The equivalence of this definition to the one given above follows easily from Proposition 8.

Intuitively, this definition says that initially, nothing is known about which instance we are playing in. Opponent progressively reveals the "game board" ; at each stage, Player is constrained to play within the instance *thus far revealed* by Opponent.

The advantage of the definition we have given above is that it avoids quantification over subgames of B in favour of purely local conditions on the plays.

Proposition 9. *The relative polymorphic product commutes with substitution.*

1. $\Pi_i(A,B)[C/X_i] = \Pi_i(A,B)$.
2. *If* $A \in \mathcal{G}(k+1)$ *and* $C_1, \ldots, C_k \in \mathcal{G}(n)$, *then:*

$$\Pi_{k+1}(A,B)[\boldsymbol{C}] \quad = \quad \Pi_{n+1}(A[\boldsymbol{C}, X_{n+1}], B).$$

4.3 A Domain Equation for System F

We define a variable game $\mathcal{U} \in \mathcal{G}(\omega)$ of System F types by the following recursive equation:

$$\mathcal{U} \quad = \quad \&_{i>0} X_i \ \& \ \mathbf{1} \ \& \ (\mathcal{U} \,\&\, \mathcal{U}) \ \& \ (\mathcal{U} \Rightarrow \mathcal{U}) \ \& \ \&_{i>0} \Pi_i(\mathcal{U}, \mathcal{U}).$$

Explicitly, $\mathcal{U}$ is being defined as the least fixed point of a function $F : \mathcal{G}(\omega) \longrightarrow \mathcal{G}(\omega)$. The theory developed in [8] can be used to guarantee the existence of this least fixpoint.

We can then define second-order quantification by:

$$\forall X_i.\, A \ \triangleq \ \Pi_i(A, \mathcal{U}).$$

Although it is not literally the case that

$$X_i \trianglelefteq \mathcal{U}, \qquad \mathcal{U} \Rightarrow \mathcal{U} \trianglelefteq \mathcal{U}, \qquad \text{etc.}$$

for trivial reasons of how disjoint union is defined, with a little adjustment of definitions we can arrange things so that we indeed have

- $X_i \trianglelefteq \mathcal{U}$
- $\mathbf{1} \trianglelefteq \mathcal{U}$
- $A, B \trianglelefteq \mathcal{U} \Longrightarrow \quad A \,\&\, B \trianglelefteq \mathcal{U} \,\&\, \mathcal{U} \trianglelefteq \mathcal{U}$
- $A, B \trianglelefteq \mathcal{U} \Longrightarrow \quad A \Rightarrow B \trianglelefteq \mathcal{U} \Rightarrow \mathcal{U} \trianglelefteq \mathcal{U}$
- $A \trianglelefteq \mathcal{U} \quad \Longrightarrow \forall X_i.\, A = \Pi_i(A, \mathcal{U}) \trianglelefteq \Pi_i(\mathcal{U}, \mathcal{U}) \trianglelefteq \mathcal{U}.$

Thus we get a direct inductive definition of the types of System F as sub-games of $\mathcal{U}$.

Moreover, if A and B are (the variable games corresponding to) System F types, then a simple induction on the structure of A using Propositions 3, 4 and 9 shows that

$$A[B/X_i] \trianglelefteq \mathcal{U},$$

and similarly for simultaneous substitution.

5 Strategies

Fix a variable game A. Let

$$g : \mathcal{O}_A \longrightarrow\!\!\!\!\!\cdot\ \ \mathcal{O}_A$$

be a partial function. We can extend g to a partial function

$$\hat{g} : M_{A[\boldsymbol{\mathcal{U}}]} \longrightarrow\!\!\!\!\!\cdot\ \ M_{A[\boldsymbol{\mathcal{U}}]}$$

by

$$\hat{g}(m[m']) \ = \ \begin{cases} g(m)[m'], \ g(m) \text{ defined} \\ \text{undefined otherwise} \end{cases}$$

Now we can define a set of plays $\sigma_g \subseteq M^*_{A[\boldsymbol{\mathcal{U}}]}$ inductively as follows:

$$\sigma_g \ = \ \{\varepsilon\} \ \cup \ \{sab \mid s \in \sigma_g \ \wedge \ sa \in P_{A[\boldsymbol{\mathcal{U}}]} \ \wedge \ \hat{g}(a) = b\}.$$

For all $\boldsymbol{B} \trianglelefteq \boldsymbol{\mathcal{U}}$, we can define the restriction of σ_g to $\boldsymbol{B}$ by:

$$\sigma_{\boldsymbol{B}} = \{\varepsilon\} \ \cup \ \{sab \in \sigma_g \mid sa \in P_{A[\boldsymbol{B}]}\}.$$

(Note that $\sigma_g = \sigma_{\boldsymbol{\mathcal{U}}}$ in this notation.) We say that σ_g is a *generic strategy* for A, and write $\sigma_g : A$, if the following *restriction condition* is satisfied:

- $\sigma_{\boldsymbol{B}} \subseteq P_{A[\boldsymbol{B}]}$ for all $\boldsymbol{B} \trianglelefteq \boldsymbol{\mathcal{U}}$, so that the restrictions are well-defined.

Note that $\sigma = \sigma_g$ has the following properties.

- σ is a non-empty set of even-length sequences, closed under even-length prefixes.
- σ is *deterministic*, meaning that

$$sab \in \sigma \ \wedge \ sac \in \sigma \ \ \Rightarrow \ \ b = c.$$

- σ is *history-free*, meaning that

$$sab \in \sigma \ \wedge \ t \in \sigma \ \wedge \ ta \in P_{A[\boldsymbol{\mathcal{U}}]} \ \ \Rightarrow \ \ tab \in \sigma.$$

- σ is *generic*:

$$s{\cdot}m_1[m'_1]{\cdot}m_2[m'_2] \in \sigma \wedge t \in \sigma \wedge t{\cdot}m_1[m''_1] \in P_{A[\boldsymbol{\mathcal{U}}]} \ \Rightarrow \ t{\cdot}m_1[m''_1]{\cdot}m_2[m''_1] \in \sigma.$$

These conditions imply that

$$s \cdot m_1[m_1'] \cdot m_2[m_2'] \in \sigma \;\Rightarrow\; m_1' = m_2').$$

Moreover, for any set $\sigma \subseteq P_{A[\mathcal{U}]}$ satisfying the above conditions, there is a least partial function $g : \mathcal{O}_A \rightharpoonup \mathcal{O}_A$ such that $\sigma = \sigma_g$. This function can be defined explicitly by

$$g(m_1) = m_2 \iff \exists s.\, s \cdot m_1[a] \cdot m_2[a] \in \sigma.$$

The equivalence $\approx_A$ on plays can be lifted to a partial equivalence (*i.e.* a symmetric and transitive relation) on strategies on A, which we also write as $\approx$. This is defined most conveniently in terms of a partial pre-order (transitive relation) $\lessapprox$, which is defined as follows.

$$\sigma \lessapprox \tau \;\equiv\; sab \in \sigma \wedge t \in \tau \wedge sa \approx_A ta' \implies \exists b'.\, ta'b' \in \tau \wedge sab \approx_A ta'b'.$$

We can then define

$$\sigma \approx \tau \;\equiv\; \sigma \lessapprox \tau \wedge \tau \lessapprox \sigma.$$

A basic well-formedness condition on strategies σ is that they satisfy this relation, meaning $\sigma \approx \sigma$. Note that for a generic strategy $\sigma = \sigma_{\mathcal{U}}$, using the equivalence on plays in $A[\mathcal{U}]$:

$$\sigma \approx \sigma \implies \sigma_{\boldsymbol{B}} \approx \sigma_{\boldsymbol{B}} \quad \text{for all } \boldsymbol{B} \trianglelefteq \mathcal{U}.$$

A cartesian closed category of games is constructed by taking *partial equivalence classes* of strategies, *i.e.* strategies modulo $\approx$, as morphisms. See [5] for details.

5.1 Copy-Cat Strategies

One additional property of strategies will be important for our purposes. A partial function $f : X \rightharpoonup X$ is said to be a *partial involution* if it is symmetric, *i.e.* if

$$f(x) = y \iff f(y) = x.$$

It is *fixed-point free* if we never have $f(x) = x$. Note that fixed-point free partial involutions on a set X are in bijective correspondence with pairwise disjoint families $\{x_i, y_i\}_{i \in I}$ of two-element subsets of X (*i.e.* the set of pairs $\{x, y\}$ such that $f(x) = y$, and hence also $f(y) = x$). Thus they can thought of as "abstract systems of axiom links". See [6,7] where a combinatory algebra of partial involutions is introduced, and an extensive study is made of realizability over this combinatory algebra.

For us, the important correspondence is with *copy-cat strategies*, first identified in [3] as central to the game-semantical analysis of proofs (and so-named there). We say that σ is a *copy-cat strategy* if $\sigma = \sigma_g$ where g is a fixed-point free partial involution.

Lemma 1 (The Copy-Cat Lemma). *Let $\sigma_g : A$ be a generic copy-cat strategy. If $g(m) = m'$, then for all $s \in \sigma$:*

$$s\lceil m \;=\; s\lceil m'.$$

6 The Model

We shall use the hyperdoctrine formulation of model of System F, as originally proposed by Seeley [22] based on Lawvere's notion of hyperdoctrines [14], and simplified by Pitts [18].

We begin by defining:

$$\mathcal{G}_{\mathcal{U}}(k) \;=\; \mathsf{Sub}(\mathcal{U}) \cap \mathcal{G}(k),$$

where $\mathcal{U}$ is the universe of System F types constructed in Section 6.

6.1 The Base Category

We firstly define a base category $\mathbb{B}$. The objects are natural numbers. A morphism $n \longrightarrow m$ is an m-tuple

$$\langle A_1, \ldots, A_m \rangle, \qquad A_i \in \mathcal{G}_{\mathcal{U}}(n),\ 1 \leq i \leq m.$$

Composition of $\langle A_1, \ldots, A_m \rangle : n \longrightarrow m$ with $\langle B_1, \ldots, A_n \rangle : k \longrightarrow n$ is by substitution:

$$\langle A_1, \ldots, A_m \rangle \circ \boldsymbol{B} \;=\; \langle A_1[\boldsymbol{B}], \ldots, A_m[\boldsymbol{B}] \rangle : k \longrightarrow m.$$

The identities are given by:

$$\mathsf{id}_n \;=\; \langle X_1, \ldots, X_n \rangle.$$

Note that variables act as projections:

$$X_i : n \longrightarrow 1$$

and we can define pairing by

$$\langle \boldsymbol{A}, \boldsymbol{B} \rangle \;=\; \langle A_1, \ldots, A_n, B_1, \ldots, B_m \rangle : k \longrightarrow m + n$$

where

$$\langle A_1, \ldots, A_m \rangle : k \longrightarrow n, \qquad \langle B_1, \ldots, B_m \rangle : k \longrightarrow m.$$

Thus this category has finite products, and is generated by the object 1, in the sense that all objects are finite powers of 1.

6.2 The Indexed CCC

Nest, we define a functor

$$\mathcal{C} : \mathbb{B}^{\mathrm{op}} \longrightarrow \mathbf{CCC}$$

where **CCC** is the category of cartesian closed categories with *specified* products and exponentials, and functors preserving this specified structure.

The cartesian closed category $\mathcal{C}(k)$ has as objects $\mathcal{G}_{\mathcal{U}}(k)$. Note that the objects of $\mathcal{C}(k)$ are the morphisms $\mathbb{B}(k, 1)$; this is part of the Seeley-Pitts definition.

The cartesian closed structure at the object level is given by the constructions on variable games which we have already defined: $A \Rightarrow B$, $A \,\&\, B$, $\mathbf{1}$. Note that $\mathcal{G}_{\mathcal{U}}(k)$ is closed under these constructions by Proposition 7.

A morphism $A \longrightarrow B$ in $\mathcal{C}(k)$ is a generic copy-cat strategy $\sigma : A \Rightarrow B$. Recall that this is actually defined at the "global instance" $\mathcal{U}$:

$$\sigma = \sigma_{\mathcal{U}} : (A \Rightarrow B)[\boldsymbol{\mathcal{U}}] \;\; = \;\; A[\boldsymbol{\mathcal{U}}] \Rightarrow B[\boldsymbol{\mathcal{U}}].$$

More precisely, morphisms are partial equivalence classes of strategies modulo $\approx$.

The cartesian closed structure at the level of morphisms is defined exactly as in [5].

Reindexing It remains to describe the functorial action of morphisms in $\mathbb{B}$. For each $\boldsymbol{C} : n \to m$, we must define a cartesian closed functor

$$\boldsymbol{C}^* : \mathcal{C}(m) \longrightarrow \mathcal{C}(n).$$

We define:

$$\boldsymbol{C}^*(A) \;=\; A[\boldsymbol{C}].$$

If $\sigma : A \Rightarrow B$,

$$\boldsymbol{C}^*(\sigma) = \sigma_{\boldsymbol{C}} : (A \Rightarrow B)[\boldsymbol{C}] \;\; = \;\; A[\boldsymbol{C}] \Rightarrow B[\boldsymbol{C}].$$

For functoriality, note that

$$\boldsymbol{C}^*(\sigma) \circ \boldsymbol{C}^*(\tau) = \sigma_{\boldsymbol{C}} \circ \tau_{\boldsymbol{C}} = (\sigma \circ \tau)_{\boldsymbol{C}} = \boldsymbol{C}^*(\sigma \circ \tau).$$

By Proposition 4, $\boldsymbol{C}^*$ preserves the cartesian closed structure.

6.3 Quantifiers as Adjoints

The second-order quantifiers are interpreted as right adjoints to projections. For each n, we have the projection morphism

$$\langle X_1, \ldots, X_n \rangle : n + 1 \longrightarrow n$$

in $\mathbb{B}$. This yields a functor

$$\boldsymbol{X}^* : \mathcal{C}(n) \longrightarrow \mathcal{C}(n + 1).$$

We must specify a right adjoint

$$\Pi_n : \mathcal{C}(n + 1) \longrightarrow \mathcal{C}(n)$$

to this functor. For $A \in \mathcal{G}_{\mathcal{U}}(n+1)$, we define

$$\Pi_n(A) \;=\; \forall X_{n+1}.\, A.$$

To verify the universal property, for each $C \in \mathcal{G}_{\mathcal{U}}(n)$ we must establish a bijection

$$\Lambda : \mathcal{C}(n)(C, \forall X_{n+1}.\, A) \;\stackrel{\cong}{\longrightarrow}\; \mathcal{C}(n+1)(\boldsymbol{X}^*(C), A).$$

Concretely, note firstly that

$$\boldsymbol{X}^*(C) \;=\; C[\boldsymbol{X}] \;=\; C.$$

Next, note that in both hom-sets the strategies are subsets of $P_{C[\boldsymbol{\mathcal{U}}] \Rightarrow A[\boldsymbol{\mathcal{U}}, \mathcal{U}/X_{n+1}]}$. In the case of generic strategies σ into A, these are subject to the constraint of the *restriction condition*: that is, for each instance $\boldsymbol{B}, B$,

$$\sigma_{\boldsymbol{B},B} \subseteq P_{C[\boldsymbol{B}] \Rightarrow A[\boldsymbol{B},B]}.$$

In the case of strategies σ into $\forall X_{n+1}.\, A$, these are subject to the constraint that for each instance $\boldsymbol{B}$,

$$\sigma_{\boldsymbol{B}} \subseteq P_{C[\boldsymbol{B}] \Rightarrow \forall X_{n+1}.\, A[\boldsymbol{B}, X_{n+1}]}.$$

The equivalence of these conditions follows straightforwardly from Proposition 8. This shows that the required correspondence between these hom-sets is simply the identity (which also disposes of the naturality requirements)!

Naturality (Beck-Chevalley) Finally, we must show that the family of right adjoints Π_n form an indexed (or fibred) adjunction. This amounts to the following: for each $\alpha : m \longrightarrow n$ in $\mathbb{B}$, we must show that

$$\alpha^* \circ \Pi_n = \Pi_m \circ (\alpha \times \mathsf{id}_1)^*.$$

Concretely, if $\alpha = \boldsymbol{C}$, we must show that for each $A \in \mathcal{G}_{\mathcal{U}}(n+1)$,

$$(\forall X_{n+1}.\, A)[\boldsymbol{C}] \;=\; \forall X_{m+1}.\, A[\boldsymbol{C}, X_{m+1}].$$

This is Proposition 9.

Remark We are now in a position to understand the logical significance of the relative polymorphic product $\Pi_i(A, B)$. We could define

$$\mathcal{G}_B(k) \;=\; \mathsf{Sub}(B) \,\cap\, \mathcal{G}(k),$$

and obtain an indexed category $\mathcal{C}_B(k)$ based on $\mathcal{G}_B(k)$ instead of $\mathcal{G}_{\mathcal{U}}(k)$. We would still have an adjunction

$$\mathcal{G}(n)(C, \Pi_{n+1}(A, B)) \;\cong\; \mathcal{C}_B(n+1)(\boldsymbol{X}^*(C), A).$$

However, in general B would not have sufficiently strong closure properties to give rise to a model of System F. Obviously, $\mathsf{Sub}(B)$ must be closed under the cartesian closed operations of product and function space. More subtly, $\mathsf{Sub}(B)$ must be closed under the polymorphic product $\Pi_i(-, B)$. (This is, essentially, the "small completeness" issue [13], although our ambient category of games does not have the requisite exactness properties to allow our construction to be internalised in the style of realizability models.[1]) This circularity, which directly reflects the impredicativity of System F, is resolved by the recursive definition of $\mathcal{U}$.

7 Homomorphisms

We shall now view games as *structures*, and introduce a natural notion of homomorphism between games. These will serve as a useful auxiliary tool in obtaining our results on genericity.

A homomorphism $h : A \longrightarrow B$ is a function

$$h : P_A \longrightarrow P_B$$

which is

- *length-preserving*: $|h(s)| = |s|$
- *prefix-preserving*: $s \sqsubseteq t \;\Rightarrow\; h(s) \sqsubseteq h(t)$
- *equivalence-preserving*: $s \approx t \;\Rightarrow\; h(s) \approx h(t)$.

There is an evident category **Games** with variable games as objects, and homomorphisms as arrows.

Lemma 2 (Play Reconstruction Lemma). *Let A, B be variable games. If we are given $s \in P_A$, and for each $m \in \mathcal{O}^i_A$, a play $t_m \in P_B$ with $|t_m| = \mathsf{numoccs}(m, s)$, then there is a unique $u \in P_{A[B/X_i]}$ such that:*

$$u{\restriction}A = s, \qquad u{\restriction}m = t_m \;\; (m \in \mathcal{O}^i_A).$$

This Lemma makes it easy to define a functorial action of variable games on homomorphisms. Let A be a variable game, and $h : B \longrightarrow C$ a homomorphism. We define

$$A(h) : A[B/X_i] \longrightarrow A[C/X_i]$$

by $A(h)(s) = t$, where

$$t{\restriction}A = s{\restriction}A, \qquad t{\restriction}m = h(s{\restriction}m), \;\; (m \in \mathcal{O}^i_A).$$

[1] However, by the result of Pitts [18], *any* hyperdoctrine model can be fully and faithfully embedded in an (intuitionistic) set-theoretic model.

Lemma 3 (Functoriality Lemma). *$A(h)$ is a well-defined homomorphism, and moreover this action is functorial:*

$$A(g \circ h) = A(g) \circ A(h), \qquad A(\mathsf{id}_B) = \mathsf{id}_{A[B/X_i]}.$$

The second important property is that *homomorphisms preserve plays of generic strategies.*

Lemma 4 (Homomorphism Lemma). *Let A be a variable game, $\sigma : A$ a generic strategy, and $h : C \longrightarrow D$ a homomorphism. Then*

$$s \in \sigma_{A[C/X_i]} \implies h(s) \in \sigma_{A[D/X_i]}.$$

8 Genericity

Our aim in this section is to show that there are generic types in our model, and indeed that, in a sense to be made precise, *most types are generic.*

We fix a variable game $A \in \mathcal{G}(1)$. Out aim is to find conditions on variable games B which imply that, for generic strategies $\sigma, \tau : A$:

$$\sigma_B \approx \tau_B \implies \Lambda(\sigma) \approx \Lambda(\tau) : \forall X.\, A.$$

Since, as explained in Section 8.3,

$$\Lambda(\sigma) \;=\; \sigma \;=\; \sigma_{\mathcal{U}},$$

this reduces to proving the implication

$$\sigma_B \approx \tau_B \implies \sigma_{\mathcal{U}} \approx \tau_{\mathcal{U}}.$$

Our basic result is the following.

Lemma 5 (Genericity Lemma). *If there is a homomorphism $h : \mathcal{U} \longrightarrow B$, then B is generic.*

Remark The Genericity Lemma applies to *any* variable type A; in particular, it is *not* required that A be a sub-game of $\mathcal{U}$. Thus our analysis of genericity is quite robust, and in particular is not limited to System F.

We define the *infinite plays* over a game A as follows: $s \in P_A^\infty$ if every finite prefix of s is in P_A. We can use this notion to give a simple sufficient condition for the hypothesis of the Genericity Lemma to hold.

Lemma 6. *If $P_B^\infty \neq \varnothing$, then B is generic.*

We now apply these ideas to the denotations of System F types, the objective being to show that "most" System F types denote generic instances in the model. Firstly, we define a notion of *length* for games, which we then transfer to types via their denotations as games.

We define

$$|A| \;=\; \sup\{|s| \mid s \in P_A\}.$$

Note that $|A| \leq \omega$.

We now show that any System F type whose denotation admits plays of length greater than 2 is in fact generic!

Lemma 7 (One, Two, Infinity Lemma). *If $|T| > 2$, then T is generic.*

We now give explicit syntactic conditions on System F types which imply that they are generic.

Proposition 10. *Let $T = \forall \boldsymbol{X}. T_1 \to \cdots \to T_k \to X$.*

1. *If for some $i : 1 \leq i \leq k$, $T_i = \forall \boldsymbol{Y}. U_1 \to \cdots \to U_l \to X$, then T is generic.*
2. *If for some $i : 1 \leq i \leq k$, $T_i = \forall \boldsymbol{Y}. U_1 \to \cdots \to U_l \to Y$, and for some $j : 1 \leq j \leq l$, $U_j = \forall \boldsymbol{Z}. V_1 \to \cdots \to V_m \to W$, where W is **either** some $Z_p \in \boldsymbol{Z}$, **or** Y, **or** some $X_q \in \boldsymbol{X}$, then T is generic.*

We apply this to the simple and familiar case of "ML types".

Corollary 1. *Let $T = \forall X. U$, where U is built from the type variable X and $\to$. If U is non-trivial (i.e. it is not just X), then T is generic.*

Examples The following are all examples of generic types.

- $\forall X. X \to X$
- $\forall X. (X \to X) \to X$
- $\forall X. (\forall Y. Y \to Y \to Y) \to X$.

Non-examples The following illustrate the (rather pathological) types which do not fall under the scope of the above results. Note that the first two both have length 1; while the third has length 2.

- $\forall X. X$
- $\forall X. \forall Y. X \to Y$.
- $\forall X. X \to \forall X. X$

Remark An interesting point illustrated by these examples is that our conditions on types are orthogonal to the issue of whether the types are inhabited in System F. Thus the type $\forall X. (X \to X) \to X$ is not inhabited in System F, but is generic in the games model, while the type $\forall X. X \to \forall X. X$ is inhabited in System F, but does not satisfy our conditions for genericity.

9 Related Work

A game semantics for System F was developed by Dominic Hughes in his D.Phil. thesis [12]. A common feature of his approach with ours' is that both give a direct interpretation of open types as certain games, and of type substitution as an operation on games. However, his approach is in a sense rather closer to syntax; it involves carrying type information in the moves, and the resulting model is much more complex. For example, showing that strategies in the model are closed under composition is a major undertaking. Moreover, the main result in [12] is a full completeness theorem essentially stating that the model is isomorphic to the term model of System F (with $\beta\eta$-equivalence), modulo types being reduced to their normal forms. As observed by Longo [16], the term model of System F *does not satisfy Genericity*; in fact, it does not satisfy Axiom (C). It seems that the presence of explicit type information in the moves will preclude the model in [12] from having genericity properties comparable to those we have established for our model.

References

1. M. Abadi, L. Cardelli and P.-L. Curien. Formal Parametric Polymorphism. In *Proc. 20th ACM Symposium on Principles of Programming Languages*, 1993.
2. S. Abramsky. Semantics of Interaction. In *Semantics and Logics of Computation*, edited by A. Pitts and P. Dybjer, Cambridge University Press 1997, 1–32.
3. S. Abramsky, R. Jagadeesan. Games and Full Completeness for Multiplicative Linear Logic, *J. of Symbolic Logic* **59**(2), 1994, 543–574.
4. S. Abramsky, R. Jagadeesan. A Game Semantics for Generic Polymorphism. Oxford University Computing Laboratory Programming Research Group, Research Report RR-03-02, 2003. Available on-line at `http://web.comlab.ox.ac.uk/oucl/publications/tr/rr-03-02`.
5. S. Abramsky, R. Jagadeesan, P. Malacaria. Full Abstraction for PCF, *Inf. and Comp.* **163**, 2000, 409–470.
6. S. Abramsky, M. Lenisa. A Fully-complete PER Model for ML Polymorphic Types, *CSL'00* Conf. Proc., P. Clote, H.Schwichtenberg eds., LNCS **1862**, 2000, 140–155.
7. S. Abramsky, M. Lenisa. A Fully Complete Minimal PER Model for the Simply Typed λ-calculus, *CSL'01* Conf. Proc., LNCS 2001.
8. S. Abramsky and G. McCusker. Games for Recursive Types. In C. Hankin, I. Mackie and R. Nagarajan, eds. *Theory and Formal Methods of Computing 1994*. Imperial College Press, 1995.
9. G. Berry. The Foundations of Esterel. In *Proof, Language and Interaction: Essays in honour of Robin Milner*, eds. G. Plotkin, C. Stirling and M. Tofte. MIT Press 2000, 425–454.
10. J.Y. Girard. Interprétation functionelle et élimunation des coupures de l'arithmètique d'ordre supérieur, Thèse d'Etat, Université Paris VII, 1972.
11. J.-Y. Girard, Y. Lafont and P. Taylor. *Proofs and Types*. Cambridge University Press 1989.
12. D. J. D. Hughes. *Hypergame Semantics: Full Completeness for System F*. D.Phil. thesis, University of Oxford, 1999.

13. J. M. E. Hyland. A small complete category. *Annals of Pure and Applied Logic*, 40, 1988.
14. F. W. Lawvere. Equality in hyperdoctrines and the comprehension schema as an adjoint functor, Proc. Symp. on Applications of Categorical Logic, 1970.
15. G. Longo, K. Milsted, S. Soloviev. The Genericity Theorem and Parametricity in the Polymorphic λ-Calculus, TCS 121(1&2):323–349, 1993.
16. G. Longo. Parametric and Type-Dependent Polymorphism. *Fundamenta Informaticae* 22(1/2):69–92.
17. Q.-Q. Ma and J. C. Reynolds. Types, Abstraction and Parametric Polymorphism, Part 2. In S. Brookes et al. editors, *Mathematical Foundations of Programming Language Semantics*. LNCS **598**, 1992.
18. A. Pitts. Polymorphism is set-theoretic constructively, *CTCS'88* Conf. Proc., D.Pitt ed., LNCS **283**, 1988.
19. G. Plotkin, M. Abadi. A Logic for Parametric Polymorphism, *TLCA'93* Conf. Proc., LNCS, 1993.
20. J. C. Reynolds. Towards a Theory of Type Structure. Programming Symposium, Proceedings, Paris 1974. LNCS **19**, 1974.
21. J. C. Reynolds. Types, Abstraction and Parametric Polymorphism. *Information Processing 83*, pp. 513–523, Elsevier (North-Holland), 1983.
22. R. A. G. Seeley. Categorical semantics for higher-order polymorphic lambda calculus. *Journal of Symbolic Logic*, 52(4):969–989, 1987.