A Game Semantics for Generic Polymorphism

Samson Abramsky
Oxford University Computing Laboratory

Radha Jagadeesan
DePaul University

Dedicated to Helmut Schwichtenberg on the occasion of his sixtieth birthday. His commitment to the highest scientific standards, coupled with a wise and kind humanity, is a continuing source of inspiration.

Abstract

Genericity is the idea that the same program can work at many different data types. Longo, Milstead 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) is proved for this model.

1 Introduction

We begin with an illuminating quotation from Gérard Berry [Ber00]:

> 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.

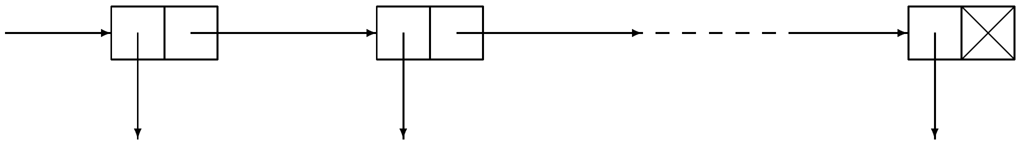

Figure 1: 'Generic' list structure

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.

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* [Rey83], which requires that relations between instances be preserved in a suitable sense by generic programs. This has led to numerous further developments, e.g. [MR92, ACC93, PA93].

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 [ACC93, PA93], 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 [LMS93, Lon95]. 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 [Gir72, Rey74]), Longo, Milsted and Soloviev proved in [LMS93] 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 [Lon95], 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 [AJM00] 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 [AJM00], 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.

- We prove Full Completeness for the ML types (*i.e.* the universal closures of quantifier-free types).

2 Background

2.1 Syntax of System F

We briefly review the syntax of System F. For further background information we refer to [GLT89].

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

$$\overline{\Gamma, x : T \vdash x : T}$$

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:

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

where each T_i is inductively of the same form.

2.2 Notation

We write ω for the set of natural numbers.

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 $X + Y$ for the disjoint union of sets X, Y.

If $Y \subseteq X$ and $s \in X^*$, we write $s \upharpoonright Y$ for the sequence obtained by deleting all elements not in Y from s. In practice, we use this notation in the context where $X = Y + Z$, and by abuse of notation we take $s \upharpoonright Y \in Y^*$, *i.e.* we elide the use of injection functions. We also use several variations on the notion of projection onto a sub-sequence, defining any which are not obvious from the context.

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":

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

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 sends i to m', and is the identity on all $j \neq i$.

Proposition 3.1 *Substitution is associative and left-cancellative:*

$$\begin{array}{ll} (1) & m_1[m_2[m_3]] = (m_1[m_2])[m_3] \\ (2) & m[m_1] = m[m_2] \implies m_1 = m_2 \end{array}$$

Note that substitution is right-cancellative *only up to permutation of generators*:

$$m[i][m'] = m[m'] = m[j][m'] \qquad \text{for all } i, j \in \omega.$$

Proposition 3.2 *Substitution interacts with λ and ρ as follows.*

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

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, $m_1 = m_2$.

Proposition 3.3 *If S is unambiguous, and for each $m \in S$ the set T_m is unambiguous, then so is the following set:*

$$\{m_1[m_2] \mid m_1 \in S \ \wedge \ m_2 \in T_{m_1}\}.$$

PROOF Suppose that $(m_1[m_2])[m_3] = (m_1'[m_2'])[m_3']$. We must show that $m_1[m_2] = m_1'[m_2']$. By associativity, $m_1[m_2[m_3]] = m_1'[m_2'[m_3']]$. Since S is unambiguous, this implies that $m_1 = m_1'$. By left cancellativity, this implies that $m_2[m_3] = m_2'[m_3']$. Since T_{m_1} is unambiguous, this implies that $m_2 = m_2'$. $\square$

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:

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

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:

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

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:

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

  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 $m \in \mathcal{O}^i_A$, $i > 0$, is an *occurrence* of the type variable X_i, while $m \in \mathcal{O}^0_A$ 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 [AJM00] in several 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 [AJM00] 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. Another motivation comes from considering copy-cat strategies, in which (essentially) the same moves are played alternately by O and P.

   Technically, modifying the alternation condition in this way simplifies the definition of substitution (see Section 3.4) and of the games X_i corresponding to type variables (see Section 3.5).

3. We have replaced the condition (**e1**) from [AJM00] with a stronger condition, which is in fact satisfied by the games in [AJM00].

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.

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

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.

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

$P_{A\Rightarrow B}$ 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 [AJM00]. 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 $B_1, \ldots, B_k \in \mathcal{G}(l)$, we define $A[\vec{B}] \in \mathcal{G}(l)$ as follows.

$$\mathcal{O}_{A[\vec{B}]} \quad = \quad \mathcal{O}_A^0 \ \cup \ \bigcup_{i=1}^{k} \{m[m'] \mid m \in \mathcal{O}_A^i \ \wedge \ m' \in \mathcal{O}_{B_i}\}.$$

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

$$s \approx_{A[\vec{B}]} t \ \equiv \ s{\restriction}A \approx_A t{\restriction}A \ \wedge \ \pi : s{\restriction}A \longleftrightarrow t{\restriction}A \ \Longrightarrow \ \forall i : 1 \leq i \leq k. \forall m \in \mathcal{O}_A^i.\, s{\restriction}m \approx_{B_i} t{\restriction}\pi(m).$$

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

Proposition 3.4 *$A[\vec{B}]$ is a well-defined game. In particular:*

1. *$\mathcal{O}_{A[\vec{B}]}$ is unambiguous.*
2. *$P_{A[\vec{B}]}$ satisfies the alternation condition.*
3. *$\approx_{A[\vec{B}]}$ satisfies **(e1)**–**(e3)**.*

Proof

1. This follows directly from Proposition 3.3, since by assumption $\mathcal{O}_A$ and each $\mathcal{O}_{B_i}$ are unambiguous.
2. We begin by formulating the alternation condition more precisely. We define the parity function
$$\mathsf{parity} : \omega \longrightarrow \{-1, +1\} \qquad\qquad \mathsf{parity}(k) = (-1)^k.$$
Also, for the purposes of this argument we shall interpret P as -1 and O as $+1$. We can now define the alternation condition on a sequence s as follows:
$$\forall t \cdot m \sqsubseteq s.\, \mathsf{parity}(|t|) = \mathsf{parity}(\mathsf{numoccs}(m, t))\lambda(m).$$
We now consider a play $t \cdot m_1[m_2] \in P_{A[\vec{B}]}$. Note firstly that if $\rho(m_1) = 0$, there is nothing more to prove, since in that case $t \cdot m_1[m_2]{\restriction}A = (t{\restriction}A) \cdot m_1[m_2]$ satisfies the alternation condition by assumption, and hence, since $|t| = |t{\restriction}A|$, so does $t \cdot m_1[m_2]$.
Otherwise, $\rho(m_1) > 0$. We shall use the following identities to verify the alternation condition for this play.
$$\begin{array}{lll}
(1) \ \ |t| & = & |t{\restriction}A| \\
(2) \ \ |t{\restriction}m_1| & = & \mathsf{numoccs}(m_1, t{\restriction}A) \\
(3) \ \ \mathsf{numoccs}(m_1[m_2], t) & = & \mathsf{numoccs}(m_2, t{\restriction}m_1) \\
(4) \ \ \lambda(m_1[m_2]) & = & \lambda(m_1)\lambda(m_2) \\
(5) \ \ \mathsf{parity}(|t{\restriction}A|) & = & \mathsf{parity}(\mathsf{numoccs}(m_1, t{\restriction}A))\lambda(m_1) \\
(6) \ \ \mathsf{parity}(|t{\restriction}m_1|) & = & \mathsf{parity}(\mathsf{numoccs}(m_2, t{\restriction}m_1))\lambda(m_2).
\end{array}$$

Of these, (1)–(3) are easily verified; (4) follows from Proposition 3.2; and (5) and (6) hold by assumption for plays in A and each B_i respectively. Now

$$\begin{array}{rcll} \mathsf{parity}(|t|) & = & \mathsf{parity}(|t\upharpoonright A|) & (1) \\ & = & \mathsf{parity}(\mathsf{numoccs}(m_1, t\upharpoonright A))\lambda(m_1) & (5) \\ & = & \mathsf{parity}(|t\upharpoonright m_1|)\lambda(m_1) & (2) \\ & = & \mathsf{parity}(\mathsf{numoccs}(m_2, t\upharpoonright m_1))\lambda(m_1)\lambda(m_2) & (6) \\ & = & \mathsf{parity}(\mathsf{numoccs}(m_1[m_2], t))\lambda(m_1[m_2]) & (3), (4) \end{array}$$

3. We verify **(e3)**. Suppose that $s \approx_{A[\vec{B}]} t$ and $s \cdot m_1[m_2] \in P_{A[\vec{B}]}$. This implies that $s\upharpoonright A \approx_A t\upharpoonright A$ and $(s\upharpoonright A) \cdot m_1 \in P_A$. By **(e3)** for A, for some m_1', $(s\upharpoonright A) \cdot m_1 \approx_A (t\upharpoonright A) \cdot m_1'$, and clearly if $\pi : (s\upharpoonright A) \cdot m_1 \longleftrightarrow (t\upharpoonright A) \cdot m_1'$, then $\pi(m_1) = m_1'$. If $\rho(m_1) = 0$, there is nothing more to prove. Otherwise, if $m_1 \in \mathcal{O}_A^i$, $1 \leq i \leq k$, then $s\upharpoonright m_1 \approx_{B_i} t\upharpoonright m_1'$, and $(s\upharpoonright m_1) \cdot m_2 \in P_{B_i}$. By **(e3)** for B_i, for some m_2', $(s\upharpoonright m_1) \cdot m_2 \approx_{B_i} (t\upharpoonright m_1') \cdot m_2'$. Clearly $s \cdot m_1[m_2] \approx_{A[\vec{B}]} t \cdot m_1'[m_2']$, as required.

□

3.4.1 Variants of substitution

Firstly, 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.

More importantly, it is useful to define an operation of substitution for one type variable only. We write this as

$$A[B/X_i]$$

where B is being substituted for the i'th type variable X_i, $i > 0$.

The definition is a simple variation on that of $A[\vec{B}]$ given above. Nevertheless, we give it explicitly, as we will make significant use of this version of substitution.

$$\mathcal{O}_{A[B/X_i]} = \bigcup_{j \neq i} \mathcal{O}_A^j \cup \{m[m'] \mid m \in \mathcal{O}_A^i \wedge m' \in \mathcal{O}_B\}.$$

$$P_{A[B/X_i]} = \{s \in M^*_{A[B/X_i]} \mid s\upharpoonright A \in P_A \wedge \forall m \in \mathcal{O}_A^i.\, s\upharpoonright m \in P_B\}$$

$$s \approx_{A[B/X_i]} t \equiv s\upharpoonright A \approx_A t\upharpoonright A \wedge \pi : s\upharpoonright A \longleftrightarrow t\upharpoonright A \implies \forall m \in \mathcal{O}_A^i.\, s\upharpoonright m \approx_B t\upharpoonright \pi(m).$$

3.5 Properties of substitution

Proposition 3.5 *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:*

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

Proof We show firstly that

$$\mathcal{O}_{A[B_1[\vec{C}],\ldots,B_k[\vec{C}]]} \;=\; \mathcal{O}_{(A[B_1,\ldots,B_k])[\vec{C}]}.$$

Expanding the definitions, we can write the occurrence set of the LHS of the equation as follows:

$$\mathcal{O}_A^0 \,\cup\, \bigcup_i \mathcal{O}_A^i[\mathcal{O}_{B_i}^0] \,\cup\, \bigcup_{i,j} \mathcal{O}_A^i[\mathcal{O}_{B_i}^j[\mathcal{O}_{C_j}]]$$

using the notation $S[T] = \{m_1[m_2] \mid m_1 \in S \;\wedge\; m_2 \in T\}$.

Similarly, the occurrence set of the RHS can be expanded to

$$\mathcal{O}_A^0 \,\cup\, (\bigcup_i \mathcal{O}_A^i[\mathcal{O}_{B_i}])^0 \,\cup\, \bigcup_j((\bigcup_i \mathcal{O}_A^i[\mathcal{O}_{B_i}^j])[\mathcal{O}_{C_j}].$$

Equating terms, the equality of these two sets follows from the fact that $\rho(m[m']) = \rho(m')$, and hence $S[T]^i = S[T^i]$, and that $m_1[m_2[m_3]] = (m_1[m_2])[m_3]$, and hence $S[T[U]] = (S[T])[U]$..

Next we show that the conditions on plays on the two sides of the equation are equivalent. Expanding the condition on plays on the LHS of the equation we see that $s \in P_{A[B_1[\vec{C}],\ldots,B_k[\vec{C}]]}$ if:

1. $s{\upharpoonright}A \in P_A$
2. $\forall i.\forall m \in \mathcal{O}_A^i.\, s{\upharpoonright}m{\upharpoonright}B_i \in P_{B_i}$
3. $\forall i.\forall j.\forall m \in \mathcal{O}_A^i.\forall m' \in \mathcal{O}_{B_i}^j.\, s{\upharpoonright}m{\upharpoonright}m' \in P_{C_j}$

Similarly, expanding the condition on plays on the RHS yields:

1. $s{\upharpoonright}A[\vec{B}]{\upharpoonright}A \in P_A$
2. $\forall i.\forall m \in \mathcal{O}_A^i.\, s{\upharpoonright}A[\vec{B}]{\upharpoonright}m \in P_{B_i}$
3. $\forall j.\forall m \in \mathcal{O}_{A[\vec{B}]}^j.\, s{\upharpoonright}m \in P_{C_j}$.

Note firstly that for any $m \in \mathcal{O}_{A[\vec{B}]}^j$, for some i, $m = m_1[m_2]$ for $m_1 \in \mathcal{O}_A^i$, $m_2 \in \mathcal{O}_{B_i}^j$. Now equating terms, we see that the equivalence of the two conditions is implied by the following equations:

1. $s{\upharpoonright}A[\vec{B}]{\upharpoonright}A = s{\upharpoonright}A$
2. $s{\upharpoonright}A[\vec{B}]{\upharpoonright}m = s{\upharpoonright}m{\upharpoonright}B_i \quad (m \in \mathcal{O}_A^i)$
3. $s{\upharpoonright}m_1{\upharpoonright}m_2 = s{\upharpoonright}m_1[m_2]$

These equations are easily verified from the definitions of the projection operations. Firstly, note that every move in these games has the form (1) $m_1[m_2[m_3]]$, where for some i, j: $m_1 \in \mathcal{O}^i_A$, $m_2 \in \mathcal{O}^j_{B_i}$, and $m_3 \in \mathcal{O}_{C_j}$; or the form (2) $m_1[m_2]$, where $m_1 \in \mathcal{O}^0_A$; or (3) $m_1[m_2[m_3]]$, where $m_1 \in \mathcal{O}^i_A$, $m_2 \in \mathcal{O}^0_{B_i}$. The LHS of equation (1) projects a move (1) firstly onto $m_1[m_2]$, then onto m_1, whereas the RHS projects it directly onto m_1. Moves of the form (2) are left unchanged in both cases; while moves of the form (3) are projected onto m_1 in both cases. In equation (2), the effect of the projection operations on both sides of the equation is to restrict the sequence to moves of the form $m[m_2[m_3]]$, and to project each such move onto m_2. Finally, the effect of both sides of equation (3) is to project $m_1[m_2[m_3]]$ onto m_3.

The argument for the coincidence of the equivalence relations is similar. $\square$

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

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

Proposition 3.6 *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 $A \in \mathcal{G}(k)$: $A[X_1, \ldots, X_k] \ = \ A$.

Proposition 3.7 *The cartesian closed structure commutes with substitution:*

1. $(A \Rightarrow B)[\vec{C}] \ = \ A[\vec{C}] \Rightarrow B[\vec{C}]$.
2. $(A \,\&\, B)[\vec{C}] \ = \ A[\vec{C}] \,\&\, B[\vec{C}]$.

Combining Propositions 3.6 and 3.7, we obtain:

Proposition 3.8 *The cartesian closed constructions can be obtained by substitution from their generic forms:*

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

4 Constructing a Universe for Polymorphism

4.1 Two Orders on Games

We will make use of two partial orders on games.

- The *approximation order* $A \sqsubseteq B$. This will be used in constructing games as solutions of recursive equations.
- The *inclusion order* $A \trianglelefteq B$. This will be used to define a notion of "subgame" within a suitable "universal game" in our construction of a model of System F.

4.1.1 The Approximation Order

We define $A \sqsubseteq B$ if:

- $\mathcal{O}_A \subseteq \mathcal{O}_B$
- $P_A = P_B \cap M_A^*$
- $s \approx_A t \iff s \in P_A \wedge s \approx_B t$

Thus if we are given B and $\mathcal{O}_A \subseteq \mathcal{O}_B$, then A is completely determined by the requirement that $A \sqsubseteq B$. Note that if $A \sqsubseteq B$ and $\mathcal{O}_A = \mathcal{O}_B$, then $A = B$.

This order was studied in the context of AJM games in [AM95], and the theory of recursively defined games was developed there and shown to work very smoothly, in direct analogy with the treatment of recursion on Scott information systems [Win93]. All of this theory carries over to the present setting essentially unchanged. The main facts which we will need can be summarized as follows.

Proposition 4.1 *1. $(\mathcal{G}(\omega), \sqsubseteq)$ is a (large) cpo, with least upper bounds of directed sets being given by componentwise unions.*

2. All the standard constructions on games, in particular product and function space, are monotonic and continuous with respect to the approximation order.

3. If a function $\mathcal{G}(\omega) \longrightarrow \mathcal{G}(\omega)$ is $\sqsubseteq$-monotonic, and continuous on move-sets, then it is $\sqsubseteq$-continuous.

Thus if

$$F : (\mathcal{G}(\omega), \sqsubseteq) \longrightarrow (\mathcal{G}(\omega), \sqsubseteq)$$

is continuous, we can solve the recursive equation

$$X \quad = \quad F(X)$$

using the least fixed point theorem in the standard fashion to construct a least solution in $\mathcal{G}(\omega)$.

4.1.2 The Inclusion Order

We define $A \trianglelefteq B$ by:

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

Thus the only difference between the two orders is the condition on plays. Note that

$$A \sqsubseteq B \implies A \trianglelefteq B.$$

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 4.2 *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 4.3 *Suppose that* $\mathcal{U}$ *is such that*

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

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

Proof Firstly,

$$A, B \in \mathsf{Sub}(\mathcal{U}) \text{ implies } A \Rightarrow B \trianglelefteq \mathcal{U} \Rightarrow \mathcal{U},$$

by $\trianglelefteq$-monotonicity of $\Rightarrow$. But $\mathcal{U} \Rightarrow \mathcal{U} \sqsubseteq \mathcal{U}$ by assumption, and since $\sqsubseteq \subseteq \trianglelefteq$, $A \Rightarrow B \trianglelefteq \mathcal{U}$, *i.e.* $A \Rightarrow B \in \mathsf{Sub}(\mathcal{U})$. Similarly, $\mathsf{Sub}(\mathcal{U})$ is closed under products. □

We also note the following for future reference.

Proposition 4.4 *Substitution* $A[B_1, \ldots, B_k]$ *is both* $\trianglelefteq$*-monotonic and* $\sqsubseteq$*-monotonic in* A *and each* B_i*,* $1 \leq i \leq k$.

Proof We show $\sqsubseteq$-monotonicity for plays. Suppose $A \sqsubseteq A'$ and $\vec{B} \sqsubseteq \vec{B}'$. If $s \in M^*_{A[\vec{B}]}$, then $s \restriction A = s \restriction A'$, and for $m \in \mathcal{O}^j_A$, $1 \leq j \leq k$, $s \restriction m \in M^*_{B_j}$, and hence, since $B_j \sqsubseteq B'_j$,

$$s \restriction m \in P_{B_j} \iff s \restriction m \in P_{B'_j}.$$

□

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 $\unlhd$) 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 4.5 *With notation as in the preceding paragraph, let $B = A_i^*(s)$.*

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

Proof Fix $s \in P_{A[\mathcal{U}/X_i]}$. For $C \in \mathsf{Sub}(\mathcal{U})$,,

$$s \in P_{A[C/X_i]} \iff \{s{\restriction}m \mid m \in \mathcal{O}_A^i\} \subseteq P_C.$$

By Proposition 4.2(2), $A_i^*(s)$ is the least $B \in \mathsf{Sub}(\mathcal{U})$ containing this set. □

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.* [Abr96]):

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

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 4.5.

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 4.6 *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:*

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

Proof We prove (2). Firstly, we compare the occurrence sets. Expanding the definitions on the LHS of the equation, we obtain

$$\mathcal{O}^0_A \;\cup\; \mathcal{O}^{k+1}_A[0] \;\cup\; \bigcup_{i=1}^{k}\{m_1[m_2] \mid m_1 \in \mathcal{O}^i_A \;\wedge\; m_2 \in \mathcal{O}_{C_i}\}$$

Similarly, on the RHS we obtain

$$\mathcal{O}^0_A \;\cup\; \bigcup_{i=1}^{k}\{m_1[m_2] \mid m_1 \in \mathcal{O}^i_A \;\wedge\; m_2 \in \mathcal{O}_{C_i}\} \;\cup\; \mathcal{O}^{n+1}_{A[\vec{C},X_{n+1}]}[0]$$

Since $\mathcal{O}^{n+1}_{A[\vec{C},X_{n+1}]}[0] = \mathcal{O}^{k+1}_A[n+1][0] = \mathcal{O}^{k+1}_A[0]$, we conclude that these two sets are equal.

We now show the equivalence of the conditions on plays. In similar fashion to the proof of associativity of substitution (Proposition 3.5), this is a straightforward matter of expanding the definitions. The main point is to show the equivalence of the conditions restricting plays in the polymorphic products. This reduces to showing that

$$A^*_{k+1}(s{\restriction}\Pi_{k+1}(A,B)) \;=\; A[\vec{C},X_{n+1}]^*_{n+1}(s),$$

which in turn reduces to showing that

$$\{s{\restriction}\Pi_{k+1}(A,B){\restriction}m \mid m \in \mathcal{O}^{k+1}_A\} \;=\; \{s{\restriction}m[n+1] \mid m \in \mathcal{O}^{k+1}_A\},$$

and finally to showing that for $m \in \mathcal{O}^{k+1}_A$,

$$s{\restriction}\Pi_{k+1}(A,B){\restriction}m \;=\; s{\restriction}m[n+1].$$

This holds because the projection $s{\restriction}\Pi_{k+1}(A,B)$ projects moves of the form $m'[m'']$ with $m' \in \mathcal{O}^i_A$, $1 \leq \rho_A(m') \leq k$, onto m', and leaves the sub-sequence of elements of the form $m[m'']$ unchanged. Finally, we note that projecting with m or $m[n+1]$ yields identical results. □

Proposition 4.7 *The relative polymorphic product* Π_i *is* $\trianglelefteq$*-monotonic and* $\sqsubseteq$*-continuous as a function*

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

Proof For $\sqsubseteq$-monotonicity, suppose $A \sqsubseteq A'$ and $B \sqsubseteq B'$. By Proposition 4.4, $A[B/X_i] \sqsubseteq A'[B'/X_i]$. For $t{\cdot}a \in M^*_{A[B/X_i]}$, the further conditions on plays $C^*_i(t{\cdot}a) = C^*_i(t)$, for $C = A$ or A', depend only on the sets

$$\{u{\restriction}m \mid m \in \mathcal{O}^i_A\}, \quad u = t \text{ or } t\cdot a$$

which depend only on u and not on C.

For $\sqsubseteq$-continuity, we use Proposition 4.1(3), by which it suffices to show continuity on occurrence sets. The action of Π_i on occurrence sets is just that of substitution, which is defined pointwise and hence preserves unions. □

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 $F : \mathcal{G}(\omega) \longrightarrow \mathcal{G}(\omega)$. This function is continuous by Propositions 4.1 and 4.7.

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

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

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

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

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 3.6, 3.7 and 4.6 shows that

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

and similarly for simultaneous substitution.

5 Strategies

Fix a variable game A. Let

$$g : \mathcal{O}_A \rightharpoonup \mathcal{O}_A$$

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

$$\hat{g} : M_{A[\vec{\mathcal{U}}]} \rightharpoonup M_{A[\vec{\mathcal{U}}]}$$

by

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

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

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

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

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

(Note that $\sigma_g = \sigma_{\vec{\mathcal{U}}}$ 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:

- $\sigma_{\vec{B}} \subseteq P_{A[\vec{B}]}$ for all $\vec{B} \trianglelefteq \vec{\mathcal{U}}$, 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

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

- σ is *generic*:

$$s \cdot m_1[m_1'] \cdot m_2[m_2'] \in \sigma \;\wedge\; t \in \sigma \;\wedge\; t \cdot m_1[m_1''] \in P_{A[\vec{\mathcal{U}}]} \;\;\Rightarrow\;\; t \cdot m_1[m_1''] \cdot m_2[m_1''] \in \sigma.$$

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 $\sigma \subseteq P_{A[\vec{\mathcal{U}}]}$ 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.

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

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 $\sigma = \sigma_{\vec{\mathcal{U}}}$, using the equivalence on plays in $A[\vec{\mathcal{U}}]$:

$$\sigma \approx \sigma \;\Longrightarrow\; \sigma_{\vec{B}} \approx \sigma_{\vec{B}} \quad \text{for all } \vec{B} \trianglelefteq \vec{\mathcal{U}}.$$

A cartesian closed category of games is constructed by taking *partial equivalence classes* of strategies, *i.e.* strategies modulo $\approx$, as morphisms. See [AJM00] 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 [AL00, AL01] 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 [AJ94a] 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 5.1 (The Copy-Cat Lemma) *Let $\sigma_g : A$ be a generic copy-cat strategy. If $g(m) = m'$, then for all $s \in \sigma$:*

$$s{\upharpoonright}m \quad = \quad s{\upharpoonright}m'.$$

PROOF By induction on $|s|$. The base case is immediate. Suppose that $s = t \cdot m_1[a] \cdot m_2[a]$ and that $g(m_3) = m_4$. By the partial involution property of g,

$$\{m_1, m_2\} = \{m_3, m_4\} \text{ or } \{m_1, m_2\} \cap \{m_3, m_4\} = \varnothing.$$

In the first case,

$$s{\upharpoonright}m_1 = (t{\upharpoonright}m_1) \cdot a = (t{\upharpoonright}m_2) \cdot a = s{\upharpoonright}m_2,$$

where the middle equation follows from the induction hypothesis.
In the second case,

$$s{\upharpoonright}m_3 = t{\upharpoonright}m_3 = t{\upharpoonright}m_4 = s{\upharpoonright}m_4,$$

where the middle equation again follows from the induction hypothesis. □

5.2 Cartesian Closed Structure

The required operations on morphisms to give the structure of a cartesian closed category can be defined exactly as for AJM games [AJM00]. We give the basic definitions, referring to [AJM00] for motivation and technical details.

We write $\mathsf{PInv}(X)$ for the set of partial involutions on a set X.

Proposition 5.2 *1. If $f \in \mathsf{PInv}(X)$ and $g \in \mathsf{PInv}(Y)$, then $f + g \in \mathsf{PInv}(X + Y)$.*

2. If $f \in \mathsf{PInv}(Y)$, then $\mathsf{id}_X \times f \in \mathsf{PInv}(X \times Y)$.

3. *Partial involutions are closed under conjugation by isomorphisms:*

$$f \in \mathsf{PInv}(X) \ \wedge \ \alpha : X \xrightarrow{\cong} Y \ \Longrightarrow \ \alpha \circ f \circ \alpha^{-1} \in \mathsf{PInv}(Y).$$

Our basic examples of partial involutions will be "twist maps" (*i.e.* symmetries) on disjoint unions:

$$\mathsf{twist}_X = [\mathsf{in}_2, \mathsf{in}_1] : X + X \longrightarrow X + X.$$

More generally, to get a partial involution on $\mathcal{O}_A$ we will specify $\mathcal{O}'_A \subseteq \mathcal{O}_A$ and $\mathcal{O}_1, \ldots, \mathcal{O}_k$ such that:

$$\mathcal{O}'_A \ \cong \ (\mathcal{O}_1 + \mathcal{O}_1) + \cdots + (\mathcal{O}_k + \mathcal{O}_k).$$

We then define a partial involution by conjugation by the indicated isomorphism of the evident disjoint union of k twist maps. The partial involution is undefined on $\mathcal{O}_A \setminus \mathcal{O}'_A$.

Identity For identity morphisms $\mathsf{id}_A : A \Rightarrow A$,

$$\mathcal{O}_{A \Rightarrow A} = \omega \times \mathcal{O}_A + \mathcal{O}_A.$$

Define $\mathcal{O}'_A = \{0\} \times \mathcal{O}_A + \mathcal{O}_A \subseteq \mathcal{O}_{A \Rightarrow A}$. Then

$$\mathcal{O}'_A \cong \mathcal{O}_A + \mathcal{O}_A,$$

so we obtain the required partial involution as a twist map. This is the basic example of a copy-cat strategy.

Projections Take for example $\pi_1 : A \,\&\, B \Rightarrow A$.

$$\mathcal{O}_{A \,\&\, B \Rightarrow A} = \omega \times (\mathcal{O}_A + \mathcal{O}_B) + \mathcal{O}_A.$$

Define

$$\mathcal{O}'_{A \,\&\, B \Rightarrow A} = \{0\} \times (\mathcal{O}_A + \varnothing) + \mathcal{O}_A \subseteq \mathcal{O}_{A \,\&\, B \Rightarrow A}.$$

Then $\mathcal{O}'_A \cong \mathcal{O}_A + \mathcal{O}_A$, and we obtain the required partial involution by conjugating the twist map by the evident isomorphism.

Pairing Suppose we are given partial involutions

$$f \in \mathsf{PInv}(\mathcal{O}_{C \Rightarrow A}), \qquad g \in \mathsf{PInv}(\mathcal{O}_{C \Rightarrow B}).$$

$$\mathcal{O}_{C \Rightarrow A \,\&\, B} = \omega \times \mathcal{O}_C + \mathcal{O}_A + \mathcal{O}_B.$$

Using some bijection $\omega \cong \omega + \omega$,

$$\begin{array}{rcl} \mathcal{O}_{C \Rightarrow A \,\&\, B} & \cong & (\omega + \omega) \times \mathcal{O}_C + \mathcal{O}_A + \mathcal{O}_B \\ & \cong & \omega \times \mathcal{O}_C + \omega \times \mathcal{O}_C + \mathcal{O}_A + \mathcal{O}_B \\ & \cong & (\omega \times \mathcal{O}_C + \mathcal{O}_A) + (\omega \times \mathcal{O}_C + \mathcal{O}_B) \\ & = & \mathcal{O}_{C \Rightarrow A} + \mathcal{O}_{C \Rightarrow B}. \end{array}$$

Then $f + g \in \mathsf{PInv}(\mathcal{O}_{C \Rightarrow A} + \mathcal{O}_{C \Rightarrow B})$, and conjugating by the indicated isomorphism yields the required partial involution.

Application For application

$$\mathsf{Ap}_{A,B} : (A \Rightarrow B) \,\&\, A \Rightarrow B,$$

$$\begin{array}{rcl}
\mathcal{O}_A & = & \omega \times ((\omega \times \mathcal{O}_A + \mathcal{O}_B) + \mathcal{O}_A) + \mathcal{O}_B \\
& \cong & \omega \times (\omega \times \mathcal{O}_A + \mathcal{O}_B) + \omega \times \mathcal{O}_A + \mathcal{O}_B \\
& \supseteq & \{0\} \times (\omega \times \mathcal{O}_A + \mathcal{O}_B) + \omega \times \mathcal{O}_A + \mathcal{O}_B \\
& \cong & (\omega \times \mathcal{O}_A + \omega \times \mathcal{O}_A) + (\mathcal{O}_B + \mathcal{O}_B),
\end{array}$$

yielding the required partial involution.

Currying Suppose that $f \in \mathsf{PInv}(\mathcal{O}_{A \,\&\, B \Rightarrow C})$.

$$\begin{array}{rcl}
\mathcal{O}_{A \,\&\, B \Rightarrow C} & = & \omega \times (\mathcal{O}_A + \mathcal{O}_B) + \mathcal{O}_C \\
& \cong & \omega \times \mathcal{O}_A + (\omega \times \mathcal{O}_B + \mathcal{O}_C) \\
& = & \mathcal{O}_{A \Rightarrow (B \Rightarrow C)}.
\end{array}$$

Conjugating f by the indicated isomorphism yields the required partial involution.

Composition Finally, we consider composition. We begin with some preliminaries on partial involutions. We write $\mathsf{Rel}(X)$ for the set of relations on a set X, *i.e.* $\mathsf{Rel}(X) = \mathcal{P}(X \times X)$. Note that $\mathsf{PInv}(X) \subseteq \mathsf{Rel}(X)$. We assume the usual regular algebra operations on relations: composition $R \cdot S$, union $R \cup S$, and reflexive transitive closure: $R^* = \bigcup_{k \in \omega} R^k$.

Any $R \in \mathsf{Rel}(X + Y)$ can be written as a disjoint union

$$R = R_{XX} \cup R_{XY} \cup R_{YX} \cup R_{YY},$$

where

$$R_{ST} = \{(a, b) \in R \mid a \in S \ \wedge \ b \in T\}.$$

Now given $R \in \mathsf{Rel}(X + Y)$, $S \in \mathsf{Rel}(Y + Z)$, we define $R \bowtie S \in \mathsf{Rel}(X + Z)$ as follows:

$$\begin{array}{rcl}
(R \bowtie S)_{XX} & = & R_{XX} \ \cup \ R_{XY} \cdot S_{YY} \cdot (R_{YY} \cdot S_{YY})^* \cdot R_{YX} \\
(R \bowtie S)_{XZ} & = & R_{XY} \cdot (S_{YY} \cdot R_{YY})^* \cdot S_{YZ} \\
(R \bowtie S)_{ZX} & = & S_{ZY} \cdot (R_{YY} \cdot S_{YY})^* \cdot R_{YX} \\
(R \bowtie S)_{ZZ} & = & S_{ZZ} \ \cup \ S_{ZY} \cdot R_{YY} \cdot (S_{YY} \cdot R_{YY})^* \cdot S_{YZ}.
\end{array}$$

Proposition 5.3 *1. $\bowtie$ is associative, with identity given by the twist map.*

2. If $f \in \mathsf{PInv}(X + Y)$ and $g \in \mathsf{PInv}(Y + Z)$, then $f \bowtie g \in \mathsf{PInv}(X + Z)$.

Proof For (1), see [AJ94a]. For (2), we write R^{c} for relational converse. Note that

$$(R \cdot S)^{\mathsf{c}} = S^{\mathsf{c}} \cdot R^{\mathsf{c}}, \quad (R \cup S)^{\mathsf{c}} = R^{\mathsf{c}} \cup S^{\mathsf{c}}, \quad (R^*)^{\mathsf{c}} = (R^{\mathsf{c}})^*, \quad R^{\mathsf{c}\,\mathsf{c}} = R.$$

If $R \in \mathsf{Rel}(X + Y)$, then

$$R = R^{\mathsf{c}} \iff R^{\mathsf{c}}_{XX} = R_{XX} \ \wedge \ R^{\mathsf{c}}_{XY} = R_{YX} \ \wedge \ R^{\mathsf{c}}_{YX} = R_{XY} \ \wedge \ R^{\mathsf{c}}_{YY} = R_{YY}.$$

Now if $R = R^{\mathsf{c}}, S = S^{\mathsf{c}}$:

$$\begin{array}{rcl}(R \bowtie S)^{\mathsf{c}}_{XX} & = & (R_{XX} \ \cup \ R_{XY} \cdot S_{YY} \cdot (R_{YY} \cdot S_{YY})^{*} \cdot R_{YX})^{\mathsf{c}} \\ & = & R^{\mathsf{c}}_{XX} \ \cup \ R^{\mathsf{c}}_{YX} \cdot (S^{\mathsf{c}}_{YY} \cdot R^{\mathsf{c}}_{YY})^{*} \cdot S^{\mathsf{c}}_{YY} \cdot R^{\mathsf{c}}_{XY} \\ & = & R_{XX} \ \cup \ R_{XY} \cdot (S_{YY} \cdot R_{YY})^{*} \cdot S_{YY} \cdot R_{YX} \\ & = & R_{XX} \ \cup \ R_{XY} \cdot S_{YY} \cdot (R_{YY} \cdot S_{YY})^{*} \cdot R_{YX} \\ & = & (R \bowtie S)_{XX},\end{array}$$

using the regular algebra identity $U \cdot (V \cdot U)^{*} = (U \cdot V)^{*} \cdot U$. The other cases are handled similarly. $\square$

Now suppose we are given

$$f \in \mathsf{PInv}(\mathcal{O}_{A \Rightarrow B}), \qquad g \in \mathsf{PInv}(\mathcal{O}_{B \Rightarrow C}).$$

$$\mathcal{O}_{A \Rightarrow B} = \omega \times \mathcal{O}_A + \mathcal{O}_B \qquad \mathcal{O}_{B \Rightarrow C} = \omega \times \mathcal{O}_B + \mathcal{O}_C.$$

Now $\mathsf{id}_{\omega} \times f \in \mathsf{PInv}(\omega \times (\omega \times \mathcal{O}_A + \mathcal{O}_B))$, but using some bijection $\omega \cong \omega \times \omega$,

$$\omega \times (\omega \times \mathcal{O}_A + \mathcal{O}_B) \ \cong \ (\omega \times \omega) \times \mathcal{O}_A + \omega \times \mathcal{O}_B \ \cong \ \omega \times \mathcal{O}_A + \omega \times \mathcal{O}_B.$$

Let $!f$ be the conjugation of $\mathsf{id}_{\omega} \times f$ by the indicated isomorphism. Then

$$!f \bowtie g \in \mathsf{PInv}(\omega \times \mathcal{O}_A + \mathcal{O}_C) = \mathsf{PInv}(\mathcal{O}_{A \Rightarrow C})$$

as required.

We show that composition is compatible with genericity at the level of partial involutions. Recall that $\hat{g} = g \times \mathsf{id}_{M_{\mathcal{U}}}$. Note that if $g \in \mathsf{PInv}(X)$, $\hat{g} \in \mathsf{PInv}(X \times M_{\mathcal{U}})$.

Proposition 5.4 *If $f \in \mathsf{PInv}(X + Y)$ and $g \in \mathsf{PInv}(Y + Z)$, then*

$$(f \times \mathsf{id}_U) \bowtie (g \times \mathsf{id}_U) = (f \bowtie g) \times \mathsf{id}_U \in \mathsf{PInv}((X + Z) \times U).$$

PROOF This is immediate from the definition of $\bowtie$, since $- \times \mathsf{id}_U$ distributes over composition and union:

$$(h \circ k) \times \mathsf{id}_U = (h \times \mathsf{id}_U) \circ (k \times \mathsf{id}_U), \qquad (h \cup k) \times \mathsf{id}_U = (h \times \mathsf{id}_U) \cup (k \times \mathsf{id}_U).$$

$\square$

Next, we give a direct definition of composition on strategies as sets of plays. If $\sigma_g : A \Rightarrow B$ and $\sigma_h : B \Rightarrow C$, we define

$$\sigma_g ; \sigma_h = \{ s \restriction A, C \mid s \in (\omega \times M_{A[\vec{\mathcal{U}}]} + \omega \times M_{B[\vec{\mathcal{U}}]} + M_{C[\vec{\mathcal{U}}]})^{*} \ \wedge \ s \restriction A, B \in \sigma_{!g} \ \wedge \ s \restriction B, C \in \sigma_h \}.$$

Proposition 5.5 $\sigma_g ; \sigma_h = \sigma_{!g \bowtie h}$.

PROOF See [AJ94a]. $\square$

Finally, we show the compatibility of composition with restrictions to instances.

Proposition 5.6 $(\sigma ; \tau)_{\vec{D}} = \sigma_{\vec{D}} ; \tau_{\vec{D}}$.

Remark The arbitrariness involved in the choice of bijections $\omega \cong \omega + \omega$ and $\omega \cong \omega \times \omega$ and the use of 0 as a particular element of ω in the above definitions is factored out by the partial equivalence $\approx$, as explained in [AJM00]. Note that all the other ingredients used in constructing the above isomorphisms are canonical, arising from the symmetric monoidal structures of cartesian product and disjoint union on the category of sets, and the distributivity of cartesian product over disjoint union. For the general axiomatics of the situation, see [AHS02].

6 The Model

We shall use the hyper-doctrine formulation of model of System F, as originally proposed by Seely [See87] based on Lawvere's notion of hyperdoctrines [Law70], and simplified by Pitts [Pit88]; a good textbook presentation can be found in [Cro93].

We begin with a key definition:

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

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 $\langle B_1, \ldots, B_n \rangle : k \longrightarrow n$ is by substitution:

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

The identities are given by:

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

Note that variables act as projections:

$$X_i : n \longrightarrow 1$$

and we can define pairing by

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

where

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

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

Next, 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 Seely-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 4.3.

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" $\mathcal{U}$:

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

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

The cartesian closed structure at the level of morphisms was described in Section 5.2.

Reindexing

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

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

We define:

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

If $\sigma : A \Rightarrow B$,

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

For functoriality, note that

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

By Proposition 3.7, $\vec{C}^*$ 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

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

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

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

Concretely, note firstly that

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

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

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

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

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

Thus if we show that these conditions are equivalent, the required correspondence between these hom-sets is simply the identity (which also disposes of the naturality requirements)!

Suppose firstly that σ satisfies the restriction condition. Assuming that $sab \in \sigma$, we must show that $A^*_{n+1}(sab) = A^*_{n+1}(sa)$. But if we let $B = A^*_{n+1}(sa)$, then by Proposition 4.5(1),

$$sa \in P_{C[\vec{B}]\Rightarrow A[\vec{B},B]},$$

and the restriction condition implies that

$$sab \in P_{C[\vec{B}]\Rightarrow A[\vec{B},B]}.$$

For the converse, suppose that $\sigma : C \Rightarrow \forall X_{n+1}.\, A$. To show that σ satisfies the restriction condition, choose an instance B. Suppose that $sab \in \sigma$ and $sa \in P_{C[\vec{B}]\Rightarrow A[\vec{B},B]}$. We must show that $sab \in P_{C[\vec{B}]\Rightarrow A[\vec{B},B]}$. Let $D = A^*_{n+1}(sa)$. Then by definition of $\forall X_{n+1}.\, A$, $sab \in A[\vec{B}, D]$, and by Proposition 4.5(2), $D \trianglelefteq B$. Hence by Proposition 4.4, $sab \in P_{C[\vec{B}]\Rightarrow A[\vec{B},B]}$ as required. □

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

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

This is Proposition 4.6.

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

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

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 [Hyl88], 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 7.1 (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:*

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

[1] However, by the result of Pitts [Pit88], *any* hyperdoctrine model can be fully and faithfully embedded in an (intuitionistic) set-theoretic model.

PROOF We can define u explicitly by:

$$\begin{array}{rcll} u_j & = & s_j, & \rho_A(s_j) \neq i \\ u_j & = & s_j[m], & \rho_A(s_j) = i \ \wedge \ (t_{s_j})_k = m, \end{array}$$

where j is the k'th position in s at which s_j occurs. $\Box$

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

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

Lemma 7.2 (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 7.3 (Homomorphism Lemma) *Let A be a variable game, $\sigma : A$ a generic strategy, and $h : C \longrightarrow D$ a homomorphism. Then*

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

PROOF By induction on $|s|$. The base case is trivial. For the inductive step, let

$$u \equiv s \cdot m_1[a] \cdot m_2[a] \in \sigma_{A[C/X_i]}.$$

By induction hypothesis, $A(h)(s) \in \sigma_{A[D/X_i]}$. By the Copy-Cat Lemma, $u \upharpoonright m_1 = u \upharpoonright m_2$. Let $h(u \upharpoonright m_1) = v \cdot b$. Then $A(h)(u) = A(h)(s) \cdot m_1[b] \cdot m_2[b]$, which is in $\sigma_{A[D/X_i]}$ by genericity of σ. $\Box$

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 5,

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

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

Proof We assume that $\sigma_B \approx \tau_B$, and show that $\sigma_{\mathcal{U}} \lessapprox \tau_{\mathcal{U}}$; a symmetric argument shows that $\tau_{\mathcal{U}} \lessapprox \sigma_{\mathcal{U}}$.
Suppose then that

$$s \cdot m_1[a] \cdot m_2[a] \in \sigma, \quad t \in \tau, \quad s \cdot m_1[a] \approx t \cdot m_1'[a'].$$

Let

$$\begin{array}{rcl} s' \cdot m_1[b] \cdot m_2[b] & = & A(h)(s \cdot m_1[a] \cdot m_2[a]), \\ t' \cdot m_1'[b'] & = & A(h)(t \cdot m_1'[a']). \end{array}$$

Then since $A(h)$ is a homomorphism,

$$s' \cdot m_1[b] \cdot m_2[b] \in P_B, \quad t' \cdot m_1'[b'] \in P_B, \quad s' \cdot m_1[b] \approx t' \cdot m_1'[b'].$$

By the Homomorphism Lemma,

$$s' \cdot m_1[b] \cdot m_2[b] \in \sigma, \quad t' \in \tau.$$

Since by assumption $\sigma_B \approx \tau_B$, there exists m_2' such that:

$$t' \cdot m_1'[b'] \cdot m_2'[b'] \in \tau \;\; \wedge \;\; s' \cdot m_1[b] \cdot m_2[b] \approx t' \cdot m_1'[b'] \cdot m_2'[b'].$$

Since τ is generic, this implies that

$$t \cdot m_1'[a'] \cdot m_2'[a'] \in \tau.$$

It remains to show that $s_1 \approx s_2$, where

$$s_1 \equiv s \cdot m_1[a] \cdot m_2[a], \quad s_2 \equiv t \cdot m_1'[a'] \cdot m_2'[a'].$$

Since by assumption

$$s \cdot m_1[a] \approx t \cdot m_1'[a'],$$

and $s' \cdot m_1[b] \cdot m_2[b] \approx t' \cdot m_1'[b'] \cdot m_2'[b']$ implies that $s_1 \restriction A \approx s_2 \restriction A$, it suffices to show that $s_1 \restriction m_2 \approx s_2 \restriction m_2'$. But by the Copy-Cat Lemma,

$$s_1 \restriction m_2 = (s \restriction m_1) \cdot a, \quad s_2 \restriction m_2' = (t \restriction m_1') \cdot a'.$$

But

$$s \cdot m_1[a] \approx t \cdot m_1'[a'] \implies (s \restriction m_1) \cdot a \approx (t \restriction m_1') \cdot a',$$

and the proof is complete. $\square$

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 8.2 *If $P_B^{\infty} \neq \varnothing$, then B is generic.*

PROOF Suppose $s \in P_B^{\infty}$. Let $s_n \in P_B$ be the restriction of s to the first n elements. We define $h : \mathcal{U} \longrightarrow B$ by: $h(t) = s_{|t|}$. It is trivially verified that this is a homomorphism. Genericity of B then follows by the Genericity Lemma. $\square$

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

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

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

PROOF Consider the normal form of T, which can be written as

$$\forall \vec{X}. T_1 \to \cdots \to T_k \to X.$$

If $|T| \geq 3$, then there is a play of length three, in which the first move must be made in the rightmost occurrence of X, the second in a copy of some T_i (by the definition of plays in the polymorphic product), and the third must also be played in that same copy of T_i (by the usual switching conditions). But then the second and third moves can be repeated arbitrarily often in different copies of T_i, giving rise to an infinite play. $\square$

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

Proposition 8.4 *Let $T = \forall \vec{X}. T_1 \to \cdots \to T_k \to X$.*

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

PROOF It is easily seen that types of the shapes described in the statement of the Proposition have plays of length 3. Indeed in the first case O plays in the rightmost occurrence of X in T, P responds in the rightmost occurrence of X in the given T_i, and then O can respond in that same occurrence of X. In the second case, O plays in X, P plays in Y, and then O can play in W. We then apply the previous Lemma. $\square$
We apply this to the simple and familiar case of "ML types".

Corollary 8.5 *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 Full Completeness

In this section, we prove full completeness for ML types. The full completeness proof exploits the decomposition of Intuitionist implication into Linear connectives. We give the basic definitions, referring to [AJM00] for motivation and technical details.

9.1 Linear Structure

The required operations on morphisms to give the categorical structure required to model the connectives of intuitionist multiplicative exponential linear logic can be defined exactly as for AJM games [AJM00].

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

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

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

$$m \ ::= \ i \ (i \in \omega) \ \mid \ \mathtt{p}(m) \ \mid \ \mathtt{q}(m) \ \mid \ \mathtt{k}_i(m) \ (i \in \omega) \ \mid \ \mathtt{l}(m) \mid \ \mathtt{r}(m) \mid \ \mathtt{l}^{\mathtt{t}}(m) \mid \ \mathtt{r}^{\mathtt{t}}(m).$$

The *labelling map* $\lambda : \mathcal{M}' \longrightarrow \{P, O\}$. The polarity algebra on the carrier $\{P, O\}$ interprets $\mathtt{p}$, $\mathtt{q}$, $\mathtt{r}, \mathtt{l}^{\mathtt{t}}, \mathtt{r}^{\mathtt{t}}$ and each $\mathtt{k}_i$ as the identity, and $\mathtt{l}$ 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.

Bang: !

$!A$ is defined as follows.

$$\mathcal{O}_{!A} \ = \ \{\mathtt{k}_i(m) \mid i \in \omega \ \wedge \ m \in \mathcal{O}_A\}.$$

$P_{!A}$ is defined to be the set of all sequences in $M_{!A}^*$ satisfying the alternation condition, and such that:

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

Let $S = \{\mathtt{k}_i(1) \mid i \in \omega\}$. Given a permutation α on ω, we define

$$\breve{\alpha}(\mathtt{k}_i(1)) = \mathtt{k}_{\alpha(i)}(1).$$

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

$$\exists \alpha \in S(\omega).\, \breve{\alpha}^*(s{\restriction}S) = t{\restriction}S.$$

This is essentially identical to the definition in [AJM00]. The only difference is that we use the revised version of the alternation condition in defining the positions.

Linear function space: $A \multimap B$

The linear function space $A \multimap B$ is defined as follows.

$$\mathcal{O}_{A \multimap B} \ = \ \{\mathtt{l}(m) \mid \ m \in \mathcal{O}_A\} \ \cup \ \{\mathtt{r}(m) \mid m \in \mathcal{O}_B\}.$$

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

- $s\restriction \mathtt{l}(1) \in P_A$.
- $s\restriction \mathtt{r}(1) \in P_B$.

Let $S = \{\mathtt{l}(1), \mathtt{r}(1)\}$. The equivalence relation $s \approx_{A \multimap B} t$ is defined by the condition

$$s\restriction S = t\restriction S \ \wedge \ s\restriction \mathtt{r}(1) \approx_B t\restriction \mathtt{r}(1) \ \wedge \ s\restriction \mathtt{l}(1) \approx_A t\restriction \mathtt{l}(1)).$$

Tensor: $A \otimes B$

The tensor $A \otimes B$ is defined as follows.

$$\mathcal{O}_{A\otimes B} \quad = \quad \{\mathtt{l}^{\mathtt{t}}(m) \mid \ m \in \mathcal{O}_A\} \ \cup \ \{\mathtt{r}^{\mathtt{t}}(m) \mid m \in \mathcal{O}_B\}.$$

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

- $s\restriction \mathtt{l}^{\mathtt{t}}(1) \in P_A$.
- $s\restriction \mathtt{r}^{\mathtt{t}}(1) \in P_B$.

Let $S = \{\mathtt{l}^{\mathtt{t}}(1), \mathtt{r}^{\mathtt{t}}(1)\}$. The equivalence relation $s \approx_{A\otimes B} t$ is defined by the condition

$$s\restriction S = t\restriction S \ \wedge \ s\restriction \mathtt{r}^{\mathtt{t}}(1) \approx_B t\restriction \mathtt{r}^{\mathtt{t}}(1) \ \wedge \ s\restriction \mathtt{l}^{\mathtt{t}}(1) \approx_A t\restriction \mathtt{l}^{\mathtt{t}}(1)).$$

9.2 Domain equation

Define the two orders $\trianglelefteq$, $\sqsubseteq$ on games as before (Section 4). We define a variable game $\mathcal{U}' \in \mathcal{G}(\omega)$ of second order types by the following recursive equation:

$$\mathcal{U}' = \&_{i>0} X_i \ \& \ \mathbf{1} \ \& \ (\mathcal{U}' \,\&\, \mathcal{U}') \ \& \ (\mathcal{U}' \multimap \mathcal{U}') \ \& \ (\mathcal{U}' \otimes \mathcal{U}') \ \& \ (!\mathcal{U}') \ \& \ \&_{i>0} \Pi_i(\mathcal{U}', \mathcal{U}').$$

Explicitly, $\mathcal{U}'$ is being defined as the least fixed point of a continuous function $F : \mathcal{G}(\omega) \longrightarrow \mathcal{G}(\omega)$.

We first summarize the key facts required to relate $\mathcal{U}$ and $\mathcal{U}'$. Define $A \Rightarrow B \ = \ !A \multimap B$.

Proposition 9.1

- $\Rightarrow$, $\&$, *Substitution and Relative Polymorphic Product are all* $\trianglelefteq$*-monotone.*
- $(\mathsf{Sub}(\mathcal{U}'), \trianglelefteq)$ *is a complete lattice.*

From this proposition, it is clear that $\mathcal{U}$ is essentially a subgame of $\mathcal{U}'$, with the proviso that the universe of moves underlying $\mathcal{U}$ is different from the universe of moves in $\mathcal{U}'$. More precisely, consider a *renaming map* $R : \mathcal{M} \longrightarrow \mathcal{M}'$, that interprets $\mathtt{p}$, $\mathtt{q}$, $\mathtt{r}$ of $\mathcal{M}$ as the operations with the same name on $\mathcal{M}'$, and $\mathtt{l}_i$ as $\mathtt{l} \circ \mathtt{k}_i$. The map on the generators is the "identity" map sending each $i \in \mathcal{M}$ to $i \in \mathcal{M}'$. Modulo this renaming map, $\mathcal{U}$ is a subgame of $\mathcal{U}'$.

The genericity results for $\mathcal{U}$ carry over to $\mathcal{U}'$, in particular the analog of lemma 8.1.

Lemma 9.2 *If $P_B^\infty \neq \varnothing$, then B is generic.*

In this light, since $\mathcal{U}$ is essentially a subgame of $\mathcal{U}'$, a full completeness result for $\mathcal{U}'$ implies full completeness for $\mathcal{U}$.

9.3 Full completeness

Consider an ML (universal closures of quantifier-free types) type T, *i.e.* $T = \forall \vec{X}.\,U$, where U is quantifier-free. In the light of lemma 9.2, it suffices to prove the result when the type variables are instantiated with a game ι such that $P_\iota^\infty \neq \varnothing$. Explicitly, suppose that given a strategy σ of type T, we can find a term $M : T$ such that $[\![M]\!]_\iota \approx \sigma_\iota$. Then genericity implies that $[\![M]\!]_{\mathcal{U}'} \approx \sigma_{\mathcal{U}'}$, and hence that $[\![M]\!] \approx \sigma$, as required.

We define ι as a well-opened subgame of $\mathcal{U}'$, to enable us to directly adopt the proofs from [AJM00]. A game B is *well-opened* [AJM00] if the opening moves of B can *only* appear as O-moves in opening positions. That is, for all $a \in M_B$, if $a \in P_B$ then

$$sa \in P_A \;\wedge\; |s| \text{ even} \implies s = \epsilon.$$

For notational convenience, we define ι as a subgame of $\mathcal{U}$. By the earlier discussion, there is a variant of ι that is a subgame of $\mathcal{U}'$. Consider the System F type $(\forall X)[(X \Rightarrow X) \Rightarrow X]$. Let $n \in \omega$. Consider the infinite position s given by:

$$\mathtt{r}(n) \cdot \mathtt{l}_1(\mathtt{r}(n)) \cdot \mathtt{l}_1(\mathtt{l}_1(n)) \cdot \mathtt{l}_2(\mathtt{r}(n)) \cdot \mathtt{l}_2(\mathtt{l}_1(n)) \cdot \mathtt{l}_3(\mathtt{r}(n)) \ldots$$

Define ι as the minimum game under the $\trianglelefteq$ order containing all the finite prefixes of s. This is constructed as in Lemma 4.2. Explicitly, ι is given as the set of positions that are equivalent to finite prefixes of s in $\mathcal{U}$. An examination of the equivalence in $\mathcal{U}$ reveals that ι is well-opened.

Consider an ML type in which all type variables are instantiated by ι. We now relate strategies in such types to $\beta\eta\Omega$-normal forms in the simply typed lambda calculus built on a single base type ι with a constant Ω at each type. Ω is interpreted in the model as the strategy $\perp$ which only contains the empty sequence. For completeness, we record the $\beta\eta\Omega$-normal forms.

- For all types T, $\Omega : T$ and $x : T$ are $\beta\eta\Omega$ normal forms.
- Let M_i be $\beta\eta\Omega$-normal forms at types T_i, $1 \le i \le k$. Let x be of type $T_1 \to \cdots \to T_k \to \iota$. Then $\lambda\vec{x}.\,xM_1 \ldots M_k$ is a $\beta\eta\Omega$-normal form.

The statement of the decomposition theorem requires some further notation from [AJM00]. Consider

$$(A_1 \,\&\, \ldots \,\&\, A_k) \Rightarrow \iota$$

where

$$A_i = B_{i,1} \Rightarrow \ldots B_{i,l_i} \Rightarrow \iota, \quad (1 \le i \le l_i).$$

If for some $1 \le i \le k$ and each $1 \le j \le l_i$ we have

$$\sigma_j : \tilde{A} \Rightarrow B_{i,j}$$

then we define

$$\mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}) : \tilde{A} \Rightarrow \iota$$

by

$$\mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}) = \mathtt{Ap} \circ \langle \ldots \mathtt{Ap} \circ \langle \pi_i, \sigma_1 \rangle, \ldots, \sigma_{l_i} \rangle.$$

With this notation, we are ready to state the decomposition lemma.

Proposition 9.3 (Decomposition Lemma) *Let $\sigma : (A_1 \& \ldots \& A_p) \Rightarrow (A_{p+1} \Rightarrow \ldots A_q \Rightarrow \iota)$ be any strategy, where*

$$A_i = B_{i,1} \Rightarrow \ldots B_{i,l_i} \Rightarrow \iota, \;\; 1 \leq i \leq q$$

We write $\tilde{C} = A_1, \ldots, A_p$, $\tilde{D} = A_{p+1}, \ldots, A_q$. (Notation : if $\tau : \tilde{C}, \tilde{D} \Rightarrow \iota$, then $\Lambda_{\tilde{D}}(\tau) : \tilde{C} \Rightarrow (A_{p+1} \Rightarrow \cdots \Rightarrow A_q \Rightarrow \iota)$.)

Then exactly one of the following cases applies.

(i) $\sigma = \Lambda_{\tilde{D}}(\bot_{\tilde{C},\tilde{D}})$.

(ii) $\sigma = \Lambda_{\tilde{D}}(\mathbf{C}_i(\sigma_1, \ldots, \sigma_{l_i}))$, *where $1 \leq i \leq q$, and*

$$\sigma_j : \tilde{C}, \tilde{D} \;\; \Rightarrow \;\; B_{i,j}, \;\; 1 \leq j \leq l_i$$

The proof follows standard arguments [AJM00, AL00]. In particular, since ι is well-opened, the Bang Lemma (Proposition 3.3.4 of [AJM00]) applies. The remainder of the proof follows Proposition 3.4.5 of [AJM00].

The Decomposition Lemma provides for one step of decomposition of an arbitrary strategy into a form matching that of $\beta\eta\Omega$ normal forms in the $\lambda\Omega$ calculus. However, infinite strategies such as the Y combinator will not admit a well-founded inductive decomposition process.

We conclude by describing a "finiteness" notion on strategies to identify the strategies for which the decomposition terminates. We define a notion of positive occurrences of !, following the usual definition of positive occurrences of variables in a formula. Consider a linear type built out of $!, \multimap$ and type variables. We define positive and negative occurrences of ! by structural induction.

- In $!A$, the positive occurrences of ! are the positive occurrences in A and the outermost !. The negative occurrences of ! in $!A$ are the negative occurrences in A.
- In $A \multimap B$, the positive occurrences of ! are the positive occurrences in B and the negative occurrences in A. The negative occurrences of ! are the negative occurrences in B and positive occurrences in A.

For any linear type T built out of $!, \multimap$ and ι, consider T' obtained by erasing the positive occurrences of ! from T. There is a canonical morphism $\delta_T : T \multimap T'$ built by structural induction from dereliction maps (at the positive occurrences of !) and identities (everywhere else). A strategy σ for an ML type $\forall X.F[X]$ is *finite* if there is a finite partial involution f inducing $\sigma_\iota ; \delta_{F[\iota]}$.

The decomposition process is well-founded for *finite* strategies.

Theorem 9.4 *For any ML type $\forall X.F[X]$, every finite strategy σ is definable by a $\lambda\Omega$ term in $\beta\eta\Omega$-normal form.*

Stronger results can be proved, although we will not enter into details here because of space restrictions. Firstly, if we extend the syntax of $\lambda\Omega$ terms to allow *infinite* terms (*i.e.* we take the ideal completion under the Ω-match ordering), then we can remove the finiteness hypothesis in the Theorem. Secondly, if we refine the game model to introduce a notion of winning infinite play, and use this to restrict to *winning strategies*, as in [Abr96], then we can obtain a full completeness result for the ML types of System F itself, without any need to introduce Ω into the syntax.

10 Related Work

A game semantics for System F was developed by Dominic Hughes in his D.Phil. thesis [Hug99]. A common feature of his approach with our's 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 [Hug99] 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 [Lon95], 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 [Hug99] from having genericity properties comparable to those we have established for our model.

The D.Phil thesis of Andrzej Murawski [Mur01] takes a broadly similar approach to modelling polymorphism to that of [Hug99], although the main focus in [Mur01] is on modelling Light Linear Logic.

Acknowledgements. Samson Abramsky was supported in part by UK EPSRC GR/R88861. Radha Jagadeesan was supported in part by NSF CCR-0244901.

References

[ACC93] M. Abadi, L. Cardelli and P.-L. Curien. Formal Parametric Polymorphism. In *Proc. 20th ACM Symposium on Principles of Programming Languages*, 1993.

[Abr96] S. Abramsky. Semantics of Interaction. In *Semantics and Logics of Computation*, edited by A. Pitts and P. Dybjer, Cambridge University Press 1997, 1–32.

[AHS02] S. Abramsky, E. Haghverdi and P. Scott. Geometry of Interaction and linear combinatory algebras. *Math. Struct. in Comp. Science* 12:625–665, 2002.

[AJ94a] S. Abramsky, R. Jagadeesan. Games and Full Completeness for Multiplicative Linear Logic, *J. of Symbolic Logic* **59**(2), 1994, 543–574.

[AJM00] S. Abramsky, R. Jagadeesan, P. Malacaria. Full Abstraction for PCF, *Inf. and Comp.* **163**, 2000, 409–470.

[AL00] 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.

[AL01] S. Abramsky, M. Lenisa. A Fully Complete Minimal PER Model for the Simply Typed λ-calculus, *CSL'01* Conf. Proc., LNCS 2001.

[AM95] 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.

[Ber00] 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.

[Cro93] R. Crole, *Categories for Types*, Cambridge University Press, 1993.

[Gir72] J.Y. Girard. Interprétation functionelle et élimunation des coupures de l'arithmètique d'ordre supérieur, Thèse d'Etat, Université Paris VII, 1972.

[GLT89] J.-Y. Girard, Y. Lafont and P. Taylor. *Proofs and Types.* Cambridge University Press 1989.

[Hug99] D. J. D. Hughes. *Hypergame Semantics: Full Completeness for System F.* D.Phil. thesis, University of Oxford, 1999.

[Hyl88] J. M. E. Hyland. A small complete category. *Annals of Pure and Applied Logic*, 40, 1988.

[HRR90] J. M. E. Hyland, E. Robinson, G. Rosolini. Algebraic types in PER models, *MFPS* Conf. Proc., M. Main *et al.* eds, LNCS **442**, 1990, 333–350.

[Law70] F. W. Lawvere. Equality in hyperdoctrines and the comprehension schema as an adjoint functor, Proc. Symp. on Applications of Categorical Logic, 1970.

[LMS93] G. Longo, K. Milsted, S. Soloviev. The Genericity Theorem and Parametricity in the Polymorphic λ-Calculus, TCS 121(1&2):323–349, 1993.

[Lon95] G. Longo. Parametric and Type-Dependent Polymorphism. *Fundamenta Informaticae* 22(1/2):69–92.

[MR92] 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.

[Mur01] A. Murawski. *On Semantic and Type-Theoretic Aspects of Polynomial-Time Computability.* D.Phil thesis, University of Oxford, 2001.

[Pit88] A. Pitts. Polymorphism is set-theoretic constructively, *CTCS'88* Conf. Proc., D.Pitt ed., LNCS **283**, 1988.

[PA93] G. Plotkin, M. Abadi. A Logic for Parametric Polymorphism, *TLCA'93* Conf. Proc., LNCS, 1993.

[Rey74] J. C. Reynolds. Towards a Theory of Type Structure. Programming Symposium, Proceedings, Paris 1974. LNCS **19**, 1974.

[Rey83] J. C. Reynolds. Types, Abstraction and Parametric Polymorphism. *Information Processing 83*, pp. 513–523, Elsevier (North-Holland), 1983.

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

[Win93] G. Winskel. *The Formal Semantics of Programming Languages.* MIT Press, 1993.