A Game Semantics for Generic Polymorphism

Samson Abramsky (University of Oxford)

Joint work with Radha Jagadeesan (DePaul University)

Research report OUCL PRG RR-03-02 available at
`http://web.comlab.ox.ac.uk/oucl/publications/tr/rr-03-02.html`

Theories of Genericity

Generics: the *same* program can work at many *different* data types.

Language feature: Parametrized types.
E.g. $\mathsf{List}[X]$.

Why is this important in programming?
It allows program to be written *at their maximum level of generality*, and hence of reusability.

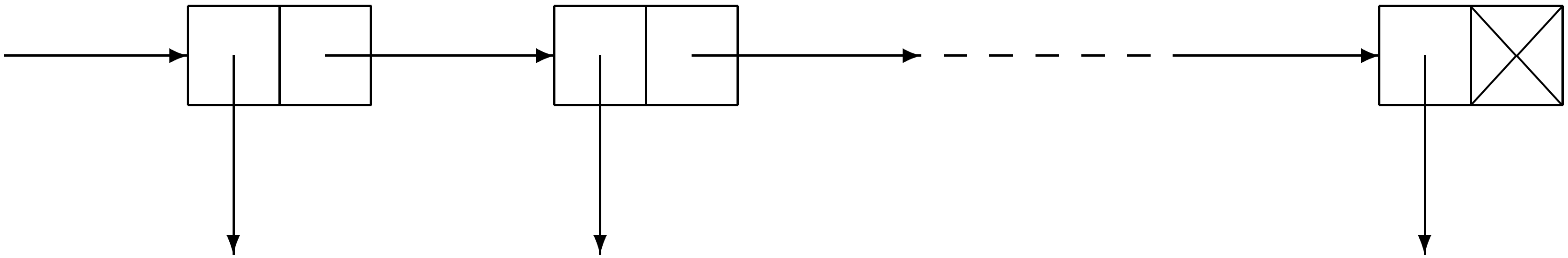
Although it is not always made explicit, the Write Things Once or WTO principle is clearly the basis for loops, procedures, higher-order functions, object-oriented programming and inheritance, concurrency vs. choice between interleavings, etc.

Gérard Berry:

In short, much of the search for high-level structure in programming is the search for *commonality.*

Genericity is one of the key examples.

Generic Programs



$$
\begin{array}{lcl}
\texttt{reverse} & : & \mathsf{List}[X] \longrightarrow \mathsf{List}[Y] \\
\texttt{maplist} & : & [X \to Y] \times \mathsf{List}[X] \longrightarrow \mathsf{List}[Y] \\
\texttt{sort} & : & \mathsf{List}[X] \times [X \times X \to \mathbf{Bool}] \longrightarrow \mathsf{List}[X] \\
\texttt{sumlist} & : & \mathsf{List}[\mathbf{num}] \longrightarrow \mathbf{num}
\end{array}
$$

Beyond concrete data structures

- Higher types
- Recursive types
- Embedded polymorphic types, $\forall X.\, T[X]$
- Sub-typing
- etc.

Second-order polymorphic λ-calculus

A core syntax for parametrized types and polymorphism is provided by the polymorphic λ-calculus. Despite its apparent simplicity, this little calculus is very powerful, and it is by no means straightforward to give it any reasonable semantics at all!

Types (Formulas):

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

Terms in context:

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

Typing rules

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

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

We seek a *semantic* understanding and characterization of genericity.

We will discuss two approaches:

- A 'functional' or extensional approach, exemplified by John Reynolds' relational parametricity.
- Genericity as a *constraint on the dynamics*, *i.e.* on the behaviour of programs. A syutactic approach to this idea was pioneered by Longo, Milsted and Soloviev, and a semantic approach initiated in the paper

  'A Game Semantics for Generic Polymorphism'

  by A. and Jagadeesan.

Relational Parametricity (Reynolds)

The idea is to understand a generic program

$$t_X : T[X]$$

in terms of the *uniformity properties* satisfied by the family of (monomorphic) instances

$$t_A : T[A].$$

Reynolds' idea is to express this uniformity in terms of *preservation of relations between instances*:

$$A \stackrel{R}{\longleftrightarrow} B \quad \Longrightarrow \quad t_A \stackrel{T[R]}{\longleftrightarrow} t_B.$$

Cf. Logical relations.

Relational parametricity is a beautiful and important notion, but not the whole story. In particular:

- It is a "pointwise" notion, which gets at genericity indirectly, rather than directly capturing the idea of a program written at the generic level.
- It is closely linked to strong extensionality principles, as shown e.g. in work by Plotkin, Abadi, Cardelli and Curien, whereas the intuition of generic programs not probing the structure of instances is *prima facie* an intensional notion.

A different idea

Take seriously the idea of a generic program as something existing in advance and independently of its instances, and characterized by its inability to probe the structure of its instances.

An ingenious *formal equational consequence* of this idea was proposed by Giuseppe Longo around 1993.

Genericity equationally

The Genericity Rule:

For $t, u : \forall X.\, A$:

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

Longo, Milsted and Soloviev proved 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). (NB: Genericity is a *conditional rule*).

Incompatibility of Parametricity and Genericity

In a relationally parametric model, the type

$$\forall X.\, X \to X$$

has only one element (the polymorphic identity function). Then the terms

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

will be equated at this instance, and hence by Genericity equated at all instances. But then for any $t, u : A$:

$$t = (\Lambda X.\, \lambda x{:}X.\, \lambda y{:}X.\, x)\{A\}\, t\, u = (\Lambda X.\, \lambda x{:}X.\, \lambda y{:}X.\, y)\{A\}\, t\, u = u \quad !$$

‘Duality’ of Parametricity and Genericity

interface type

implementation 1		client 1
implementation 2		client 2

Logical Relations csn be used to show equivalence of implementations; but in general, this will only hold with respect to the interface type provided to the clients. Genericity of the *clients* wrt. the interface type ensures that they can only observe the methods provided by the interface type, and hence cannot make irrelevant distinctions between the implementations.

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:

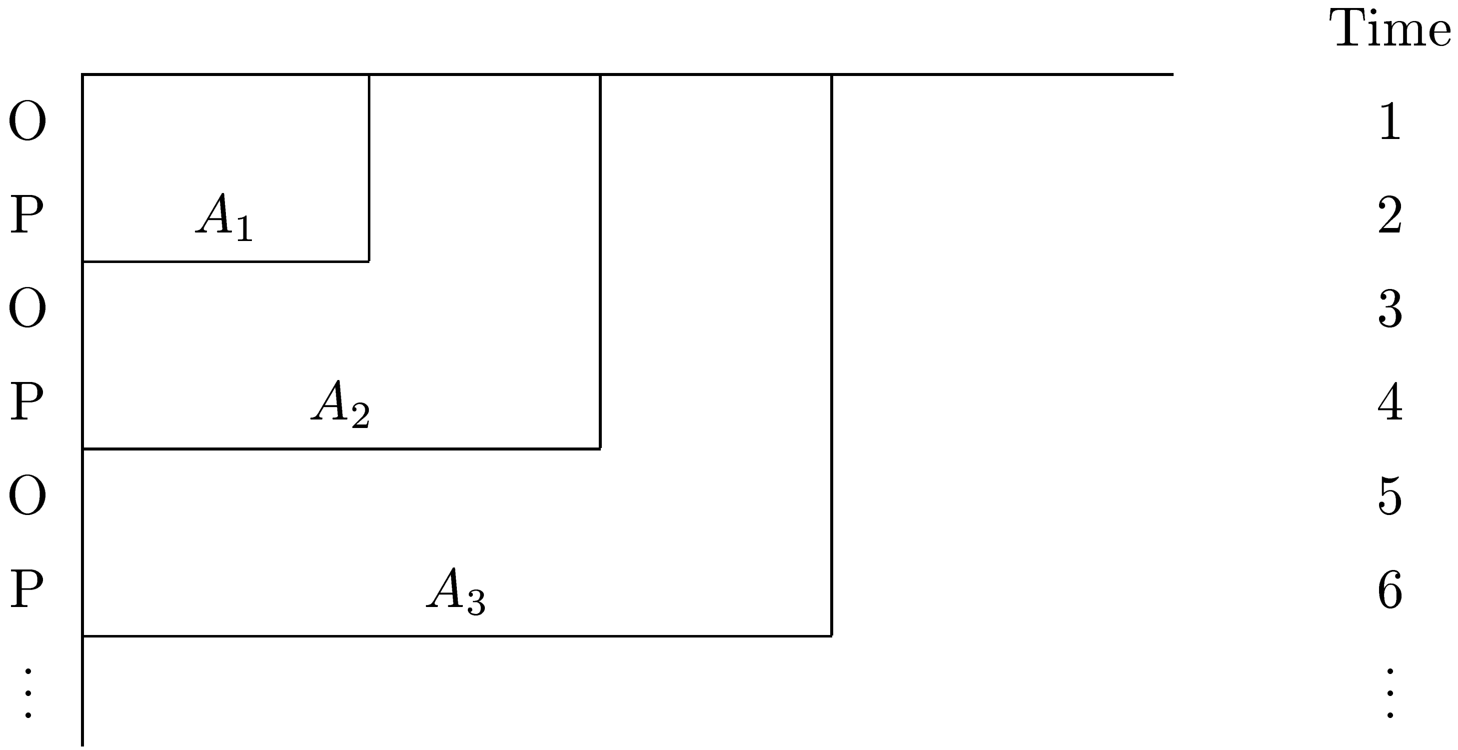
> Construct, at least, some (categorical) models that contain a collection of "generic" types. ...If our intuition about constructivity is correct, infinite objects in categories of (effective) sets should satisfy this property.

This is the problem to which a solution is given in the paper by A. and Jagadeesan.

Key novel ingredients in the model of generic polymorphism.

1. Game/interactive analysis of polymorphic quantification.
2. A novel treatment of the problem of size.
3. Semantic modelling of open types: the decomposition property.

Game for $\forall X.\, A$: the 'veil of ignorance'



Example

$\forall X.$ X	$\rightarrow$	X	
		a	O can play any opening move a
a			This is the only response P can make at this stage
b			O has to respond in the same sub-game (rules of $\rightarrow$), but can make any move in response
		b	Again P is forced

Note that P is essentially forced to play as the copy-cat strategy, i.e. the polymorphic identity $\Lambda X.\, \lambda x : X.\, x$, which is indeed the only normal form at the type $\forall X.\, X \rightarrow X$!

Formal definition: note alternating quantifiers

$\forall(F)$ is defined inductively as follows:

$$\begin{aligned}\forall(F) &= \{\epsilon\} \\ &\cup \{sa \mid s \in \forall(F)^{\text{even}} \;\wedge\; \exists A.\, sa \in F(A)\} \\ &\cup \{sab \mid sa \in \forall(F)^{\text{odd}} \;\wedge\; \forall A.\, sa \in F(A) \;\Rightarrow\; sab \in F(A)\}\end{aligned}$$

The first clause in the definition of $P_{\forall(F)}$ 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 (extending the current one). 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.

Using recursion to turn relative into absolute universes

We need a 'universe' of types (games) to bound the instances we need to consider.

We define a 'relative polymorphic product' $\forall_B X.\, A$ which uses B as a specified 'universe'. In general, B will not have good enough closure properties to make a suitable universe for modelling polymorphic λ-calculus.

However, we can define a 'Domain Equation for polymorphism':

$$\mathcal{U} = (\mathcal{U} \Rightarrow \mathcal{U}) \;\&\; \forall_{\mathcal{U}} X.\mathcal{U} \;\&\; \cdots$$

and then define

$$\forall X.\, A \;= \forall_{\mathcal{U}} X.\, A$$

and this works!

Semantic modelling of open types

We model open types (with free type variables) directly as 'generic games', rather than as functions from games to games.

Analogy: Polynomials vs. polynomial functions.

This allows us to *decompose* 'moves' or 'actions' within an instance $A[T]$ into the generic part from $A[X]$, and the part pertaining to the instance T. (**Intuition:** think of pattern-matching). This in turn lets us form a suitable constraint on strategies that what they do is determined purely by the generic part of the moves they are responding to, and is independent of the instance-specific part.

So we get a *direct*, semantic description of what it is to be a 'generic interactive process or strategy', in terms of positive, compositionally specified properties, which can be given independently of any particular syntax.

A Universe of Moves

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

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

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

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

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

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

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

- The *labelling map* $\lambda : \mathcal{M} \longrightarrow \{P, O\}$. The polarity algebra on the carrier $\{P, O\}$ interprets $\mathtt{p}$, $\mathtt{q}$, $\mathtt{r}$ as the identity, and each $\mathtt{l}_i$ as the involution $(\bar{\ })$, where $\bar{P} = O$, $\bar{O} = P$. The map on the generators is the constant map sending each i to O.

- The map $\rho : \mathcal{M} \longrightarrow \omega$ sends each move to the unique generator occurring in it.

- The substitution map. For each move $m' \in \mathcal{M}$, there is a map

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

  induced by the constant map on ω which sends each i to m'. We write $m[m']$ for $h_{m'}(m)$.

- An alternative form of substitution is written $m[m'/i]$. This is induced by the map which send i to m', and is the identity on all $j \neq i$.

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

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

We say that $m_1, m_2 \in \mathcal{M}$ are *unifiable* if for some $m_3, m_4 \in \mathcal{M}$, $m_1[m_3] = m_2[m_4]$. A set $S \subseteq \mathcal{M}$ is *unambiguous* if whenever $m_1, m_2 \in S$ are unifiable, $m_1 = m_2$.

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

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

Projections

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

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:

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

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:
  - $\lambda_A = \lambda \restriction \mathcal{O}_A$.
  - $\rho_A = \rho \restriction \mathcal{O}_A$.
  - $M_A = \{m[m'] \mid m \in \mathcal{O}_A^0 \ \wedge \ m' \in \mathcal{M}\} \ \cup \ \bigcup_{j>0} \mathcal{O}_A^j$.
- 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*.

- $\approx_A$ is an equivalence relation on P_A such that:

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

Here $s \longleftrightarrow t$ holds if

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

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

$$\pi : s \longleftrightarrow t$$

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

Occurrences, Open and Closed Games

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 the AJm paper in several minor respects.

1. The notion of bracketing condition, requiring a classification of moves as *questions* or *answers*, has been omitted.

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.

3. We have replaced the condition (**e1**) from the AJM paper with a slightly stronger condition, which is in fact satisfied by AJM games.

Constructions on games

The cartesian closed structure on AJM games can be lifted straighforwardly to variable games.

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

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

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

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

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

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

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

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

This is essentially identical to the definition in the AJM paper. 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 $!$.

Substitution

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

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

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

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

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

Properties of substitution

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

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

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

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

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

2. For all $A \in \mathcal{G}(k)$*:* $A[X_1, \ldots, X_k] \;=\; A$*.*

We can define a useful variant of substitution by:

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

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

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

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

Combining Propositions 3 and 4, we obtain:

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

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

The Inclusion Order We define $A \trianglelefteq B$ by:

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

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

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

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

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:

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

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 the minimality of $A_i^*(t)$.

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.

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

We can then define second-order quantification by:

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

Although it is not literally the case that

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

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

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

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

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:

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

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

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

- σ is *history-free*, meaning that

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

- σ is *generic*:

$$\begin{array}{l} s \cdot m_1[m_1'] \cdot m_2[m_2'] \in \sigma \ \wedge \ t \in \sigma \ \wedge \ t \cdot m_1[m_1''] \in P_{A[\vec{\mathcal{U}}]} \quad \Rightarrow \\ \qquad m_1' = m_2' \ \wedge \ t \cdot m_1[m_1''] \cdot m_2[m_1''] \in \sigma. \end{array}$$

Copy-Cat Strategies

A partial function $f : X \rightharpoonup X$ is said to be a *partial involution* if it is symmetric, *i.e.* if

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

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. They can thought of as "abstract systems of axiom links".

We say that σ is a *copy-cat strategy* if $\sigma = \sigma_g$ where g is a fixed-point free partial involution.

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

$$s \restriction m \quad = \quad s \restriction m'.$$

The Model

We shall use the hyperdoctrine formulation of model of System F.

We begin by defining:

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

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

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

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

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

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

The Indexed CCC

Nest, we define a functor

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

where **CCC** is the category of cartesian closed categories and functors.

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 $\mathbb{B}(k, 1)$

The cartesian closed structure at the object level is given by the constructions on variable games which we have already defined: $A \Rightarrow B$, $A \,\&\, B$, $\mathbf{1}$.

A morphism $A \longrightarrow B$ in $\mathcal{C}(k)$ is a generic copy-cat strategy $\sigma : A \Rightarrow B$. The cartesian closed structure at the level of morphisms is defined exactly as in the AJM paper.

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 $\sigma : A \Rightarrow B$,

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

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

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

Concretely, note firstly that

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

In both cases, the strategies are subsets of $P_{C[\vec{\mathcal{U}}] \Rightarrow A[\vec{\mathcal{U}}, \mathcal{U}/X_{n+1}]}$.
Generic strategies σ into A must satisfy, for each instance $\vec{B}, B$:

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

Strategies σ into $\forall X_{n+1}.\, A$ must satisfy

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

The equivalence of these conditions follows straightforwardly, so the required correspondence between these hom-sets is simply the identity!

Remark

For any 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. In particular, $\mathsf{Sub}(B)$ must be closed under the polymorphic product $\Pi_i(-, B)$. (This is, essentially, the “small completeness” issue). This circularity, which directly reflects the impredicativity of System F, is resolved by the recursive definition of $\mathcal{U}$.

Homomorphisms

A homomorphism $h : A \longrightarrow B$ is a function $h : P_A \longrightarrow P_B$ such that

$$|h(s)| = |s|, \quad s \sqsubseteq t \;\Rightarrow\; h(s) \sqsubseteq h(t), \quad s \approx t \;\Rightarrow\; h(s) \approx h(t).$$

Let A be a variable game, and $h : B \longrightarrow C$ a homomorphism. We define

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

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

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

Lemma 2 (Homomorphism Lemma) *Let A be a variable game, $\sigma : A$ a generic strategy, and $h : C \longrightarrow D$ a homomorphism. Then*

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

Genericity

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 $\sigma, \tau : A$:

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

Lemma 3 (Genericity Lemma) *If there is a homomorphism* $h : \mathcal{U} \longrightarrow B$*, then* B *is generic.*

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

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

Lemma 4 *If $P_B^\infty \neq \varnothing$, then B is generic.*

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

We define

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

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

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

Proposition 6 *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.*

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

Corollary 1 *Let $T = \forall \vec{X}. U$, where U is quantifier-free and 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.

Conclusion

- Details in paper in Proceedings of FOSSACS 2003.
- Even more details in full paper, web-accessible as Technical Report, details in Proceedings.
- Taking my own medicine: what we need is a *generic theory or tool-kit* for genericity.