Semantics of Interaction

Samson Abramsky

Abstract

The "classical" paradigm for denotational semantics models data types as *domains*, *i.e.* structured sets of some kind, and programs as (suitable) *functions* between domains. The semantic universe in which the denotational modelling is carried out is thus a category with domains as objects, functions as morphisms, and composition of morphisms given by function composition. A sharp distinction is then drawn between denotational and operational semantics. Denotational semantics is often referred to as "mathematical semantics" because it exhibits a high degree of mathematical structure; this is in part achieved by the fact that denotational semantics abstracts away from the dynamics of computation—from time. By contrast, operational semantics is formulated in terms of the syntax of the language being modelled; it is highly intensional in character; and it is capable of expressing the dynamical aspects of computation.

The classical denotational paradigm has been very successful, but has some definite limitations. Firstly, fine-structural features of computation, such as sequentiality, computational complexity, and optimality of reduction strategies, have either not been captured at all denotationally, or not in a fully satisfactory fashion. Moreover, once languages with features beyond the purely functional are considered, the appropriateness of modelling programs by functions is increasingly open to question. Neither concurrency nor "advanced" imperative features such as local references have been captured denotationally in a fully convincing fashion.

This analysis suggests a desideratum of *Intensional Semantics*, interpolating between denotational and operational semantics as traditionally conceived. This should combine the good mathematical structural properties of denotational semantics with the ability to capture dynamical aspects and to embody computational intuitions of operational semantics. Thus we may think of Intensional semantics as "Denotational semantics + time (dynamics)", or as "Syntax-free operational semantics".

A number of recent developments (and, with hindsight, some older ones) can be seen as contributing to this goal of Intensional Semantics. We will focus on the recent work on Game semantics, which has led to some striking advances in the Full Abstraction problem for PCF and other programming languages (Abramsky *et al.* 1995) (Abramsky and McCusker 1995) (Hyland and Ong 1995) (McCusker 1996a) (Ong 1996). Our aim is to give a genuinely elementary first introduction; we therefore present a simplified version of game semantics, which nonetheless

contains most of the essential concepts. The more complex game semantics in (Abramsky *et al.* 1995) (Hyland and Ong 1995) can be seen as refinements of what we present. Some background in category theory, type theory and linear logic would be helpful in reading these notes; suitable references are (Crole 1994), (Girard *et al.* 1989), (Girard 1995) (which contain much more than we will actually need).

Acknowledgements I would like to thank the Edinburgh "interaction group" (Kohei Honda, Paul-André Melliès, Julo Chroboczek, Jim Laird and Nobuko Yoshida) for their help in preparing these notes for publication, Peter Dybjer for his comments on a draft version, and Peter Dybjer and Andy Pitts for their efforts in organizing the CLiCS summer school and editing the present volume.

Contents

Notation

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 won't distinguish notationally between an element and the corresponding unit sequence. Thus as denotes the sequence with first element a and tail s.

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, $1 \leq i \leq |s|$.

Given a set S of sequences, we write S^{even}, S^{odd} for the subsets of even- and odd-length sequences respectively.

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

If $Y \subseteq X$ and $s \in X^*$, we write $s \restriction Y$ 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 $s \restriction Y \in Y^*$, *i.e.* we elide the use of injection functions.

We write $s \sqsubseteq t$ if s is a prefix of t, *i.e.* $t = su$ for some u.

$\mathrm{Pref}(S)$ is the set of prefixes of elements of $S \subseteq X^*$. S is *prefix-closed* if $S = \mathrm{Pref}(S)$.

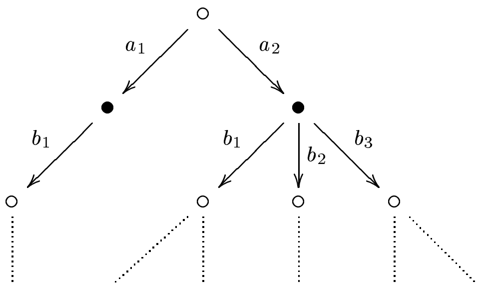
1 Game Semantics

We give a first introduction to game semantics. We will be concerned with 2-person games. Why the number 2? The key feature of games, by comparison with the many extant models of computation (labelled transition systems, event structures, etc. etc.) is that they provide an *explicit representation of the environment*, and hence model interaction in an intrinsic fashion. (By contrast, interaction is modelled in, say, labelled transition systems using some additional structure, typically a "synchronization algebra" on the labels.) One-person games would degenerate to transition systems; it seems that multi-party interaction can be adequately modeled by two-person games, in much the same way that functions with multiple arguments can be reduced to one-place functions and tupling. We will use such games to model interactions between a System and its Environment. One of the players in the game is taken to represent the System, and is referred to as Player or Proponent; the other represents the Environment and is referred to as Opponent. Note that the distinction between System and Environment and the corresponding designation as Player or Opponent depend on *point of view*:

> If Tom, Tim and Tony converse in a room, then from Tom's point of view, he is the System, and Tim and Tony form the Environment; while from Tim's point of view, he is the System, and Tom and Tony form the Environment.

A single 'computation' or 'run' involving interaction between Player and Opponent will be represented by a sequence of *moves*, made alternately by Player and Opponent. We shall adopt the convention that *Opponent always makes the first move*. This avoids a number of technical problems which would otherwise arise, but limits what we can successfully model with games to the *negative fragment* of Intuitionistic Linear Logic. (This is the $\otimes$, $\multimap$, $\&$, $!$, $\forall$ fragment).

A game specifies the set of possible runs (or 'plays'). It can be thought of as a tree



where hollow nodes $\circ$ represent positions where Opponent is to move; solid nodes $\bullet$ positions where Player is to move; and the arcs issuing from a node are labelled with the moves which can be made in the position represented by that node.

Formally, we define a game G to be a structure (M_G, λ_G, P_G), where

- M_G is the set of *moves* of the game;
- $\lambda_G : M_G \longrightarrow \{P, O\}$ is a labelling function designating each move as by Player or Opponent;
- $P_G \subseteq^{\mathrm{nepref}} M_G^{\mathrm{alt}}$, *i.e.* P_G is a non-empty, prefix-closed subset of M_G^{alt}, the set of alternating sequences of moves in M_G.

More formally, M_G^{alt} is the set of all $s \in M_G^*$ such that

$$\begin{array}{rl} \forall i : 1 \leq i \leq |s| & \mathrm{even}(i) \Longrightarrow \lambda_G(s_i) = P \\ \wedge & \mathrm{odd}(i) \Longrightarrow \lambda_G(s_i) = O \end{array}$$

i.e.

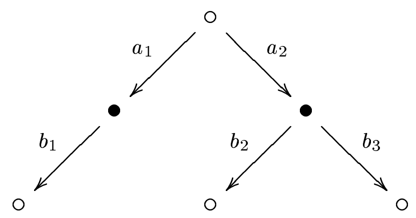
$$\begin{array}{cccccccc} s & = & a_1 & a_2 & \cdots & a_{2k+1} & a_{2k+2} & \cdots . \\ & \lambda_G & \downarrow & \downarrow & & \downarrow & \downarrow & \\ & & O & P & & O & P & \end{array}$$

Thus P_G represents the game tree by the prefix-closed language of strings labelling paths from the root. Note that the tree can have infinite branches, corresponding to the fact that there can be infinite plays in the game. In terms of the representation by strings, this would mean that all the finite prefixes of some infinite sequence of moves would be valid plays.

For example, the game

$$(\{a_1, a_2, b_1, b_2, b_3\}, \; \left\{ \begin{array}{ccccccccc} a_1 & , & a_2 & , & b_1 & , & b_2 & , & b_3 \\ \downarrow & & \downarrow & & \downarrow & & \downarrow & & \downarrow \\ O & & O & & P & & P & & P \end{array} \right\}, \; \{\epsilon, a_1, a_1 b_1, a_2, a_2 b_2, a_2 b_3\})$$

represents the tree



We are using games to represent the meaning of *logical formulas* or *types*. A game can be seen as specifying the possible interactions between a System and its Environment. In the traditional interpretation of types as structured sets of some kind, types are used to classify *values*. By contrast, games classify *behaviours*. *Proofs* or *Programs* will be modelled by *strategies*, *i.e.* rules specifying how the System should actually play.

Formally, we define a (deterministic) strategy σ on a game G to be a non-empty subset $\sigma \subseteq P_G^{\text{even}}$ of the game tree, satisfying:

$$\begin{array}{ll} (\mathbf{s1}) & \epsilon \in \sigma \\ (\mathbf{s2}) & sab \in \sigma \implies s \in \sigma \\ (\mathbf{s3}) & sab, sac \in \sigma \implies b = c. \end{array}$$

To understand this definition, think of

$$s = a_1 b_1 \cdots a_k b_k \in \sigma$$

as a record of repeated interactions with the Environment following σ. It can be read as follows:

If the Environment initially does a_1,
then respond with b_1;
If the Environment then does a_2,
then respond with b_2;
$\vdots$
If the Environment finally does a_k,
then respond with b_k.

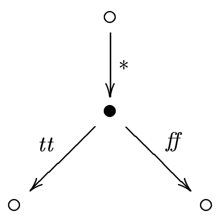
The first two conditions on σ say that it is a sub-tree of P_G of even-length paths. The third is a determinacy condition.

This can be seen as generalizing the notion of graph of a relation, *i.e.* of a set of ordered pairs, which can be read as a set of stimulus-response instructions. The generalization is that ordinary relations describe a single stimulus-response event only (giving rules for what the response to any given stimulus may be), whereas strategies describe repeated interactions between the System and the Environment. We can regard $sab \in \sigma$ as saying: 'when given the stimulus a in the context s, respond with b'. Note that, with this reading, the condition (s3) generalizes the usual single-valuedness condition for (the graphs of) partial functions. Thus a useful slogan is:

"Strategies are (partial) functions extended in time."

Example 1.1 Let $\mathbb{B}$ be the game

$$(\{*, tt, ff\}, \{* \mapsto O, tt \mapsto P, ff \mapsto P\}, \{\epsilon, *, *tt, *ff\})$$



This game can be seen as representing the data type of booleans. The opening move $*$ is a request by Opponent for the data, which can be answered by either tt or ff by Player. The strategies on $\mathbb{B}$ are as follows:

$$\{\epsilon\} \quad \mathrm{Pref}\{*tt\} \quad \mathrm{Pref}\{*\mathit{ff}\}$$

The first of these is the undefined strategy ('$\perp$'), the second and third correspond to the boolean values tt and ff. Taken with the inclusion ordering, this "space of strategies" corresponds to the usual flat domain of booleans:

$$\begin{tikzcd}
tt \arrow[dr, dash] & & \mathit{ff} \arrow[dl, dash] \\
& \perp &
\end{tikzcd}$$

Constructions on games

We will now describe some fundamental constructions on games.

Tensor Product

Given games A, B, we describe the tensor product $A \otimes B$.

$$\begin{array}{lcl}
M_{A\otimes B} & = & M_A + M_B \\
\lambda_{A\otimes B} & = & [\lambda_A, \lambda_B] \\
P_{A\otimes B} & = & \{s \in M_{A\otimes B}^{\mathrm{alt}} \mid s{\restriction}M_A \in P_A \wedge s{\restriction}M_B \in P_B\}
\end{array}$$

We can think of $A \otimes B$ as allowing play to proceed in *both* the subgames A and B in an interleaved fashion. It is a form of 'disjoint (*i.e.* non-communicating or interacting) parallel composition'.

A first hint of the additional subtleties introduced by the explicit representation of both System and Environment is given by the following result.

Proposition 1.1 *(Switching condition)*
*In any play $s \in P_{A\otimes B}$, if successive moves s_i, s_{i+1} are in different subgames (*i.e. *one is in A and the other in B), then $\lambda_{A\otimes B}(s_i) = P$, $\lambda_{A\otimes B}(s_{i+1}) = O$.*

In other words, only Opponent can switch from one subgame to another; Player must always respond in the same subgame that Opponent just moved in.

To prove this, consider for each $s \in P_{A\otimes B}$ the 'state'

$$\ulcorner s \urcorner = (\mathrm{parity}(s \restriction A), \mathrm{parity}(s \restriction B))$$

We will write O for even parity, and P for odd parity, since *e.g.* after a play of even parity, it is Opponent's turn to move. Initially, the state is $\ulcorner \epsilon \urcorner = (O, O)$.

Note that O can move in either sub-game in this state. If O moves in A, then the state changes to (P, O). P can now only move in the first component. After he does so, the state is back to (O, O). Thus we obtain the following 'state transition diagram':

$$
\begin{tikzcd}
& {} \arrow[d] & \\
& (O,O) \arrow[dl, "O"'] \arrow[dr, "O"] & \\
(P,O) \arrow[ur, bend left=60, "P"] & & (O,P) \arrow[ul, bend right=60, "P"']
\end{tikzcd}
$$

We see immediately from this that the switching condition holds; and also that the state (P, P) can never be reached (*i.e.* for no $s \in P_{A\otimes B}$ is $\ulcorner s \urcorner = (P, P)$).

Linear Implication

Given games A, B, we define the game $A \multimap B$ as follows:

$$
\begin{array}{lcl}
M_{A\multimap B} & = & M_A + M_B \\
\lambda_{A\otimes B} & = & [\overline{\lambda_A}, \lambda_B] \qquad\qquad \text{where } \overline{\lambda}_{\mathrm{A}}(m) = \begin{cases} P \text{ when } \lambda_A(m) = O \\ O \text{ when } \lambda_A(m) = P \end{cases} \\
P_{A\multimap B} & = & \{s \in M^{\mathrm{alt}}_{A\multimap B} \mid s \upharpoonright M_A \in P_A \wedge s \upharpoonright M_B \in P_B\}
\end{array}
$$

This definition is *almost* the same as that of $A \otimes B$. The crucial difference is the inversion of the labelling function on the moves of A, corresponding to the idea that on the left of the arrow the rôles of Player and Opponent are interchanged.

If we think of 'function boxes', this is clear enough:

$$
\begin{array}{ccc}
\text{Input} & & \text{Output} \\
\\
\longrightarrow & \boxed{\text{System}} & \longrightarrow
\end{array}
$$

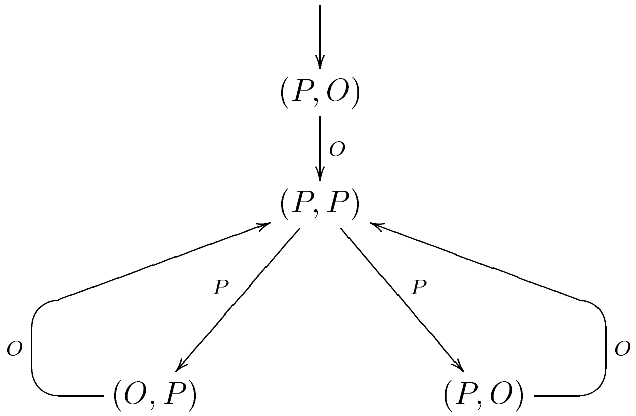
On the output side, the System is the producer and the Environment is the consumer; these rôles are reversed on the input side.

Note that $M^{\mathrm{alt}}_{A\multimap B}$, and hence $P_{A\multimap B}$, are in general quite different to $M^{\mathrm{alt}}_{A\otimes B}$, $P_{A\otimes B}$ respectively. In particular, the first move in $P_{A\multimap B}$ must always be in B, since the first move must be by Opponent, and all opening moves in A are labelled P by $\overline{\lambda_A}$.

We obtain the following switching condition for $A \multimap B$:

> If two consecutive moves are in different components, the first was by Opponent and the second by Player; so only Player can switch components.

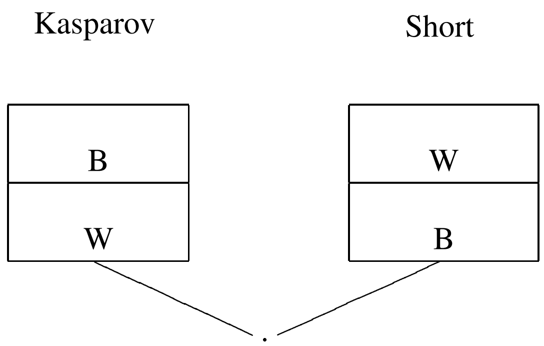
This is supported by the following state-transition diagram:



Example 1.2 The copy-cat strategy.

For any game A, we define a strategy on $A \multimap A$. This will provide the identity morphisms in our category, and the interpretation of logical axioms $A \vdash A$.

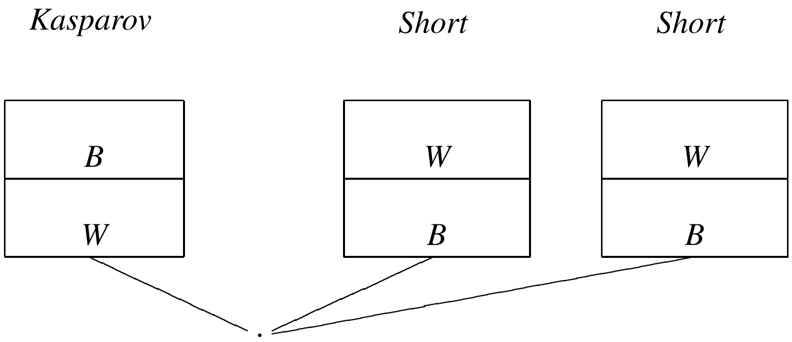
To illustrate this strategy, we undertake by the power of pure logic to beat a Grand-Master in chess. To do this, we play two games, one against, say, Kasparov, as White, and one against Short) as Black. The situation is as follows:



We begin with the game against Short. He plays his opening move, and we play his move in our game against Kasparov. After Kasparov responds, we play his move as our response to Short. In this way, we *play the same game twice*, but *once as White* and *once as Black*. Thus, whoever wins, we win one game. Otherwise put, we act as a buffer process, indirectly playing Kasparov off against Short.

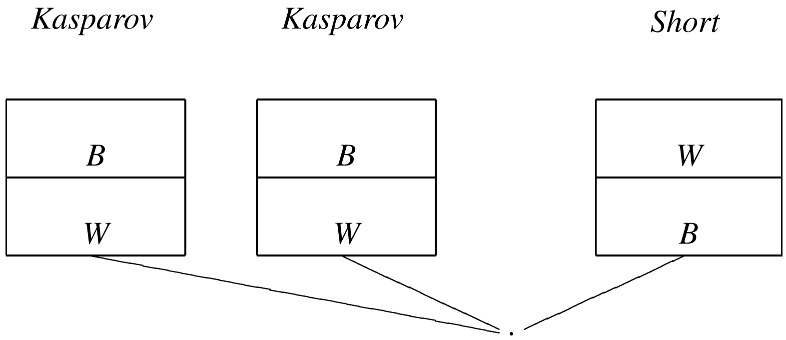
This copy-cat process can be seen as a 'dynamic tautology', by contrast with classical propositional tautologies, which are vacuous static descriptions of states of affairs. The logical aspect of this process is a certain 'conservation of flow of information' (which ensures that we win one game).

Exercise 1.1 *Suppose we had to play in* two games *against Short, both as Black,* *as well as one game against Kasparov as White.*



Would the same idea work?

How about playing in two games against Kasparov, both as White?



Comment on the logical significance of these observations.

In general, a copy-cat strategy on A proceeds as follows:

$$\begin{array}{ccccc} & A & \multimap & A & \\ \text{Time} & & & & \\ 1 & & & a_1 & \text{O} \\ 2 & a_1 & & & \text{P} \\ 3 & a_2 & & & \text{O} \\ 4 & & & a_2 & \text{P} \\ \vdots & & \vdots & & \vdots \end{array}$$

$$\mathtt{id}_A = \{s \in P^{\text{even}}_{A_1 \multimap A_2} \mid \forall t \text{ even-length prefix of } s:\ t{\restriction}A_1 = t{\restriction}A_2\}$$

(Here, we write A_1, A_2 to index the two occurrences of A in $A \multimap A$ for ease of reference. Note also that we write $s \restriction A_1$ rather than $s{\restriction}M_{A_1}$. We will continue with both these notational "abuses").

We indicate such a strategy briefly by $\overset{\frown}{A \;\multimap\; A}$, alluding to axiom links in the proof nets of Linear Logic.

Example 1.3 Application (*Modus Ponens*).

$$\mathtt{Ap}_{A,B} : (A \multimap B) \otimes A \multimap B$$

This is the conjunction of two copy-cat strategies

$$(A \quad \multimap \quad B) \quad \otimes \quad A \quad \multimap \quad B$$

Note that A and B each occur once positively and once negatively in this formula; we simply connect up the positive and negative occurrences by ‘copy-cats’.

$$\mathtt{Ap}_{A,B} = \{s \in P^{\text{even}}_{(A_1 \multimap B_1) \otimes A_2 \multimap B_2} \mid \forall t \text{ even-length prefix of } s : \; t{\restriction}A_1 = t{\restriction}A_2 \;\wedge\; t{\restriction}B_1 = t{\restriction}B_2\}$$

To understand this strategy as a protocol for function application, consider the following play:

	$(A$	$\multimap$	$B)$	$\otimes$	A	$\multimap$	B
O							ro
P			ro				
O	ri						
P					ri		
O					id		
P	id						
O			od				
P							od

ro — request for output
ri — request for input
id — input data
od — output data

The request for output to the application function is copied to the output side of the function argument; the function argument’s request for input is copied to the other argument; the input data provided at the second argument is copied back to the function argument; the output from the function argument is copied back to answer the original request. It is a protocol for *linear* function application since the state of both the function and the argument will change as we interact with them; we have no way of returning to the original state. Thus we “consume” our “resources” as we produce the output. In this way there is a natural match between game semantics and linear logic.

The Category of Games $\mathcal{G}$

- Objects: Games
- Morphisms: $\sigma : A \longrightarrow B$ are strategies σ on $A \multimap B$.
- Composition: interaction between strategies.

This interaction can be described as "parallel composition plus hiding".

$$\frac{\sigma : A \to B \quad \tau : B \to C}{\sigma ; \tau : A \to C}$$

$$\begin{aligned} \sigma;\tau \ &= \ (\sigma \parallel \tau)/B \ = \ \{s{\restriction}A, C \mid s \in \sigma \parallel \tau\} \\ \sigma \parallel \tau \ &= \ \{s \in (M_A + M_B + M_C)^* \mid s{\restriction}A, B \in \sigma \ \wedge \ s{\restriction}B, C \in \tau\}. \end{aligned}$$

(Note that we extend our abuse of notation for restriction here; by $s{\restriction}A, B$ we mean the restriction of s to $M_A + M_B$ as a "subset" of $M_A + M_B + M_C$, and similarly for $s{\restriction}A, C$ and $s{\restriction}B, C$.) This definition looks very symmetric, but the actual possibilities are highly constrained by the switching condition.

$$\begin{array}{ccccccc} A & \stackrel{\sigma}{\multimap} & B & \quad & B & \stackrel{\tau}{\multimap} & C \\ & & & & & & c_1 \\ & & & & b_1 & & \\ & & b_1 & & & & \\ & & b_2 & & & & \\ & & & & b_2 & & \\ & & \vdots & & \vdots & & \\ & & & & b_k & & \\ & & b_k & & & & \\ a_1 & & & & & & \end{array}$$

Initially, Opponent must move in C (say with c_1). We consider τ's response. If this is in C, then this is the response of $\sigma;\tau$ to c_1. If τ responds in B, say with b_1, then a move by Player in B in $B \multimap C$ is a move by Opponent in $A \multimap B$. So it makes sense to consider σ's response to b_1. If it is in A, this is the overall response of $\sigma;\tau$ to c_1. If σ responds with b_2 in B, then b_2 is a move by Opponent in $B \multimap C$, and we consider τ's response. Continuing in this way, we obtain a uniquely determined sequence.

$$c_1 b_1 b_2 \cdots b_k \cdots$$

If the sequence ends in a visible action in A or C, this is the response by the strategy $\sigma;\tau$ to the initial move c_1, with the internal dialogue between σ and τ in B being hidden from the Environment. Note that σ and τ may continue their internal dialogue in B forever. This is "infinite chattering" in CSP terminology, and "divergence by an infinite τ-computation" in CCS terminology.

As this discussion clearly shows composition in $\mathcal{G}$ is interaction between strategies. The following fact is useful in the analysis of composition.

The map $s \mapsto s{\restriction}A, C$ induces a surjective map

$$\psi : \sigma \parallel \tau \ \longrightarrow \ \sigma;\tau$$

Covering Lemma. ψ is injective (and hence bijective) so for each $t \in \sigma;\tau$ there is a unique $s \in \sigma \parallel \tau$ such that $s \upharpoonright A, C \ = t$.

If $t = m_1 m_2 \ldots. m_k$, then s has the form

$$m_1 u_1 m_2 u_2 \ldots. u_{k-1} m_k$$

where $u_i \in M_B^*$, $1 \leq i < k$.

Exercise 1.2 Prove the Covering lemma by formalizing the preceding discussion.

An alternative definition of Cut

We give a more direct, 'computational' definition.

$$\sigma;\tau = \{s;t \mid s \in \sigma \wedge t \in \tau \wedge s{\upharpoonright}B = t{\upharpoonright}B\}.$$

This defines Cut 'pointwise' via an operation on single plays. This latter operation is defined by mutual recursion of four operations covering the following situations:

$$\begin{array}{lll}
1. & s \,\lfloor\!\lfloor\, t & \text{O is to move in } A. \\
2. & s \,\rfloor\!\rfloor\, t & \text{O is to move in } C. \\
3. & s \,\backslash\!\!\backslash\, t & \sigma \text{ to move.} \\
4. & s \,/\!\!/\, t & \tau \text{ to move.}
\end{array}$$

$$\begin{array}{rcll}
\gamma s \,\lfloor\!\lfloor\, t & = & \gamma(s \,\backslash\!\!\backslash\, t) & \\
\varepsilon \,\lfloor\!\lfloor\, t & = & \varepsilon & \\
s \,\rfloor\!\rfloor\, bt & = & b(s \,/\!\!/\, t) & \\
s \,\rfloor\!\rfloor\, \varepsilon & = & \varepsilon & \\
\gamma s \,\backslash\!\!\backslash\, t & = & \gamma(s \,\lfloor\!\lfloor\, t) & (\gamma \in M_\Gamma) \\
as \,\backslash\!\!\backslash\, at & = & s \,/\!\!/\, t & (a \in M_A) \\
s \,/\!\!/\, bt & = & b(s \,\rfloor\!\rfloor\, t) & (b \in M_B) \\
as \,/\!\!/\, at & = & s \,\backslash\!\!\backslash\, t & (a \in M_A)
\end{array}$$

We can then define

$$s;t \ = \ s \,\rfloor\!\rfloor\, t.$$

Exercise 1.3 Prove that the two definitions of $\sigma;\tau$ coincide.

Proposition 1.2 *$\mathcal{G}$ is a category.*

In particular, $\mathtt{id}_A : A \longrightarrow A$ is the copy-cat strategy described previously.

Exercise 1.4 Verify this Proposition.

Exercise 1.5 Define a strategy $\mathtt{not}\colon \mathbb{B} \longrightarrow \mathbb{B}$ on the boolean game defined previously to represent Boolean complement. Calculate explicitly the strategies

$$\bot; \mathtt{not} \qquad tt; \mathtt{not} \qquad \mathit{ff}; \mathtt{not}$$

and hence show that this strategy does indeed represent the intended function. (For this purpose, treat strategies σ on $\mathbb{B}$ as strategies $\sigma : I \longrightarrow \mathbb{B}$ where

$$I \;=\; (\emptyset,\; \emptyset,\; \{\varepsilon\})$$

is the empty game, so that the above compositions make sense).

Exercise 1.6 Embed the category of sets and partial functions faithfully into $\mathcal{G}$. Is your embedding full? What about the category of flat domains and monotone maps?

Tensor structure of $\mathcal{G}$

We will now see (in outline) that $\mathcal{G}$ is an "autonomous" = symmetric monoidal closed category, and hence a model for IMLL, Intuitionistic Multiplicative Linear Logic.

We have already defined the tensor product $A \otimes B$ on objects. Now we extend it to morphisms:

$$\frac{\sigma : A \to B \quad \tau : A' \to B'}{\sigma \otimes \tau : A \otimes A' \to B \otimes B'}$$

$$\sigma \otimes \tau \;=\; \{s \in P^{\mathrm{even}}_{A \otimes A' \multimap B \otimes B'} \mid s \upharpoonright A, B \in \sigma \;\wedge\; s \upharpoonright A', B' \in \tau\}.$$

This can be seen as disjoint (i.e. non-communicating) parallel composition of σ and τ.

Exercise 1.7 Check functoriality, i.e. the equations

- $(\sigma \otimes \tau); (\sigma' \otimes \tau') \;=\; (\sigma; \sigma') \otimes (\tau; \tau')$.
- $\mathtt{id}_A \otimes \mathtt{id}_B \;=\; \mathtt{id}_{A \otimes B}$.

The tensor unit is defined by:

$$I \;=\; (\emptyset,\; \emptyset,\; \{\varepsilon\})$$

The canonical isomorphisms are conjunctions of copy-cat strategies.

$\mathtt{assoc}_{A,B,C}$: $$(A \otimes B) \otimes C \xrightarrow{\sim} A \otimes (B \otimes C)$$

$$(A \quad \otimes \quad B) \quad \otimes \quad C \quad \multimap \quad A \quad \otimes \quad (B \quad \otimes \quad C)$$

$\mathtt{symm}_{A,B}$: $$A \otimes B \xrightarrow{\sim} B \otimes A$$

$$A \quad \otimes \quad B \quad \multimap \quad B \quad \otimes \quad A$$

$\mathtt{unitl}_{A}$: $$(I \otimes A) \xrightarrow{\sim} A$$

$$(I \quad \otimes \quad A) \quad \multimap \quad A$$

$\mathtt{unitr}_{A}$: $$(A \otimes I) \xrightarrow{\sim} A$$

$$(A \quad \otimes \quad I) \quad \multimap \quad A$$

The application (or evaluation) morphisms

$$\mathtt{Ap}_{A,B} : (A \multimap B) \otimes A \longrightarrow B$$

have already been defined. For currying, given

$$\sigma : A \otimes B \multimap C$$

define

$$\Lambda(\sigma) : A \longrightarrow (B \multimap C)$$

by

$$\Lambda(\sigma) = \{\alpha^*(s) \mid s \in \sigma\}$$

where $\alpha : (M_A + M_B) + M_C \xrightarrow{\sim} M_A + (M_B + M_C)$ is the canonical isomorphism in **Set**.

Exercise 1.8 Verify that the above definitions work! *E.g.* verify the equations $\mathtt{Ap} \circ (\Lambda(\sigma) \otimes \mathrm{id}_A) \ = \ \sigma$:

$$\begin{tikzcd}
(A \multimap B) \otimes A \arrow[r, "\mathtt{Ap}"] & B \\
C \otimes A \arrow[u, "\Lambda(\sigma) \otimes \mathtt{id}_A"] \arrow[ur, "\sigma"'] &
\end{tikzcd}$$

and $\Lambda(\mathtt{Ap} \circ (\tau \otimes \mathtt{id}_A)) \; = \; \tau$ for $\tau : C \longrightarrow (A \multimap B)$.

Exercise 1.9 Prove that I is terminal in $\mathcal{G}$, i.e. for each A there is a unique morphism $t_A : A \longrightarrow I$.

This shows that $\mathcal{G}$ is really a model of Affine Logic, in which (unlike in Linear Logic proper) the Weakening rule is valid. Indeed, tensor has "projections":

$$A \otimes B \stackrel{\mathrm{id_A \otimes t_B}}{\longrightarrow} A \otimes I \stackrel{\mathrm{unitr}}{\stackrel{\sim}{\longrightarrow}} A.$$

Exercise 1.10 Given A, B define $A\&B$ by

$$\begin{array}{lcl} M_{A\&B} & = & M_A + M_B \\ \lambda_{A\&B} & = & [\lambda_A, \lambda_B] \\ P_{A\&B} & = & \{\mathtt{inl}^*(\mathtt{s}) \mid \mathtt{s} \in \mathtt{P_A}\} \cup \{\mathtt{inr}^*(\mathtt{t}) \mid \mathtt{t} \in \mathtt{P_B}\}. \end{array}$$

(Draw a picture of the game tree of $A\&B$; it is formed by gluing together the trees for A and B at the root. There is no overlap because we take the disjoint union of the alphabets.) Prove that $A\&B$ is the product of A and B in $\mathcal{G}$, i.e. define projections

$$A \stackrel{\mathtt{fst}}{\longleftarrow} A\&B \stackrel{\mathtt{snd}}{\longrightarrow} B$$

and pairing

$$\langle\ ,\ \rangle : \mathcal{G}(C, A) \times \mathcal{G}(C, B) \longrightarrow \mathcal{G}(C, A\&B)$$

and verify the equations

$$\begin{array}{lcl} \langle \sigma,\ \tau \rangle ; \mathtt{fst} & = & \sigma \\ \langle \sigma,\ \tau \rangle ; \mathtt{snd} & = & \tau \\ \langle \upsilon ; \mathtt{fst},\ \upsilon ; \mathtt{snd} \rangle & = & \upsilon \quad \text{for } \upsilon : C \longrightarrow A\&B \end{array}$$

Exercise 1.11 Try to define coproducts in $\mathcal{G}$. What is the problem?

Exercise 1.12 A strategy σ on A is *history-free* if it satisfies

- $sab,\ tac \in \sigma \;\Rightarrow\; b = c.$
- $sab,\ t \in \sigma,\ ta \in P_A \;\Rightarrow\; tab \in \sigma.$

Prove that $\mathtt{id}_A$, $\mathtt{assoc}_{A,B,C}$, $\mathtt{sym}_{A,B}$, $\mathtt{Ap}_{A,B}$, $\mathtt{unitl}_A$, $\mathtt{unitr}_A$, $\mathtt{fst}_{A,B}$, $\mathtt{snd}_{A,B}$ are all history-free; and that if σ and τ are history free so are $\sigma\ ;\ \tau$, $\sigma \otimes \tau$, and $\Lambda(\sigma)$. Conclude that the sub-category $\mathcal{G}^{\mathrm{hf}}$, of history-free strategies is also a model of IMLL. What about the pairing operation $\langle \sigma,\ \tau \rangle$? Does $\mathcal{G}^{\mathrm{hf}}$ have binary products?

2 Winning Strategies

As we have seen, deterministic strategies can be viewed as partial functions extended in time. This partiality is appropriate when we aim to model programming languages with general recursion, in which the possibility of non-termination arises. However we would also like to use game semantics to model logical systems satisfying Cut Elimination or Strong Normalization. We would therefore like to find a condition on strategies generalizing totality of functions. The obvious candidate is to require that at each stage of play, a strategy σ on A has some response to every possible move by opponent.

$$(\mathbf{tot}) \qquad s \in \sigma, sa \in \mathrm{P}_A \Rightarrow \exists b : \ sab \in \sigma$$

Call a strategy *total* if it satisfies this condition. However, totality as so defined does not suffice ; in particular, it is not closed under composition.

Exercise 2.1 Find games A, B, C and strategies $\sigma : A \to B$ and $\tau : B \to C$, such that

- σ and τ are total
- $\sigma; \tau$ is not total.

(Hint: use infinite chattering in B.)

The best analogy for understanding this fact is with the untyped λ-calculus: the class of strongly normalizing terms is not closed under application. Thus in the Tait/Girard method for proving strong normalization in various systems of typed λ-calculus, one introduces a stronger property which does ensure closure under application. The approach we will pursue with strategies can be seen as a semantic analogue of this idea.

The idea is to take *winning* strategies. Given a game A, define P_A^∞, the infinite plays over A, by

$$\mathrm{P}_A^\infty = \{s \in M_A^\omega \mid \mathrm{Pref}(s) \subseteq \mathrm{P}_A\}$$

(By $\mathrm{Pref}(s)$ we mean the set of *finite* prefixes.) Thus the infinite plays correspond exactly to the infinite branches of the game tree.

Now a set $W \subseteq \mathrm{P}_A^\infty$ can be interpreted as designating those infinite plays which are “wins” for Player. We say that σ is a *winning strategy* with respect to W (notation: $\sigma \models W$), if:

- σ is total
- $\{\mathrm{s} \in \mathrm{P}_A^\infty \mid \mathrm{Pref}(s) \subseteq \sigma\} \subseteq W$.

Thus σ is winning if at each finite stage when it is Player's turn to move it has a well defined response, and moreover every infinite play following σ is a win for Player.

We introduce an expanded of refined notion of game as a pair (A, W_A), where A is a game as before, and $W_A \subseteq \mathrm{P}^{\infty}_A$ is the designated set of winning infinite plays for Player. A winnining strategy for (A, W_A) is a strategy for A which is winning with respect to W_A.

We now extend the definitions of $\otimes$ and $\multimap$ to act on the winning set specifications:

$$\begin{array}{rcl} (A, W_A) \otimes (B, W_B) & = & (A \otimes B, W_{A \otimes B}) \\ (A, W_A) \multimap (B, W_B) & = & (A \multimap B, W_{A \multimap B}) \end{array}$$

where

$$\begin{array}{rcl} W_{A \otimes B} & = & \{ s \in \mathrm{P}^{\infty}_{A \otimes B} \mid s \restriction A \in \mathrm{P}_A \cup W_A \wedge s \restriction B \in \mathrm{P}_B \cup W_B \} \\ W_{A \multimap B} & = & \{ s \in \mathrm{P}^{\infty}_{A \multimap B} \mid s \restriction A \in \mathrm{P}_A \cup W_A \Rightarrow s \restriction B \in W_B \} \end{array}$$

Exercise 2.2 Why did we not define

$$W_{A \otimes B} \; = \; \{ s \in \mathrm{P}^{\infty}_{A \otimes B} \mid s \restriction A \in W_A \wedge s \restriction B \in W_B \}?$$

(Hint: consider the switching condition for $\otimes$).

In order to check that these definitions work well, we must show that the constructions on strategies we have introduced in order th model the proof rules of Linear Logic are well-defined with respect to winning strategies.

Exercise 2.3 Show that, for any (A, W_A), the copy-cat strategy $\mathtt{id}_A$ is a winning strategy.

Now we consider the crucial case of the Cut rule.

Suppose then that $\sigma : (A, W_A) \multimap (B, W_B)$ and $\tau : (B, W_B) \multimap (C, W_C)$. We want to prove that $\sigma ; \tau$ is total, i.e. that there can be no infinite chattering in B. Suppose for a contradiction that there is an infinite play

$$t = s b_0 b_1 \cdots \in \sigma \| \tau$$

with all moves after the finite prefix s in B. Then $t \restriction A, B$ is an infinite play in $A \multimap B$ following σ, while $t \restriction B, C$ is an infinite play in $B \multimap C$ following τ. Since σ is winning and $t \restriction A$ is finite, we must have $t \restriction B \in W_B$. But then since τ is winning we must have $t \restriction C \in W_C$, which is impossible since $t \restriction C$ is finite.

Exercise 2.4 Give a direct proof (not using proof by contradiction) that winning stratregies compose.

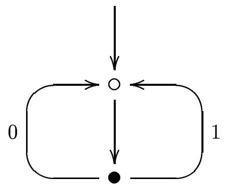
Exercise 2.5 *Prove that* $\mathtt{id}_A$, $\mathtt{assoc}_{A,B,C}$, $\mathtt{sym}_{A,B}$, $\mathtt{Ap}_{A,B}$, $\mathtt{unitl}_A$, $\mathtt{unitr}_A$, $\mathtt{fst}_{A,B}$, $\mathtt{snd}_{A,B}$ *are all winning strategies; and that if* σ *and* τ *are winning, so are* $\sigma\ ;\ \tau$, $\sigma \otimes \tau$, $\langle \sigma,\ \tau \rangle$, *and* $\Lambda(\sigma)$.

Exercise 2.6 Verify that the total strategies

$$\sigma : \mathbb{B} \to \mathbb{B}$$

correspond exactly to the total functions on the booleans.

Exercise 2.7 Consider a game of binary streams *Str*



with plays $*b_1 * b_2 * b_3 \ldots$, alternating between requests for data by Opponent and bits supplied by Player. Let W_{Str} be all infinite plays of this game.
Verify that the winning strategies on (Str, W_{Str}) correspond exactly to the infinite binary sequences. Verify that the winning strategies

$$\sigma : (Str, W_{Str}) \to (Str, W_{str})$$

induce functions which map infinite streams to infinite streams. Can you characterize exactly which functions on the domain

$$\{0,1\}^* \cup \{0,1\}^\omega$$

with the prefix ordering are induced by winning strategies?

3 Polymorphism

Our aim now is to use game semantics to give a model for polymorphism. We extend our notation for types with type variables $X, Y, \ldots$ and with second order quantifiers

$$\forall X.A$$

As a test case, we want our model to have the property that the interpretation it yields of the polymorphic (affine) booleans

$$\forall X.X \multimap (X \multimap X)$$

has only two elements, corresponding to the denotations of the terms

$$tt \stackrel{\text{def}}{\equiv} \Lambda X.\lambda x, y : X.x$$

$$ff \stackrel{\text{def}}{\equiv} \Lambda X.\lambda x, y : X.y$$

Firstly, we need some control over the *size* of the universe of types. To achieve this, we assume a non empty set $\mathcal{V}$ satisfying

$$\mathcal{V} + \mathcal{V} \subseteq \mathcal{V}$$

(for example take $\mathcal{V} = \{0, 1\}^*$).
Now we define a game $\mathcal{U}$ by:

$$\mathrm{M}_{\mathcal{U}} = \mathcal{V} + \mathcal{V}$$
$$\lambda_{\mathcal{U}} = [\mathbf{K}P, \mathbf{K}O]$$
$$\mathrm{P}_{\mathcal{U}} = \mathrm{M}_{\mathcal{U}}^{alt}.$$

(Here $\mathbf{K}P$ is the constant function valued at P.) We can define a partial order on games by:

$$A \trianglelefteq B \stackrel{\text{def}}{\equiv} \mathrm{M}_A \subseteq \mathrm{M}_B \ \wedge\ \lambda_A = \lambda_B \restriction \mathrm{M}_A \ \wedge\ \mathrm{P}_A \subseteq \mathrm{P}_B$$

Now define

$$\mathcal{G}_{\mathcal{U}} = \{A \in Obj(\mathcal{G}) \mid A \trianglelefteq \mathcal{U}\}$$

We define a *variable type* (in k variables) to be a function (monotone with respect to $\trianglelefteq$)

$$F : \mathcal{G}_{\mathcal{U}}^k \to \mathcal{G}_{\mathcal{U}}$$

Note that

$$A, B \in \mathcal{G}_{\mathcal{U}} \ \Rightarrow\ A \otimes B, A \multimap B \in \mathcal{G}_{\mathcal{U}}$$

(that was the point of having $\mathcal{V} + \mathcal{V} \subseteq \mathcal{V}$)

Exercise 3.1 (If you care about details) The above is not *quite* true. Amend the definition of $A \otimes B$, $A \multimap B$ slightly to make it true.

Thus variable types will be closed under $\otimes$ and $\multimap$. Given $F, G : \mathcal{G}_{\mathcal{U}}^k \to \mathcal{G}_{\mathcal{U}}$, we can define

$$\begin{aligned} F \otimes G(\vec{A}) &= F(\vec{A}) \otimes G(\vec{A}) \\ F \multimap G(\vec{A}) &= F(\vec{A}) \multimap G(\vec{A}) \end{aligned}$$

A *uniform strategy* σ on a variable type F is defined to be a strategy on $F(\vec{\mathcal{U}})$ such that, for all $\vec{A} \in \mathcal{G}_{\mathcal{U}}^k$, $\sigma_{\vec{A}}$ is a well-defined strategy on $F(\vec{A})$, where $\sigma_{\vec{A}}$ is defined inductively by

$$\sigma_{\vec{A}} = \{\epsilon\} \ \cup\ \{sab \mid s \in \sigma_{\vec{A}}, sa \in \mathrm{P}_{F(\vec{A})}, sab \in \sigma\}$$

(NB: in this notation, $\sigma = \sigma_{\vec{\mathcal{U}}}$).

Exercise 3.2 Show that the following properties hold for a uniform strategy σ on F:

(i) $\vec{A} \trianglelefteq \vec{B}$ (component-wise) $\Rightarrow \sigma_{\vec{A}} = \sigma_{\vec{B}} \cap P_{F(\vec{A})} \subseteq \sigma_{\vec{B}}$

(ii) if $(\vec{A}_i \mid i \in \mathrm{I})$ is a $\trianglelefteq$-directed family in $\mathcal{G}_{\mathcal{U}}^k$, then $\sigma_{\bigvee_{i\in\mathbf{I}}\vec{A}_i} = \bigcup_{i\in\mathbf{I}}\sigma_{\vec{A}_i}$ where $\bigvee_{i\in\mathbf{I}}\vec{A}_i$ is the directed join of the $\vec{A}_i$ (defined by component-wise union), and $\bigcup_{i\in\mathbf{I}}\sigma_{\vec{A}_i}$ is the directed union of the strategies $\sigma_{\vec{A}_i}$.

Our aim now is to show that, for each $k \in \omega$, we obtain a category $\mathcal{G}(k)$ with:

objects	:	variable types $F : \mathcal{G}_{\mathcal{U}}^k \to \mathcal{G}_{\mathcal{U}}$
morphisms	:	$\sigma : F \to G$ are uniform strategies σ on $F \multimap G$

Moreover $\mathcal{G}(k)$ is an autonomous category.

The idea is that all the structure is transferred pointwise from $\mathcal{G}$ to $\mathcal{G}(k)$. E.g if $\sigma : F \multimap G$, $\tau : G \multimap H$, then $\sigma;\tau : F \to H$ is given by $\sigma;\tau = \sigma_{\vec{\mathcal{U}}};\tau_{\vec{\mathcal{U}}}$.

Exercise 3.3 Check that $\sigma;\tau$ is a well-defined uniform strategy on $F \multimap H$.

Similarly, we define

$$\mathtt{id}_F = \mathtt{id}_{F(\vec{\mathcal{U}})}$$

$$\mathtt{Ap}_{F,G} = \mathtt{Ap}_{F(\vec{\mathcal{U}}),G(\vec{\mathcal{U}})}$$

etc.

Now we define a "base category" $\mathbb{B}$ with the objects $\mathcal{G}_{\mathcal{U}}^k$, $k \in \omega$, and $\trianglelefteq$-monotone functions as morphisms. For each object $\mathcal{G}_{\mathcal{U}}^k$ of $\mathbb{B}$, we have the autonomous category $\mathcal{G}(k)$. For each monotone

$$F = \langle F_1, \ldots, F_l \rangle : \mathcal{G}_{\mathcal{U}}^k \to \mathcal{G}_{\mathcal{U}}^l$$

we can define a functor

$$F^* : \mathcal{G}(l) \to \mathcal{G}(k)$$

by

$$F^*(G)(\vec{A}) = G(F(\vec{A}))$$

$$F^*(\sigma_{\vec{A}}) = \sigma_{F(\vec{A})}$$

Proposition 3.1 *The above defines a (strict) indexed autonomous category.*

At this point, we have enough structure to interpret types and terms with type variables. It remains to interpret the quantifiers. For notational simplicity, we shall focus on the case $\forall X.A(X)$ where X is the only type variable free in A.

Semantically A will be interpreted by a function $F : \mathcal{G}_{\mathcal{U}} \to \mathcal{G}_{\mathcal{U}}$. We must define a game $\Pi(F) \in \mathcal{G}_{\mathcal{U}}$ as the interpretation of $\forall X.A$

Corresponding to the polymorphic type inference rule $(\forall - \text{elim})\ \dfrac{\Gamma \vdash t : \forall X.A}{\Gamma \vdash t\{B\} : A[B/X]}$ we must define a uniform strategy

$$\pi : \mathbf{K}\Pi(F) \to F.$$

(Here $\mathbf{K}\Pi(F) : \mathcal{G}_{\mathcal{U}} \to \mathcal{G}_{\mathcal{U}}$ is the constant function valued at $\Pi(F)$. Note that $\mathbf{K} = t^*_{\mathcal{U}}$ where $t : \mathcal{U} \to \mathbf{1} = \mathcal{G}^0_{\mathcal{U}}$ is the map to the terminal object in $\mathbb{B}$.)

Corresponding to the type inference rule

$$(\forall - \text{intro}) \quad \frac{\Gamma \vdash t : A}{\Gamma \vdash \Lambda X.t : \forall X.A} \quad \text{if } X \notin \mathbf{FTV}(\Gamma)$$

we must prove the following universal property:

> for every $C \in \mathcal{G}_{\mathcal{U}}$ and uniform strategy $\sigma : \mathbf{K}C \to F$ there exists a unique strategy $\Lambda^2(\sigma) : C \to \Pi(F)$ such that

$$\begin{tikzcd}
\mathbf{K}\Pi(F) \arrow[r, "\pi"] & F \\
\mathbf{K}C \arrow[u, "\mathbf{K}\Lambda^2(\sigma)"] \arrow[ur, "\sigma"'] &
\end{tikzcd}$$

This says that there is an adjunction

$$\mathcal{G}_{\mathcal{U}} = \mathcal{G}_{\mathcal{U}}(0) \begin{tikzcd} {} \arrow[r, shift left, "t^*_{\mathcal{U}}"] \arrow[r, phantom, "\perp"] & {} \arrow[l, shift left, "\Pi(F)"] \end{tikzcd} \mathcal{G}_{\mathcal{U}}(1)$$

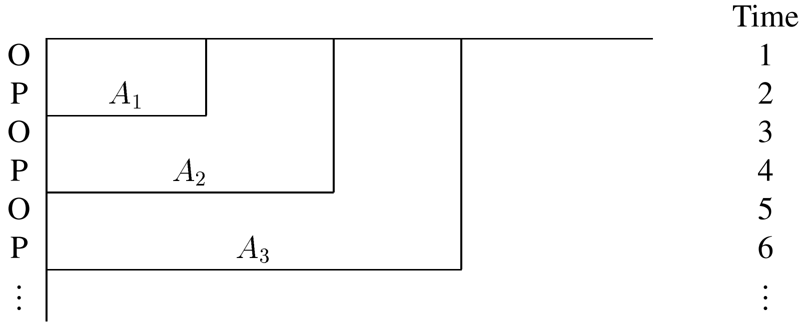
Furthermore, we must show that the Beck-Chevalley condition holds (see (Crole 1994)).

Remark 3.1 *More generally, we should show the existence of adjunctions*

$$\mathcal{G}_{\mathcal{U}} = \mathcal{G}_{\mathcal{U}}(k) \begin{tikzcd} {} \arrow[r, shift left, "p^*"] \arrow[r, phantom, "\perp"] & {} \arrow[l, shift left, "\Pi_k(F)"] \end{tikzcd} \mathcal{G}_{\mathcal{U}}(k+1)$$

where $p : \mathcal{G}^{k+1}_{\mathcal{U}} \to \mathcal{G}^{k}_{\mathcal{U}}$ is the projection function.

Now, how are we to construct the game $\Pi(F)$? Logically, Π is a second-order quantifier. Player must undertake to defend F at any instance $F(A)$, where A is specified by Opponent. If Opponent were to specify the entire instance A at the start of the game, this would in general require an infinite amount of information to be specified in a finite time, violating a basic continuity principle of computation ("Scott's axiom"). Instead we propose the metaphor of the "veil of ignorance" (*cf.* John Rawls, *A Theory of Justice*). That is, 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.



This intuition is captured by the following definition.

$$M_{\Pi(F)} = M_{F(\mathcal{U})}$$

$$\lambda_{\Pi(F)} = \lambda_{F(\mathcal{U})}$$

$P_{\Pi(F)}$ is defined inductively as follows:

$$\begin{array}{lcl} P_{\Pi(F)} & = & \{\epsilon\} \\ & \cup & \{sa \mid s \in P^{\text{even}}_{\Pi(F)} \ \wedge \ \exists A.sa \in P_{F(A)}\} \\ & \cup & \{sab \mid sa \in P^{\text{odd}}_{\Pi(F)} \ \wedge \ \forall A.sa \in P_{F(A)} \ \Rightarrow \ sab \in P_{F(A)}\} \end{array}$$

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 (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.

For the polymorphic projection

$$\Pi(F) \stackrel{\pi_A}{\to} F(A)$$

π_A plays copy-cat between $\Pi(F)$ and $F(A)$. This is uniform, witnessed by the "global copy-cat" $\mathtt{id}_{F(\mathcal{U})}$.

Why does this definition work? Consider the situation

$$\begin{array}{ccc} \Pi(F) & \to & F(A) \\ & & a \\ a & & \end{array}$$

At this stage, it is Opponent's turn to move, and of course there are many moves in $\Pi(F)$ which would not be valid in $F(A)$. However, Opponent in $\Pi(F)$ in contravariant (i.e negative) position must play as Player in $\Pi(F)$, and hence is constrained to respond to a only in a fashion which is valid in *every* instance in which a can be played, and which in particular is valid in $F(A)$. Hence Opponent's response can safely be copied back into $F(A)$.

Now for the universal property. Given uniform $\sigma : \mathbf{K}C \to F$, we define

$$\Lambda^2(\sigma) = \sigma : C \to \Pi(F)$$

That this is valid follows from the uniformity of σ so that at each stage its response must be valid in *any* instance that we might be in. It is then clear that

$$\mathbf{K}\Lambda^2(\sigma); \pi = \sigma; \mathtt{id}_{F\mathcal{U}} = \sigma$$

and hence that this definition fulfills the required properties.

Since we are interested in modeling IMLL2 (second order IMLL) we will refine our model with the notion of winning strategy, as explained in the previous section.

Firstly, we briefly indicate the additional structure required of a specification structure in order to get a model for IMLL2 in the refined category.

We assume that variable types are modeled by monotone functions $F : \mathcal{G}_{\mathcal{U}} \to \mathcal{G}_{\mathcal{U}}$ equipped with actions

$$F_A : PA \to P(FA)$$

for each $A \in \mathcal{G}_{\mathcal{U}}$.

Also there is an action:

$$\Pi_F : \mathbf{1} \to P(\Pi(F))$$

satisfying:

$(\forall - \text{elim})$ $\quad \Pi_F\{\pi_A\}\phi \quad (A \in \mathcal{G}_{\mathcal{U}}, \phi \in P(FA))$

$(\forall - \text{intro})$ $\quad (\forall A \in \mathcal{G}_{\mathcal{U}}, \psi \in PA.\, \phi\{\sigma_A\}F_A(\psi)) \Rightarrow \phi\{\Lambda^2(\sigma)\}\Pi_F.$

Now in the case of the specification structure W for winning strategies, we define:

$$\Pi_F = \{s \in \mathrm{P}^{\infty}_{\Pi(F)} \mid \forall A \in \mathcal{G}_{\mathcal{U}}, W \subseteq \mathrm{P}^{\infty}_{A}.\, s \in \mathrm{P}^{\infty}_{F(A)} \Rightarrow s \in F_A(W)\}.$$

Exercise 3.4 Verify that this satisfies ($\forall$-intro) and ($\forall$-elim).

Thus we have a game semantics for IMLL2 in which terms denote winning strategies. How good is this semantics? As a basic test, let us look at the type

$$\forall X.\, X \multimap (X \multimap X)$$

which we write as

$$\forall X.X_1 \multimap (X_2 \multimap X_3)$$

using indices to refer to the occurrences of X. What are the winning strategies for this type? Note that the first move must be in X_3. Because of the definition of Π, Player can only respond by playing the same move in a negative occurrence of X, i.e X_1 or X_2. Suppose Player responds in X_2:

$$\begin{array}{ccccc} \forall X.\, X_1 & \multimap & (X_2 & \multimap & X_3) \\ & & & & a \\ & & a & & \end{array}$$

At this point, by the switching condition Opponent must respond in X_2, say with a move b ; what can Player do next? If he were playing as the term $\Lambda X.\lambda x, y : X.y$, then he should copy b back to X_3. However there is another possiblity (pointed out by Sebastian Hunt): namely, Player can *play a in X_1*, and continue thereafter by playing copy-cat between X_1 and X_3. This certainly yields a winning strategy, but does not correspond to the denotation of any term.

To eliminate such undesirable possibilities, we introduce a constraint on strategies. Recall from Exercise 1.10 that a strategy is *history-free* if its response at any point depends only on the last move by Opponent: that is, if it satisfies:

$$sab \in \sigma, ta \in P_A \;\;\Rightarrow\;\; tab \in \sigma.$$

The history-free strategies suffice to model the multiplicatives and polymorphism, so we get a model $\mathcal{G}^{\mathrm{hf}}_{\mathcal{W}}$ of IMLL2.

Now consider again the situation

$$\begin{array}{ccccc} \forall X.\, X_1 & \multimap & (X_2 & \multimap & X_3) \\ & & & & a \\ & & a & & \\ & & b & & \end{array}$$

Player can only respond to b by copying b into X_3 if he is following a history-free strategy: the option of playing a in X_1 is not open to him, because a is not “visible” to him. Thus he can only proceed by

$$\begin{array}{ccccc} \forall X.\, X_1 & \multimap & (X_2 & \multimap & X_3) \\ & & & & a \\ & & a & & \\ & & b & & \\ & & & & b \end{array}$$

Moreover, Player must continue to play copy-cat between X_2 and X_3 ever thereafter, since the information available to him at each stage is only the move just played by Opponent.

Note also that Player must play in the same way, regardless of which move is initially made by Opponent. For example, suppose for a contradiction that Player responded to a_1, by copying it to X_1, and to a_2 by copying it to X_2. Now consider the situation:

$$\begin{array}{ccccccc}
\forall X. & X_1 & \multimap & (X_2 & \multimap & X_3) \\
 & & & & & a_1 \\
 & a_1 & & & & \\
 & b_1 & & & & \\
 & & & & & b_1 \\
 & & & & & a_2 \\
 & & & a_2 & &
\end{array}$$

Since Player is following a history-free strategy, he must *always* respond to a_2 by copying it to X_2; but the above position is clearly not valid, since there is an instance A with $P_A = \mathsf{Pref}\{a_1 b_1 a_2\}$ in which a_2 cannot be played as an initial move.

Thus we conclude that for our test case the model $\mathcal{G}^{\mathrm{hf}}_{\mathcal{W}}$ does indeed have the required property that the only strategies for the game

$$\forall X. X_1 \multimap (X_2 \multimap X_3)$$

are the denotations of the terms:

$$\begin{array}{cc}
\Lambda X.\lambda x, y : X.x & \Lambda X.\lambda x, y : X.y \\
\text{copycat between } X_1 \text{ and } X_3 & \text{copycat between } X_2 \text{ and } X_3.
\end{array}$$

Exercise 3.5 *Show that the only two strategies in $\mathcal{G}^{\mathrm{hf}}_{\mathcal{W}}$ for the game*

$$\forall X. (X \otimes X) \multimap (X \otimes X)$$

are those corresponding to the identity and the twist map.

Open problem For which class of (closed) types of IMLL2 do we get a "Full Completeness" result, i.e. that all strategies at that type in $\mathcal{G}^{\mathrm{hf}}_{\mathcal{W}}$ are definable in IMLL2?

4 Relational Parametricity

In this section, we investigate how the notion of relational parametricity can be adapted to the setting of games.

Firstly, we go back to the general level of Specification Structures. We use some notions due to Andy Pitts (1996).

Given $\phi, \psi \in PA$, we define:

$$\phi \leq \psi \quad \equiv \quad \phi\{\mathtt{id}_A\}\psi.$$

This is always a preorder by **(ss1)** and **(ss2)**. Say that the specification structure S is *posetal* if it is a partial order (i.e. antisymmetric). Now the notion of meet of properties $\bigwedge_{i \in I} \phi_i$ can be defined on PA. Say that S is *meet-closed* if it is posetal and each PA has all meets.

Now we define a notion of *relations* on games. We shall focus on binary relations. Say that R is a relation from A to B (notation: $R \subseteq A \times B$) if R is a non-empty subset $R \subseteq P_A \times P_B$ satisfying:

- $R(s,t) \;\; \Rightarrow \;\; |s| = |t|$.
- $R(sa,tb) \;\; \Rightarrow \;\; R(s,t)$.

(So R is a length-preserving non-empty prefixed closed subset).

We shall define a specification structure R on the product category $\mathcal{G} \times \mathcal{G}$ by taking $P(A,B)$ to be the set of relations $R \subseteq A \times B$. Given a relation $R \subseteq A \times B$, we lift it to a relation $\widehat{R}$ between strategies on A and strategies on B, by the following definition:

$$
\begin{array}{rcl}
\widehat{R}(\sigma,\tau) & \Longleftrightarrow & \forall s \in \sigma, t \in \tau.\, R(sa,ta') \\
& & \quad \Rightarrow \quad [(sa \in dom(\sigma) \;\; \Leftrightarrow \;\; ta' \in dom(\tau)) \\
& & \qquad\qquad \wedge\; sab \in \sigma, ta'b' \in \tau \;\; \Rightarrow \;\; R(sab, ta'b')]
\end{array}
$$

This definition is "logical relations extended in time"; it relativizes the usual clause:

$$
R(x,y) \quad \Rightarrow \quad [(fx\!\downarrow \;\Leftrightarrow\; gy\!\downarrow) \;\wedge\; (fx\!\downarrow, gy\!\downarrow \;\Rightarrow\; R(fx,gy))]
$$

to the context (previous history) s. It can also be seen as a form of bisimulation:

> "If σ and τ reach related states at P's turn to move, then one has a response iff the other does, and the states after the response are still related."

Also, if $R \subseteq A \times A'$ and $S \subseteq B \times B'$, then we define:

$$
\begin{array}{rcl}
R \otimes_{(A,\,A'),\,(B,\,B')} S & = & \{ \;\; (s,t) \in P_{A \otimes B} \times P_{A' \otimes B'} \;\; | \\
& & \quad R(s \upharpoonright A, t \upharpoonright A') \;\wedge\; S(s \upharpoonright B, t \upharpoonright B') \\
& & \quad \wedge\; \mathtt{out}^*(\mathtt{s}) = \mathtt{out}^*(\mathtt{t})) \;\; \}
\end{array}
$$

where $\mathtt{out} : \mathtt{M_A} + \mathtt{M_B} \rightarrow \{\mathtt{0},\mathtt{1}\}$ is given by:

$$
\mathtt{out} = [\mathbf{K}\mathtt{0}, \mathbf{K}\mathtt{1}]
$$

Similarly we define:

$$
\begin{array}{rcl}
R \multimap_{(A,\,A'),\,(B,\,B')} S & = & \{ \;\; (s,t) \in P_{A \multimap B} \times P_{A' \multimap B'} \;\; | \\
& & \quad R(s \upharpoonright A, t \upharpoonright A' \;\wedge\; S(s \upharpoonright B, t \upharpoonright B') \\
& & \quad \wedge\; out^*(s) = out^*(t)) \;\; \}
\end{array}
$$

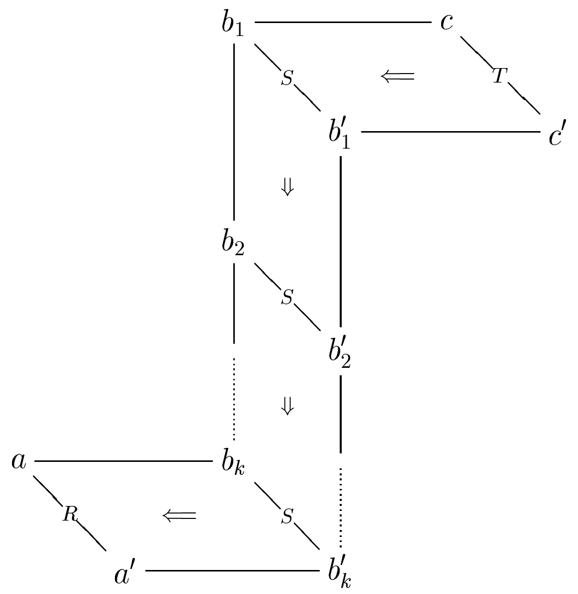
Now we define:

$$
R\{(\sigma,\tau)\}S \;\equiv\; \widehat{R \multimap S}(\sigma,\tau)
$$

Proposition 4.1 *This is a specification structure in $\mathcal{G} \times \mathcal{G}$. In particular,*

$$R\{(\sigma,\tau)\}S,\ S\{(\sigma',\tau')\}T \implies R\{(\sigma;\sigma',\tau;\tau')\}T$$

$$\begin{tikzcd}
A \arrow[rr, "\sigma"] \arrow[dr, dash, "R"] & & B \arrow[rr, "\tau"] \arrow[dr, dash, "S"] & & C \arrow[dr, dash, "T"] & \\
& A' \arrow[rr, "\sigma'"] & & B \arrow[rr, "\tau'"] & & C'
\end{tikzcd}$$



Exercise 4.1 Prove this! (The above "logical waterfall" diagram gives the idea of the proof.)

We shall in fact be more interested in "pulling back" this specification structure along the diagonal functor $\Delta : \mathcal{G} \to \mathcal{G} \times \mathcal{G}$. That is, we are interested in the category $\mathcal{G}_R$ with objects (A, R) where $R \subseteq A \times A$ and morphisms $\sigma : (A, R) \to (B, S)$ which are strategies $\sigma : A \to B$ such that $\widehat{R \multimap S}(\sigma, \sigma)$. We are also interested in the category $\mathcal{G}^{\mathrm{hf}}_{\mathcal{W}R}$ where we combine the winning strategy and relational structures, so that objects are (A, W_A, R_A), where W_A is a set of designated winning plays, and $R_A \subseteq A \times A$ is a relation and $\sigma : (A, W_A, R_A)$ is a strategy $\sigma : A \to B$ such that $W_A\{\sigma\}W_B \wedge R_A\{\sigma\}R_B$.

Now we build a model of IMLL2 by refining our previous model with this specification structure R. A variable type will now be a monotone function

$$F : (\mathcal{G}_{\mathcal{U}}, \trianglelefteq) \to (\mathcal{G}_{\mathcal{U}}, \trianglelefteq)$$

with an action

$$F_A : PA \to P(FA).$$

We assume that the specification structure is monotone, in the sense that:

$$A \trianglelefteq B \;\;\Rightarrow\;\; PA \subseteq PB$$

(this is easily seen to hold for R and W), and that

$$\begin{tikzcd} PA \arrow[r, hook] \arrow[d, "F_A"'] & PB \arrow[d, "F_B"] \\ P(FA) \arrow[r, hook] & P(FB) \end{tikzcd}$$

We also require that if $\phi \in PA$, $\psi \in PA$, $A \trianglelefteq A'$ and $B \trianglelefteq B'$, then

$$\phi\{F\}_{A,B}\psi \quad\Leftrightarrow\quad \phi\{F\}_{A',B'}\psi.$$

We further assume that the specification structure is meet-closed. Then we define:

$$\Pi_F \stackrel{\mathit{df}}{=} \bigwedge\{F_A(\phi) \mid A \in \mathcal{G}_{\mathcal{U}}, \phi \in PA\} = \bigwedge\{F_A(\phi) \mid \phi \in P\mathcal{U}\} \tag{4.1}$$

(This latter equality holds because of the above monotonicity properties).

The fact that ($\forall$-intro) and ($\forall$-elim) are satisfied then automatically holds because of the definition of Π_F as a meet.

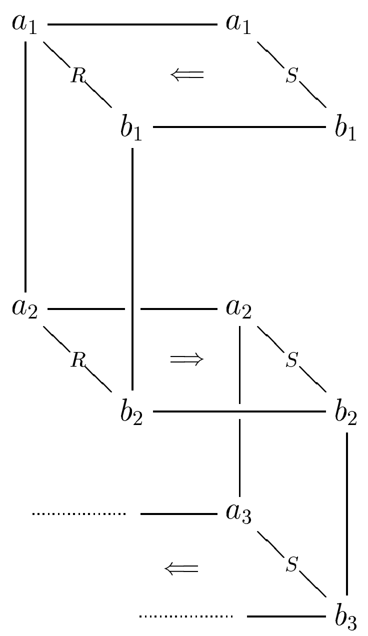
To apply this construction to R, we must show that it is meet-closed.

Firstly, we characterise the partial order on properties in R.

Proposition 4.2

$$\begin{array}{rcrcl} R \leq S & \Leftrightarrow & R^{\text{even}}(s,t) \wedge S(sa,tb) & \Rightarrow & R(sa,tb) \\ & \wedge & S^{\text{odd}}(s,t) \wedge R(sa,tb) & \Rightarrow & S(sa,tb). \end{array}$$

We can read this as: at O-moves $S \subseteq R$ and at P-moves $R \subseteq S$.



Proposition 4.3 $\bigwedge_{i \in I} R_i$ *is defined inductively by:*

$$\begin{array}{rcll} \bigwedge_{i \in I} R_i & = & & \{(\varepsilon, \varepsilon)\} \\ & \cup & \{(sa, ta') \mid & (s,t) \in \wedge_{i \in I} R_i^{\mathrm{even}} \wedge \\ & & & \exists i \in I. R_i(sa, ta')\} \\ & \cup & \{(sab, ta'b') \mid & (sa, ta') \in \wedge_{i \in I} R_i^{\mathrm{odd}} \wedge \\ & & & \forall i \in I. R_i(sa, ta') \;\; \Rightarrow \;\; R_i(sab, ta'b')\}. \end{array}$$

Note the similarity between this definition and that of $P_{\Pi(F)}$, which is in fact the unary case of the above, indexed over $P \subseteq^{\mathrm{nepref}} P_{F(U)}$.

Exercise 4.2

1. Verify these propositions.
2. For the specification structure $\mathcal{W}$, show that:
   - $V \leq W \;\; \Leftrightarrow \;\; V \subseteq W$.
   - $\bigwedge_{i \in I} W_i = \bigcap_{i \in I} W_i$.

Thus we obtain a model $\mathcal{G}^{\mathrm{hf}}_{\mathcal{W}R}$ of IMLL, incorporating both:

- the refinement to winning strategies
- a notion of "relational parametricity".

References

Abramsky, S., Gay, S. J., Nagarajan, R., (1996a) 'Specification structures and propositions-as-types for concurrency', In G. Birtwhistle and F. Moller, editors, Logics for Concurrency: Structure vs. Automata, *Proceedings of the VIIIth Banff Higher Order Workshop*, Lecture notes in Computer Science, pages 5–40, Springer Verlag.

Abramsky, S., Gay, S. J., Nagarajan, R., (1996b) 'Interaction categories and the Foundations of Typed Concurrent Programming', in Proceedings of the Nato Advanced Study Institute on Deductive Program Design, held in Marktoberdorf 1994, pages 35–113, Springer Verlag.

Abramsky, S., Jagadeesan, R., (1994) 'Games and full completeness for multiplicative linear logic', *Journal of Symbolic Logic, 59(2)*, 543 – 574. Also appeared as Technical Report 92/24 of the Department of Computing, Imperial College of Science, Technology and Medicine.

Abramsky, S., Jagadeesan, R., Malacaria, P., (1995) 'Full abstraction for PCF', Submitted for publication, ftp-able at `theory.doc.ic.ac.uk` in directory `papers/Malacaria`.

Abramsky, S., McCusker, G., (1995) 'Games for recursive types', In C. L. Hankin, I. C. Mackie, and R. Nagarajan, editors, Theory and Formal Methods of Computing 1994: Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods. Imperial College Press.

Abramsky, S., McCusker, G., (1995) 'Games and full abstraction for the lazy λ-calculus', in the LICS'95 proceedings.

Crole, R., (1994) 'Categories for Types', Cambridge University Press.

Danos, V., Herbelin, H., Regnier, L., (1996) 'Games and abstract machines', in the LICS'96 proceedings.

Girard, J.-Y., Lafont, Y., Taylor, P., (1989) 'Proofs and types', Cambridge University Press.

Girard, J.-Y., (1995) 'A survey of Linear Logic', in Advances in Linear Logic, ed. Y. Lafont, Cambridge University Press 1995.

Hyland, J. M. E., Ong, C.-H. L, (1995) 'On full abstraction for PCF: I, II, and III', submitted for publication, ftp-able at `theory.doc.ic.ac.uk` in directory `papers/Ong`.

McCusker, G., (1996a) 'Games and full abstraction for FPC', in the LICS'96 proceedings.

McCusker, G., (1996b) 'Games and full abstraction for a functional metalanguage with recursive types', Phd thesis, University of London, to appear.

Nickau, H., (1994) 'Hereditarily sequential functionals', *Proceedings of the Symposium on Logical Fondations of Computer Science: Logic at St. Petersburg*, Lecture notes in Computer Science. Springer Verlag.

Ong, C.-H. L., (1996) 'A semantic view of classical proofs', in the LICS'96 proceedings.

Pitts, A. M., (1996) 'Relational properties of domains', Information and Computation, vol. 127, no. 2, 66–90.