Introduction to Game Semantics

Samson Abramsky

Notation

X^* : the set of finite sequences (words, strings) over X

$$\frac{f : X \longrightarrow Y}{f^* : X^* \longrightarrow Y^*}$$

$|s|$: length of s. s_i i'th element of 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.

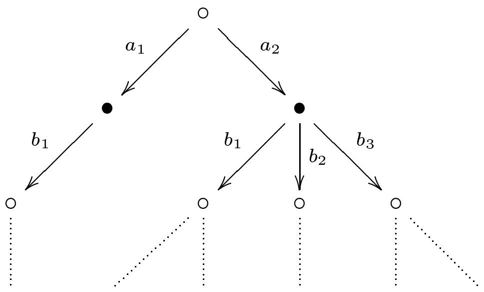
If $Y \subseteq X$ and $s \in X^*$, we write $s \upharpoonright Y$ for the sequence obtained by deleting all elements not in Y from s. In practice, we use this notation in the context where $X = Y + Z$, and by abuse of notation we take $s \upharpoonright Y \in Y^*$, *i.e.* we elide the use of injection functions.

We 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 $S = \mathrm{Pref}(S)$.

Games

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



- nodes ○ are Opponent positions
- nodes • are Player positions
- arcs are labelled with moves

Formal definition of games

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;
- $P_G \subseteq^{\text{nepref}} M_G^{\text{alt}}$, *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

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

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

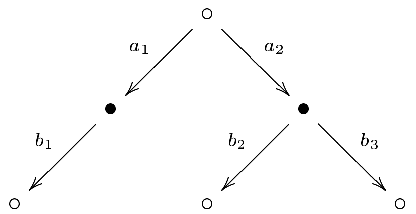
Example

The game

$$(\{a_1, a_2, b_1, b_2, b_3\}, \lambda, \{\epsilon, a_1, a_1b_1, a_2, a_2b_2, a_2b_3\})$$

$$\lambda \quad : \quad a_1, a_2 \mapsto O, \quad b_1, b_2, b_3 \mapsto P$$

represents the tree



Strategies

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;
⋮
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.

Strategies generalize functions

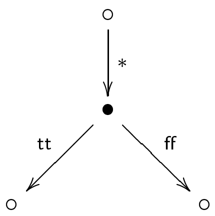
This can be seen as generalizing the notion of graph of a relation, *i.e.* of a set of ordered pairs, which can be read as a set of stimulus-response instructions. The generalization is that ordinary relations describe a single stimulus-response event only (giving rules for what the response to any given 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

Let $\mathbb{B}$ be the game

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

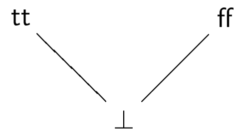


This game can be seen as representing the data type of booleans. The opening move $*$ is a request by Opponent for the data, which can be answered by either tt or ff by Player.

Strategies on $\mathbb{B}$

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

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:



Constructions on games

We will now describe some fundamental constructions on games.

Tensor Product

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

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

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'.

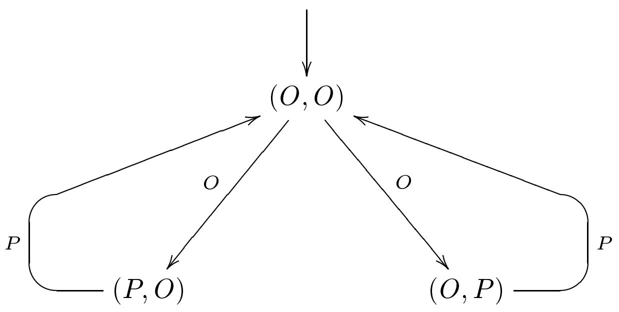
Switching Condition for Tensor Product

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 *(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.

State transition diagram for Tensor Product



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 $\ulcorner s \urcorner = (P, P)$).

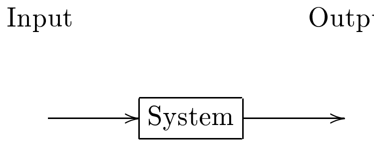
Linear Implication

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

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

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:



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

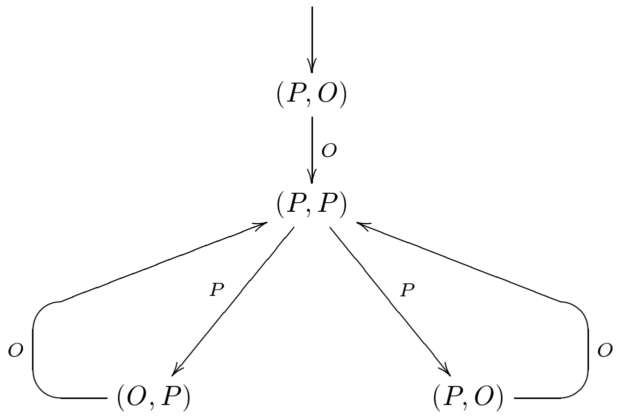
Note that $M^{\mathrm{alt}}_{A \multimap B}$, and hence $P_{A \multimap B}$, are in general quite different to $M^{\mathrm{alt}}_{A \otimes B}$, $P_{A \otimes B}$ respectively. In particular, the first move in $P_{A \multimap B}$ must always be in B, since the first move must be by Opponent, and all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

Switching Condition for Linear Implication

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

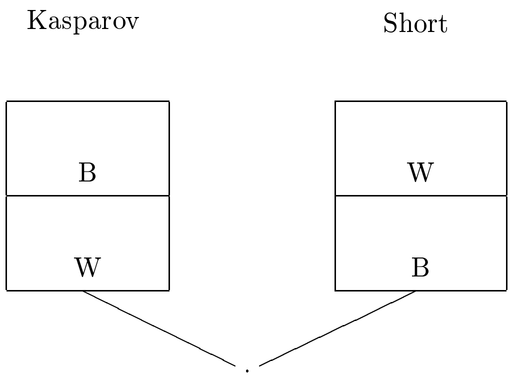
> If two consecutive moves are in different components, the first was by Opponent and the second by Player; so only Player can switch components.

This is supported by the following state-transition diagram:

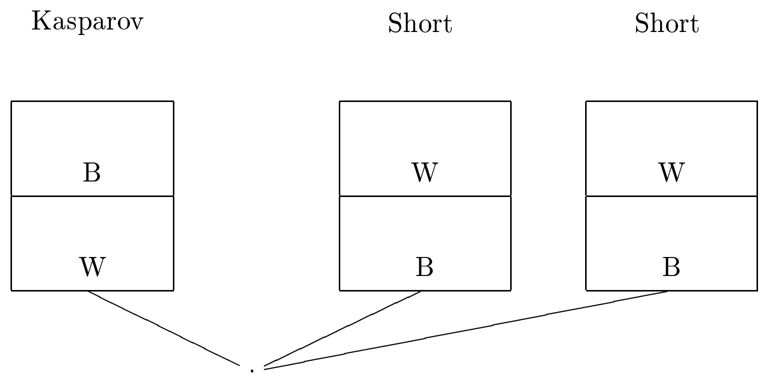


The Copy-Cat Strategy

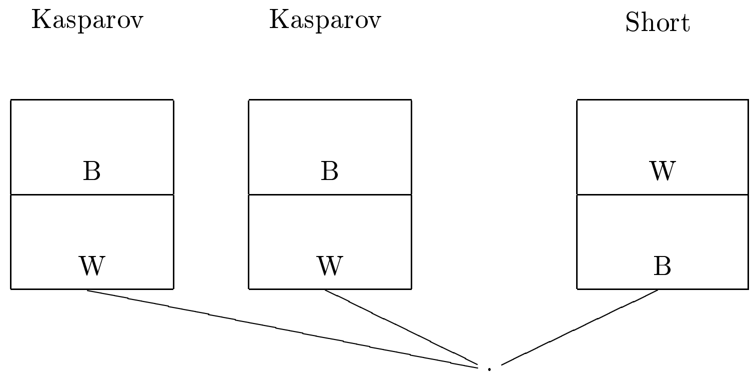
How to beat an International Grand-Master at chess by the power of Logic.



Does Copy-Cat still work here?



And here?



General definition of Copy-Cat strategy

Time	A	$\multimap$	A	
1			a_1	O
2	a_1			P
3	a_2			O
4			a_2	P
$\vdots$		$\vdots$		$\vdots$

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

We indicate such a strategy briefly by $A \multimap A$

Application (*Modus Ponens*)

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

This is the conjunction of two copy-cat strategies

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

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'.

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

The `Ap` strategy as a protocol for (linear) function application.

	(	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 output

ri — request input

id — input data

od — output data

The Category of Games $\mathcal{G}$

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

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

Composition as Interaction: The idea

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

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.

Formal Definition of Composition

'Parallel Composition + Hiding'

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

$$\sigma;\tau \; = \; (\sigma \parallel \tau)/B \; = \; \{s{\restriction}A, C \mid s \in \sigma \parallel \tau\}$$

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

(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 $s{\restriction}B, C$.)

An alternative definition of Composition

We give a more direct, 'computational' definition.

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

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:

1. $s \parallel\!\llcorner\, t$ O is to move in A.
2. $s \,\lrcorner\!\parallel t$ O is to move in C.
3. $s \setminus\!\!\setminus t$ σ to move.
4. $s \,/\!\!/\, t$ τ to move.

$$\begin{array}{rcll}
as \parallel\!\llcorner\, t & = & a(s \setminus\!\!\setminus t) & \\
\varepsilon \parallel\!\llcorner\, t & = & \varepsilon & \\
s \,\lrcorner\!\parallel ct & = & c(s \,/\!\!/\, t) & \\
s \,\lrcorner\!\parallel \varepsilon & = & \varepsilon & \\
as \setminus\!\!\setminus t & = & a(s \parallel\!\llcorner\, t) & (a \in M_A) \\
bs \setminus\!\!\setminus bt & = & s \,/\!\!/\, t & (b \in M_B) \\
s \,/\!\!/\, ct & = & c(s \,\lrcorner\!\parallel t) & (b \in M_C) \\
bs \,/\!\!/\, bt & = & s \setminus\!\!\setminus t & (a \in M_B)
\end{array}$$

We can then define

$$s;t \; = \; s \,\lrcorner\!\parallel t.$$

Proposition 2 *$\mathcal{G}$ is a category.*

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

Tensor structure of $\mathcal{G}$

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

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

$$\sigma \otimes \tau \quad = \quad \{ s \in P^{\text{even}}_{A \otimes A' \multimap B \otimes B'} \mid s \restriction A, B \in \sigma \quad \wedge \quad s \restriction A', B' \in \tau \}.$$

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

'Canonical isomorphisms' for monoidal structure

These arise as conjunctions of copy-cat strategies.

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

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

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

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

$$\texttt{unitl}_A : \qquad (I \otimes A) \xrightarrow{\sim} A$$

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

$$\texttt{u}nitr_A : \qquad (A \otimes I) \xrightarrow{\sim} A$$

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

Linear Implication

The application (or evaluation) morphisms

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

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**.

Winning Strategies

We would 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.

$$(tot) \qquad s \in \sigma, sa \in P_A \;\Rightarrow\; \exists b: \; sab \in \sigma$$

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 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.)

Given a game A, define P_A^∞, the infinite plays over A, by

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

(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 $W \subseteq P_A^\infty$ 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
- $\{\mathrm{s} \in P_A^\infty \mid \mathrm{Pref}(s) \subseteq \sigma\} \subseteq W$.

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 $W_A \subseteq P_A^\infty$ is the designated set of winning infinite plays for Player. A winning 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:

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

where

$$\begin{aligned} W_{A \otimes B} &= \{s \in P_{A \otimes B}^\infty \mid s \upharpoonright A \in P_A \cup W_A \wedge s \upharpoonright B \in P_B \cup W_B\} \\ W_{A \multimap B} &= \{s \in P_{A \multimap B}^\infty \mid s \upharpoonright A \in P_A \cup W_A \Rightarrow s \upharpoonright B \in W_B\} \end{aligned}$$

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

Exercise Show that, for any (A, W_A), the copy-cat strategy 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

$$t = sb_0b_1 \cdots \in \sigma \| \tau$$

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

The additive conjunction

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

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

$A\&B$ is the product of A and B in $\mathcal{G}$. We can define projections

$$A \xleftarrow{\mathsf{fst}} A\&B \xrightarrow{\mathsf{snd}} B$$

(Partial copy-cats) and pairing

$$\frac{\sigma : C \longrightarrow A \qquad \tau : C \longrightarrow B}{\langle \sigma, \tau \rangle : C \longrightarrow A \,\&\, B}$$

(Disjoint union)

Exercise Verify the equations

$$\begin{array}{rcl} \langle \sigma, \tau \rangle; \mathsf{fst} & = & \sigma \\ \langle \sigma, \tau \rangle; \mathsf{snd} & = & \tau \\ \langle \upsilon; \mathsf{fst},\ \upsilon; \mathsf{snd} \rangle & = & \upsilon \end{array}$$

for $\upsilon : C \longrightarrow A\&B$.

Interpreting the Linear exponential ! in $\mathcal{G}$

The resource sensitivity of games means that copying does not come for free; but it **can** be modelled explictly.

We begin with a simpler construction: the 'Tensor product of countably many copies of A', which we write as $\otimes^{\omega} A$:

- $M_{\otimes^{\omega} A} = \mathbb{N} \times M_A$, *i.e.* the disjoint union of countably many copies of M_A.
- $\lambda_{\otimes^{\omega} A}(n, a) = \lambda_A(a)$.
- $P_{\otimes^{\omega} A}$ is the set of all alternating sequences of moves in $M_{\otimes^{\omega} A}$ such that for all n, $s{\restriction}n \in P_A$.

(Switching conditions?)
Thus $\otimes^{\omega} A$ places no restriction on the order in which the copies are used.

Copying is comonoidal

The reason that we are not content with $\otimes^{\omega} A$ is that the various copies have distinct 'identities' via their indices $i \in \mathbb{N}$.

In any category with products, the diagonal — which expresses copyability — has an algebraic structure; it is a **cocommutative comonoid** — the dual of a commutative monoid.

That is, we have, for any object C:

(1) **Coassociativity**

$$\begin{tikzcd}
C \times (C \times C) \arrow[rr, "a_{C,C,C}"] & & (C \times C) \times C \\
C \times C \arrow[u, "\mathsf{id}_C \times \Delta"] & C \arrow[l, "\Delta"'] \arrow[r, "\Delta"] & C \times C \arrow[u, "\Delta \times \mathsf{id}_C"']
\end{tikzcd}$$

(2) **Counit**

$$\begin{tikzcd}
\top \times C & C \times C \arrow[l, "\top \times \mathsf{id}_C"'] \arrow[r, "\mathsf{id}_C \times \top"] & C \times \top \\
& C \arrow[ul, "l^{-1}"] \arrow[u, "\Delta"] \arrow[ur, "r^{-1}"'] &
\end{tikzcd}$$

(3) **Cocommutativity**

$$\begin{tikzcd}
C \arrow[r, "\Delta"] \arrow[dr, "\Delta"'] & C \times C \arrow[d, "s"] \\
& C \times C
\end{tikzcd}$$

Comonoids in monoidal categories

The notion of cocommutative comonoid (in future: **coalgebra** for short) makes sense in **any** symmetric monoidal category.

Let $(\mathbf{C}, \otimes, I, a, l, r, s)$ be a symmetric monoidal category. A **comonoid** in $\mathbf{C}$ is a triple (C, δ, ϵ) where C is an object, and $\delta : C \longrightarrow C \otimes C$ and $\epsilon : C \longrightarrow I$ are morphisms satisfying the commutative diagrams for Coassociativity, Counit, and Cocommutativity.

(N.B. coalgebraic structures are important in current mathematics, e.g. Hopf algebras, Quantum groups etc.)

Cofree coalgebras in $\mathcal{G}$

Let $A = (M_A, \lambda_A, P_A)$ be a game. Define $!A$ as follows.

- $M_{!A} = \mathbb{N} \times M_A$, *i.e.* the disjoint union of countably many copies of M_A.
- $\lambda_{!A}(n, a) = \lambda_A(a)$.
- $P_{!A}$ is the set of all alternating sequences of moves in $M_{!A}$ such that
  1. for all n, $s{\restriction}n \in P_A$.
  2. The first move (if any) in the $(n+1)$-th copy of A in s is made after the first move in the n-th, for all n.

Here, if $s \in M_{!A}^*$, $s{\restriction}n$ is defined by

$$\begin{array}{lcll} \epsilon{\restriction}n & = & \epsilon & \\ ((m,a)s){\restriction}n & = & a(s{\restriction}n), & m = n \\ & = & s{\restriction}n, & m \neq n. \end{array}$$

Thus a play in $!A$ must start in the 0'th copy; at some point, we may 'open' a new copy, which must be the 1st; and so on. At each stage, if n copies have been opened so far, the next copy to be opened must be the $(n+1)$-th.

Relation of $!A$ to $\otimes^\omega A$

For each permutation (*i.e.* bijective map) $\pi : \mathbb{N} \longrightarrow \mathbb{N}$, we can define a strategy $\sigma_\pi : \otimes^\omega A \longrightarrow \otimes^\omega A$ which for every $n \in \mathbb{N}$ plays copy-cat between the n'th copy of its codomain, and the $\pi(n)$'th copy of its domain. This strategy is an isomorphism in the category $\mathcal{G}$.

Proposition 3
- *There is a strategy $\eta : !A \longrightarrow \otimes^\omega A$ such that, for every permutation π on $\mathbb{N}$, $\eta ; \sigma_\pi = \eta$.*
- *Moreover, for every game C and strategy $\tau : C \longrightarrow \otimes^\omega A$ such that $\tau ; \sigma_\pi = \tau$ for every permutation π on $\mathbb{N}$, there is a unique strategy $\theta : C \longrightarrow !A$ such that $\tau = \theta ; \eta$.*

('$(!A, \eta)$ equalises $S(\omega)$'.)

The Copy-cat in the Hilbert Hotel

Comonoidal structure of $!A$:

We can define strategies $c : !A \longrightarrow !A \otimes !A$, and $w : !A \longrightarrow I$.

These will be used to interpret the Contraction and Weakening rules of Linear Logic:

$$\frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B} \qquad \frac{\Gamma \vdash B}{\Gamma, !A \vdash B}$$

$!A$ is a comonoid

E.g. Cocommutativity:

$$\begin{tikzcd} !A \arrow[r, "c"] \arrow[dr, "c"'] & !A \otimes !A \arrow[d, "s"] \\ & !A \otimes !A \end{tikzcd}$$

! is a 'comonad'

The definition of ! can be extended to a functor $! : \mathcal{G} \longrightarrow \mathcal{G}$.

We can also define, for each game A, strategies

$$d_A : !A \longrightarrow A, \qquad \delta_A : !A \longrightarrow !!A.$$

(For the latter, it is helpful to recall that $\mathbb{N} \times \mathbb{N}$ is in bijective correspondence with $\mathbb{N}$).

We can use d_A to give an interpretation of the Dereliction rule of Linear Logic:

$$\frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B}.$$

We can define for each A and B a strategy

$$m_{A,B} : !A \otimes !B \longrightarrow !(A \otimes B).$$

Question Is $m_{A,B}$ an isomorphism? Give an intuitive explanation of the difference between $!A \otimes !B$ and $!(A \otimes B)$.

We can use these constructions δ_A and $m_{A,B}$ to give an interpretation of the remaining rule for ! in Linear Logic, the Promotion rule:

$$\frac{!\Gamma \vdash A}{!\Gamma \vdash !A}$$

Exponential isomorphisms

$$\begin{array}{ccc} !(A \,\&\, B) & \cong & !A \otimes !B \\ !\top & \cong & I \end{array}$$

A Cartesian Closed Category of Games

Corresponding the the syntactic translation of $\supset$, $\wedge$ logic into Linear Logic using $\otimes$, $\multimap$, $\&$, $!$, we can build a **cartesian closed** category of games using the comonadic structure of $!$.

We build a category $\mathcal{K}_!(\mathcal{G})$ (the 'co-Kleisli category') as follows:

Objects: same as in $\mathcal{G}$.

Morphisms: $\mathcal{K}_!(A, B) = \mathcal{G}(!A, B)$.

Composition:

$$\frac{!A \xrightarrow{\sigma} B \qquad !B \xrightarrow{\tau} C}{!A \xrightarrow{\delta_A} !!A \xrightarrow{!\sigma} !B \xrightarrow{\tau} C}$$

Products: $A \times B = A \& B$

Exponentials: $A \Rightarrow B = !A \multimap B$.

Cartesian Closure

$$\begin{aligned} \mathcal{K}_!(A \times B, C) &= \mathcal{G}(!(A \& B), C) \\ &\cong \mathcal{G}(!A \otimes !B, C) \\ &\cong \mathcal{G}(!A, !B \multimap C) \\ &= \mathcal{K}_!(A, B \Rightarrow C). \end{aligned}$$

History-free strategies

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

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

Note that the first property says that what a history-free strategy does in a given position is determined only by the immediately preceding move by the Opponent, not by the previous history; while the second says that whether the strategy is defined or not similarly depends only on the immediately preceding move.

Each history-free strategy is determined by a partial function from Opponent moves to Player moves in a natural way. Given such a partial function $f : M_A \rightharpoonup M_B$, define

$$\sigma_f = \{\varepsilon\} \cup \{sab \mid s \in \sigma_f \ \wedge\ f(a) = b\}.$$

Conversely, given a history strategy σ, we can define

$$f(a) = b \ \equiv\ sab \in \sigma.$$

Then f is the least partial function such that

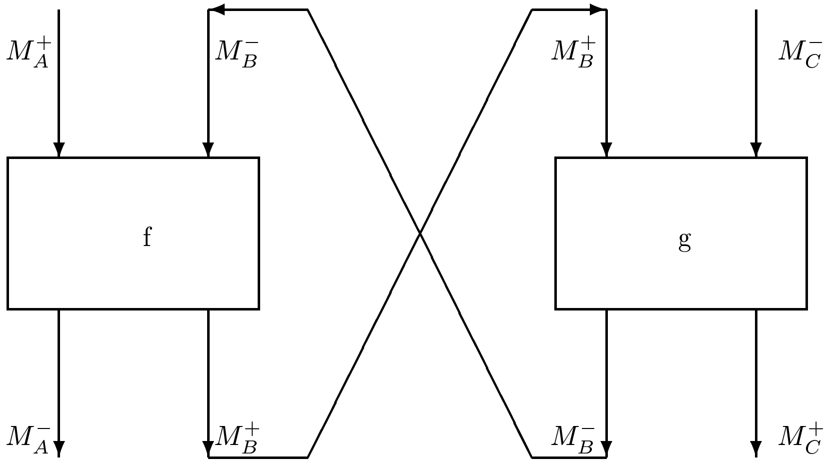
$$\sigma = \sigma_f.$$

History-free strategies suffice for Multiplicatives

- We can prove that $\mathbf{id}_A$, $\mathbf{assoc}_{A,B,C}$, $\mathbf{symm}_{A,B}$, $\mathbf{Ap}_{A,B}$ are all history-free by exhibiting partial functions which induce them.
- We can prove that if σ and τ are history-free, so are $\sigma;\tau$, $\sigma \otimes \tau$ and $\Lambda(\sigma)$ by exhibiting suitable operations on partial functions—and verifying that they work. For example, if f is a partial function inducing the strategy σ, and g is a partial function inducing the strategy τ, we can define a partial function $\mathsf{Comp}(f, g)$ and verify that this induces the strategy $\sigma;\tau$.
- It follows that the sub-category $\mathcal{G}^{\mathsf{hf}}$ of games and history-free strategies is a model of Multiplicative Linear Logic.

Composing History-Free Strategies

Say we have $\sigma_f : A \to B$, $\tau_g : B \to C$. We want to find h such that $\sigma_f;\tau_g = (\sigma;\tau)_h$.



We must give a formula for computing h according to these ideas. Firstly, we define functions

$$m_k : M_A^+ + M_C^- \rightharpoonup M_A^- + M_C^+$$

by

$$m_k = \pi^\star \circ ((f+g) \circ \mu)^k \circ (f+g) \circ \pi.$$

The idea is that m_k is the function which, when defined, feeds an input from M_A^+ or M_C^- exactly k times around the channels of the internal feedback loop and then exits from M_A^- or M_C^+.

The retraction

$$\pi : M_A + M_C \lhd M_A + M_B + M_B + M_C : \pi^\star$$

is defined by

$$\pi^\star = [\mathtt{inl}, 0, 0, \mathtt{inr}] \qquad \pi = [\mathsf{in}_1, \mathsf{in}_4]$$

and the "message exchange" function
$\mu : M_A^- + M_B^+ + M_B^- + M_C^+ \rightharpoonup M_A^+ + M_B^- + M_B^+ + M_C^-$ is defined by

$$\mu = 0 + [\mathsf{inr}, \mathsf{inl}] + 0$$

Here, 0 is the everywhere undefined partial function.

The Execution Formula

Finally, we define h by the **Execution Formula**:

$$h = \bigcup_{k \in \omega} m_k$$

It is a well-defined partial function because it is the union of a family of partial functions with pairwise disjoint domains of definition.

- History-free strategies do **not** suffice for the additives.
- We **can** interpret the exponentials with history-free strategies, but this necessitates some further complications.

Some achievements and applications of Game Semantics

Applications to Logic.

If we take the Curry-Howard isomorphism seriously, we should study the 'space of proofs' of a logical system as a mathematical object in its own right.

The usual syntactic presentation of logical systems tends to hide the real 'intrinsic' structure. (Cf. in Geometry, representation by coordinates *vs.* coordinate-free intrinsic or synthetic approaches.)

We are looking for a **syntax-free representation of proofs, and a characterization of the space of proofs.**

Traditional idea in logic of Soundness and Completeness is with respect to **provability**

$$\Gamma \vdash A \iff \Gamma \models A.$$

We are interested rather in the following notion. Given a category $\mathbf{C}$ interpreting a logical system $\mathcal{L}$, consider formulas A, B. The corresponding objects of $\mathbf{C}$ are $[\![A]\!]$, $[\![B]\!]$. The hom-set $\mathbf{C}([\![A]\!], [\![B]\!])$ will be used to interpret proofs of $A \vdash B$. We say that $\mathbf{C}$ is **fully complete** for $\mathcal{L}$ if **every** morphism

$$f : [\![A]\!] \longrightarrow [\![B]\!]$$

is the denotation of some proof of $A \vdash B$ in $\mathcal{L}$; and **fully and faithfully complete** if moreover the correspondence with proofs in normal form is $1-1$.

This notion of Full Completeness was first delineated, and a first such result proved, for a category of games and history-free strategies with respect to Multiplicative Linear Logic in

'Games and Full Completeness for Multiplicative Linear Logic', A. and R. Jagadeesan, JSL 1994.

There have been many subsequent papers on this theme, some using game semantics, others different models, applied to a range of logical systems.

Programming Language Semantics

The results on logic are closely linked with applications to Programming Language Semantics.

- Programming Languages can be formalized as typed λ-calculi.
- Game semantics can be used to give meaning to these calculi: the **types** of the programming language are modelled as **games**, and **programs** as **strategies**.
- Corresponding to Full Completeness for logics, we have **Full Abstraction** for Programming Languages.

Fully Abstract Models for Programming Languages

A classic problem was 'Full Abstraction for PCF' (simply-typed λ-calculus plus arithmetic and recursion).

The first syntax-independent descriptions of fully abstract models for PCF were achieved (in 1993) using game semantics. See

'Full Abtraction for PCF' by A., R. Jagadeesan and P. Malacaria
'On Full Abstraction for PCF' by M. Hyland and L. Ong.

Since then, the approach has been extended to many computational features, including: local state, reference types, control operators, non-determinism, probabilities, ...

Moreover, this has been done in a **systematic way**: the **presence** of computational features has been shown to correlate very precisely to the **relaxation** of various structural constraints on strategies (*e.g.* history-freeness). And there are **factorization theorems** which relate the different universes of strategies.

For an overview, see:

'Game Semantics' by A. and G. McCusker.

Algorithmic Game Semantics

Since Game Semantics is **concrete** — a strategy can be represented as a set of strings over an 'alphabet' of moves, *i.e.* a formal language — it can be represented algorithmically, *e.g.* strategies can be represented by automata. This means that semantic properties of programs can be analyzed algorithmically: giving a basis for **software model-checking and program analysis**.

Moreover, and crucially, the **compositional** nature of Game semantics means that this analysis can be done in a modular and compositional fashion — we can look at procedures, modules, objects and analyze their possible interactions with environments they might be embedded in.

See:

'Algorithmic Game Semantics: A Tutorial Introduction'