Game semantics and friends

Pierre-Louis Curien (CNRS – Université Paris Cité – Inria)

OPLSS 2022, Eugene, OR, 20-23/6/2022

Main source for the course: my Notes on game semantics (2006), available from curien.galene.org/papers

and bibliography therein
Some dual pairs in the world of programming

• memory *cell* (or location or register) versus its actual contents or *value*; in object-oriented style, record field names versus their values, method names versus their actual definition.

• *input* and *output*, or (in the language of proof theory) hypotheses and conclusions;

• *sending* and *receiving* messages (in process calculi);

• a *program* and its *context* (the libraries of your program environment – or the larger program of which the program under focus is a subpart, or a module); the programmer and the computer; two programs that call each other;

• call-by-name (CBN) and call-by-value (CBV).
Proofs as strategies

Proofs can be given a **dialogue-game** interpretation: a formula is tested through a dialogue between an **opponent** who doubts some formulas, and the **player** who justifies his proof step-by-step by exhibiting the rules he has used.

\[
\begin{align*}
    t_1 + S(t_2) &= S(t_1 + t_2) \\
    S(t_1 + S(t_2)) &= SS(t_1 + t_2) \\
    t_1 + S(t_2) &= SS(t_1 + t_2)
\end{align*}
\]
Untyped $\lambda$-calculus

The syntax of the untyped $\lambda$-calculus ($\lambda$-calculus for short) is given by:

$$M ::= x \mid MM \mid \lambda x.M,$$

where $x$ is called a variable, $M_1M_2$ is called an application, and $\lambda x.M$ is called an abstraction.

$\beta$-reduction:

$$\frac{(\lambda x.M)N \rightarrow M[x \leftarrow N]}{M \rightarrow M'} \quad \frac{N \rightarrow N'}{MN \rightarrow M'N} \quad \frac{M \rightarrow M'}{\lambda x.M \rightarrow \lambda x.M'}$$
Normal forms in the $\lambda$-calculus

Any $\lambda$-term has exactly one of the following two forms:

- head normal form $(n \geq 1, p \geq 1)$:
  \[ \lambda x_1 \cdots x_n . x M_1 \cdots M_p \]

- head redex $(n \geq 0, p \geq 1)$:
  \[ \lambda x_1 \cdots x_n . (\lambda x . M) M_1 \cdots M_p \]

Note that a normal form can be only of the first form, and recursively so.
Böhm trees

$$\omega(M) = \begin{cases} 
\Omega & \text{if } M = \lambda x_1 \cdots x_n. (\lambda x.P)M_1 \cdots M_p \\
\lambda x_1 \cdots x_n.x \omega(M_1) \cdots \omega(M_p) & \text{if } M = \lambda x_1 \cdots x_n.x M_1 \cdots M_p, 
\end{cases}$$

Then the Böhm tree of a term is the (possibly infinite) least upper bound of all \(\omega(N)\), for all \(N\) such that \(M \rightarrow^* N\).

The Böhm tree of a normalisable term is its normal form.
Böhm trees as strategies

\[
\begin{array}{c}
  \lambda x_1 \cdots x_n.x \ M_1 \cdots (\lambda y_1 \cdots y_p.y \ N_1 \cdots N_p) \cdots M_n \\
\end{array}
\]

\[
\begin{array}{c}
  \lambda x_1 \cdots x_n.x M_1 \cdots (\lambda y_1 \cdots y_p.y N_1 \cdots N_p) \cdots M_n \\
\end{array}
\]
Simply typed $\lambda$-calculus

$$\sigma ::= C \mid \sigma \to \sigma \quad (C \text{ base type})$$

Thus every type writes uniquely as $\sigma_1 \to \cdots \to \sigma_n \to C$.

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma. M : \sigma \to \tau}$$

$$\frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}$$

Böhm trees associated to simply typable terms are finite (see Silvia's lectures!).
We restrict ourselves to the simply typed setting, and impose that

- each occurrence of a variable must appear in a context where it is applied to all its arguments,

- and in each sequence of abstractions $\lambda \vec{x}.M$ the number of parameters $x_1, \ldots, x_n$ is exactly the number of arguments $N_1, \ldots, N_n$ which the term $\lambda \vec{x}.M$ can accept according to its type.
Executing Böhm trees (the abstract machine)

Terms \( M ::= (\lambda \vec{x}. W) \)

Environments \( e ::= nil \mid B \cdot e \)

Code \( W ::= y \vec{M} \)

Frames \( B ::= \langle \vec{z} \leftarrow \vec{M} \rangle[e] \).

We split \( M \) in this way even if \( \vec{x} \) is empty: in this case we write \( M = (W) \) and \( W = y \vec{N} \). We also call \( W \) a body.

\[
(K) \quad (x \vec{M})[e] \rightarrow W[\langle \vec{z} \leftarrow \vec{M} \rangle[e] \cdot e_i]
\]

where \[
\begin{aligned}
e &= B_0 \cdot \cdots \cdot B_n \\
x &= x_{ij} \\
B_i &= \langle x_i \leftarrow N_i \rangle[e_i] \\
N_{ij} &= (\lambda \vec{z}. W)
\end{aligned}
\]
Executing Böhm trees (illustration)

2' \( u(\lambda x.u(\lambda y.x)) \) \( \langle u \leftarrow (\lambda r.r(r(z)))) \rangle[ ] = \rho_0 \)

3' \( r(r(z)) \) \( \langle r \leftarrow (\lambda x.u(\lambda y.x)) \rangle[\rho_0] = \rho_1 \)

4' \( u(\lambda y.x) \) \( \langle x \leftarrow (r(z)) \rangle[\rho_1]\cdot(\rho_0 = \langle u \leftarrow (\lambda r.r(r(z)))) \rangle[ ] ) = \rho_2 \)

5' \( r(r(z)) \) \( \langle r \leftarrow (\lambda y.x) \rangle[\rho_2] = \rho_3 \)

6' \( x \) \( \langle y \leftarrow (r(z)) \rangle[\rho_3]\cdot(\rho_2 = \langle x \leftarrow (r(z)) \rangle[\rho_1]\cdot\rho_0 ) = \rho_4 \)

7' \( r(z) \) \( \langle \rangle[\rho_4]\cdot(\rho_1 = \langle r \leftarrow (\lambda x.u(\lambda y.x)) \rangle[\rho_0]) = \rho_5 \)

8' \( u(\lambda y.x) \) \( \langle x \leftarrow (z) \rangle[\rho_5]\cdot(\rho_0 = \langle u \leftarrow (\lambda r.r(r(z)))) \rangle[ ] ) = \rho_6 \)

9' \( r(r(z)) \) \( \langle r \leftarrow (\lambda y.x) \rangle[\rho_6] = \rho_7 \)

10' \( x \) \( \langle y \leftarrow (r(z)) \rangle[\rho_7]\cdot(\rho_6 = \langle x \leftarrow (z) \rangle[\rho_5]\cdot\rho_0 ) = \rho_8 \)

11' \( z \) \( \langle \rangle[\rho_8]\cdot\rho_5 \)
The language PCF

PCF is simply typed $\lambda$-calculus, with the following constants

\[ n \quad : \quad \text{Nat} \quad \quad (n \in \omega) \]
\[ T, F \quad : \quad \text{Bool} \]
\[ \text{succ}, \text{pred} \quad : \quad \text{Nat} \rightarrow \text{Nat} \]
\[ \text{zero?} \quad : \quad \text{Nat} \rightarrow \text{Bool} \]
\[ \text{if then else} \quad : \quad \text{Bool} \rightarrow \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat} \]
\[ \text{if then else} \quad : \quad \text{Bool} \rightarrow \text{Bool} \rightarrow \text{Bool} \rightarrow \text{Bool} \]

\[ \Omega \quad : \quad \sigma \quad \quad \text{for all } \sigma \]
\[ Y \quad : \quad (\sigma \rightarrow \sigma) \rightarrow \sigma \quad \quad \text{for all } \sigma \]
Operational semantics of PCF

\[(\lambda x. M)N \rightarrow M[x \leftarrow N]\]  
\[YM \rightarrow M(YM)\]

\[\text{zero?}(0) \rightarrow T\]  
\[\text{zero?}(n + 1) \rightarrow F\]

\[\text{succ}(n) \rightarrow n + 1\]  
\[\text{pred}(n + 1) \rightarrow n\]

\[\text{if } T \text{ then } N \text{ else } P \rightarrow N\]  
\[\text{if } F \text{ then } N \text{ else } P \rightarrow P\]

\[M \rightarrow M'\]  
\[\frac{M}{MN} \rightarrow M'N\]

\[\frac{M \rightarrow M'}{\text{if } M \text{ then } N \text{ else } P \rightarrow \text{if } M' \text{ then } N \text{ else } P}\]

\[\frac{M \rightarrow M'}{f(M) \rightarrow f(M')}\]

(for \(f \in \{\text{succ, pred, zero?}\}\))
PCF Böhm trees

\[ M ::= \lambda \vec{x} : \vec{\sigma}.W \quad W ::= v \mid \text{case } yM_1 \cdots M_n [v_1 \to W_1, \ldots, v_k \to W_k] \]

\[ v : C \]
\[ \Gamma \vdash v : C \]

\[ \Gamma \cdot y = \sigma_1 \to \cdots \to \sigma_n \to C \quad v_1, \ldots, v_k : C \]
\[ \Gamma \vdash M_1 : \sigma_1 \quad \cdots \quad \Gamma \vdash M_n : \sigma_n \quad \Gamma \vdash W_1 : C_1 \quad \cdots \quad \Gamma \vdash W_k : C_1 \]
\[ \Gamma \vdash \text{case } yM_1 \cdots M_n [v_1 \to W_1, \ldots, v_k \to W_k] : C_1 \]

\[ \Gamma, \vec{x} : \vec{\sigma} \vdash W : C \]
\[ \Gamma \vdash (\lambda \vec{x} : \vec{\sigma}.W) : \vec{\sigma} \to C \]
PCF Böhm trees as strategies

\[
\begin{array}{cccc}
M_1 & \cdots & M_p & v_1 \to W_1 & \cdots & v_k \to W_k \\
\hline
\lambda x. \text{case } y
\end{array}
\]
HO games

Game semantics arose as such in the early 1990’s from parallel works of

- Abramsky, Jagadeesan, Malacaria (AJM)
- Hyland and Ong (HO)

A fore-runner was sequential algorithms (Berry-Curien) (late 1970’s).

In this course, we focus on HO.
Entering the arena!

We start with the simplest kind of data type. The arena \texttt{nat} for natural numbers has the following moves: one (initial) \( \bigcirc \) move denoted \( q \), and a \( \bigcirc \) move \( n \) for each natural number \( n \).

\[
\left\{ \begin{array}{c}
\{ \epsilon \} \\
\{ \epsilon, qn \}
\end{array} \right\} \left\{ \begin{array}{c}
\bot \\
n
\end{array} \right\}
\]
Product of arenas

The arena $\text{nat} \times \text{nat}$ is made of two disjoints copies of $\text{nat}$.

$$\begin{align*}
\{\epsilon\} & \quad \{\perp, \perp\} \\
\{\epsilon, q_1m_1\} & \quad (m, \perp) \\
\{\epsilon, q_2n_2\} & \quad (\perp, n) \\
\{\epsilon, q_1m_1, q_2n_2\} & \quad (m, n)
\end{align*}$$
Function types \((\text{nat}_1 \rightarrow \text{nat}_\epsilon)\)

\[
q_\epsilon q_1 \left\{ \begin{array}{c}
0_1 3_\epsilon \\
3_1 6_\epsilon
\end{array} \right. \\
\lambda x. \text{case } x \left\{ \begin{array}{c}
0 \rightarrow 3 \\
3 \rightarrow 6
\end{array} \right.
\]

Note the inversion of polarity for \(\text{nat}_1\). Also, \(q_\epsilon \vdash q_1\).

Interaction with \(\{q_1 0\}\) converges, interaction with \(\{q_1 2\}\) diverges.
Plays and (deterministic) strategies

The tree $q_e q_1 \{ 0_1 3_\epsilon, 1_1 4_\epsilon \}$ can equivalently be described by the set of its even-length branches, which are called plays:

$$\{ \epsilon, q_e q_1, q_e q_1 0_1 3_\epsilon, q_e q_1 1_1 4_\epsilon \}$$

A strategy is a set of even-length plays. Note that plays start with $O$ and alternate between $O$ and $P$.

Here we deal with deterministic strategies $\sigma$:

if $pv, pw \in \sigma$, then $v = w$.

The player knows how to react to each opponent’s move: “my head variable is”.
Just a paraphrase of syntax?

Mathematics is full of useful equivalent presentations: descriptive / analytical geometry, matrices / linear maps, etc...

Sign of good syntax: Böhm trees!

In fact, the correspondence is an injection: games offer a wider picture (cf. IA).
Stuttering

\[ \lambda x. \text{case } x [4 \rightarrow \text{case } x [3 \rightarrow 2]] \text{ as strategy contains } q_\epsilon q_1 4_1 q_1 3_1 2_\epsilon. \]

Stuttering strategies \textit{do not} exist in the earlier model of sequential algorithms of PCF (Berry-Curien, late 1970).
A higher-order type

\[(\text{nat}_{11} \rightarrow \text{nat}_1) \rightarrow \text{nat}_\epsilon\]

\[h = \lambda f. \text{case } f(3)[4 \rightarrow 7, 6 \rightarrow 9]\]

As a strategy:

\[\lambda f. \text{case } f \left\{ \begin{array}{ll} (3) & q_1 3_11 \\ 4 \rightarrow 7 & q_1 7_\epsilon \\ 6 \rightarrow 9 & 6_1 9_\epsilon \end{array} \right.\]
Pointers

\[ K_{\text{Kierstead}}_1 = \lambda f. \text{case } f(\lambda x. \text{case } f(\lambda y. \text{case } x)) \]
\[ K_{\text{Kierstead}}_2 = \lambda f. \text{case } f(\lambda x. \text{case } f(\lambda y. \text{case } y)) \]

\[
\begin{align*}
q_\epsilon q_1 \quad q_{11} q_{111} \quad T_{111} T_{11} \\
q_1 q_{111} \quad T_1 T_{11} \\
T_1 T_\epsilon \\
F_1 F_{11} \\
F_1 F_\epsilon
\end{align*}
\]

(type \(((\text{bool}_{111} \to \text{bool}_{11}) \to \text{bool}_1) \to \text{bool}_\epsilon\))
\[
Kierstead_1 = q_\epsilon[q_1, ^0] = \begin{cases} 
q_{11}[q_1, \leftarrow] & q_{11}[q_{111}, \downarrow] \\
T_1[T_\epsilon, \leftarrow] & T_1[T_{11}, \leftarrow] \\
F_1[F_\epsilon, \leftarrow] & F_1[F_{11}, \leftarrow] 
\end{cases} \begin{cases} 
q_{11}[q_{111}, \downarrow] & T_{111}[T_{11}, \downarrow] \\
T_1[T_{11}, \leftarrow] & F_{111}[F_{11}, \leftarrow] \\
\end{cases}
\]

\[
Kierstead_2 = q_\epsilon[q_1, ^0] = \begin{cases} 
q_{11}[q_1, \leftarrow] & q_{11}[q_{111}, \leftarrow] \\
T_1[T_\epsilon, \leftarrow] & T_1[T_{11}, \leftarrow] \\
F_1[F_\epsilon, \leftarrow] & F_1[F_{11}, \leftarrow] 
\end{cases} \begin{cases} 
q_{11}[q_{111}, \leftarrow] & T_{111}[T_{11}, \leftarrow] \\
T_1[T_{11}, \leftarrow] & F_{111}[F_{11}, \leftarrow] \\
\end{cases}
\]
Inserting the implicit pointers in our previous examples

\[ h = \lambda f. \text{case } f(3)[4 \rightarrow 7, 6 \rightarrow 9] \quad \lambda x. \text{case } x[0 \rightarrow 3, 3 \rightarrow 6] \]

\[ q_\epsilon q_1 \begin{cases} q_{11}3_{11} \\ 4_17_\epsilon \\ 6_19_\epsilon \end{cases} \quad q_1 q_{11} \begin{cases} 0_{11}3_1 \\ 3_{11}6_1 \end{cases} \]

\[ q_\epsilon[q_1, \leftarrow 0] \begin{cases} q_{11}[3_{11}, \leftarrow 0] \\ 4_1[7_\epsilon, \leftarrow 1] \quad 6_1[9_\epsilon, \leftarrow 1] \end{cases} \quad q_1[q_{11}, \leftarrow 0] \begin{cases} 0_{11}[3_1, \leftarrow 1] \\ 3_{11}[6_1, \leftarrow 1] \end{cases} \]
Game abstract machine for PCF Böhm trees

- $p'$ determined from $(p - 1)$ by \[
\begin{cases}
\text{the strategy for } n \text{ even} \\
\text{the counter-strategy for } n \text{ odd}
\end{cases}
\]

- If $p'$ points to $m$, then play $p$ over $m'$
Game abstract machine for PCF Böhm trees

\[ q_\epsilon[q_1, 0] \begin{cases} 
  q_{11}[3_{11}, 0] \\
  4_1[7_\epsilon, 1] \\
  6_1[9_\epsilon, 1] 
\end{cases} \quad q_1[q_1, 0] \begin{cases} 
  0_{11}[3_1, 1] \\
  3_{11}[6_1, 1] \end{cases} \quad \text{interaction}

\begin{align*}
\langle q_\epsilon, 1\rangle[q_1, 0] & \begin{cases} 
  2' \\
  \langle q_{11}, 3\rangle[3_{11}, 0] \\
  \langle 6_1, 5\rangle[9_\epsilon, 1] 
\end{cases} \\
\langle q_1, 2\rangle[q_{11}, 0] & \begin{cases} 
  3' \\
  \langle 3_{11}, 4\rangle[6_1, 1] 
\end{cases}
\end{align*}
Executing Kierstead against

\[
\lambda g. \text{case } g(\text{case } gT \ [T \rightarrow T, F \rightarrow F]) \ [T \rightarrow F, F \rightarrow T]
\]

\[
q_1[q_{11}, \leftrightarrow] \begin{cases} 
q_{111}[q_{11}, 1] \begin{cases} 
q_{111}[F_{111}, 0] \\
T_{11}[T_{111}, 1] \\
F_{11}[F_{111}, 1]
\end{cases} \\
T_{11}[F_1, 1] \\
F_{11}[T_1, 1]
\end{cases}
\]

\[
Kierstead_1 = q_\epsilon[q_1, 0]
\begin{cases} 
q_{11}[q_1, 1] \begin{cases} 
q_{11}[F_{11}, 1] \\
T_1[T_{11}, 1] \\
F_1[F_{11}, 1]
\end{cases} \\
T_1[T_\epsilon, 1] \\
F_1[F_\epsilon, 1]
\end{cases}
\]
\[\langle q_\epsilon, 1 \rangle[q_1, 0] \left\{ \begin{array}{l}
\langle q_{11}, 3 \rangle[q_1, 1] \\
\langle q_{11}, 7 \rangle[q_1, 1] \\
\langle T_1, 19 \rangle[T_\epsilon, 1]
\end{array} \right\} \left\{ \begin{array}{l}
\langle q_{11}, 5 \rangle[q_{111}, 1] \\
\langle q_{11}, 9 \rangle[q_{111}, 1] \\
\langle T_1, 13 \rangle[T_{11}, 1]
\end{array} \right\}
\left\{ \begin{array}{l}
\langle T_{111}, 15 \rangle[T_{11}, 1] \\
\langle F_1, 17 \rangle[F_{11}, 1] \\
\langle F_{111}, 11 \rangle[F_{11}, 1]
\end{array} \right\}
\left\{ \begin{array}{l}
\langle q_{111}, 10 \rangle[F_{111}, 0] \\
\langle T_{11}, 14 \rangle[T_{111}, 1]
\end{array} \right\}
\left\{ \begin{array}{l}
\langle F_{11}, 18 \rangle[T_1, 1] \\
\langle T_{11}, 16 \rangle[F_1, 1]
\end{array} \right\}
\left\{ \begin{array}{l}
\langle F_{11}, 12 \rangle[T_1, 1]
\end{array} \right\} \]
Well-bracketing

This strategy of \((\text{nat}_{11} \times \text{nat}_{12} \rightarrow \text{nat}_1) \rightarrow \text{nat}_\epsilon\) is not the interpretation of a PCF term:

\[
\begin{align*}
q_\epsilon q_1 & \begin{cases}
q_{11}0_\epsilon \\
q_{12}1_\epsilon \\
n_1(n + 2)_\epsilon
\end{cases}
\end{align*}
\]

The initial question \(q_\epsilon\) may be answered while \(q_1\) and \(q_{11}\) are still open.
Strong (or partial) evaluation

\[
\begin{align*}
q_\epsilon q_2 \left\{ \begin{array}{c}
6_2 q_1 \\
3_2 q_1
\end{array} \right\} \\
\left\{ \begin{array}{c}
4_1 8_\epsilon \\
2_1 1_\epsilon \\
4_1 5_\epsilon \\
1_1 6_\epsilon
\end{array} \right\}
\end{align*}
\]
against \( q_1 4_1 \) yields \( q_\epsilon q_2 \left\{ \begin{array}{c}
6_2 8_\epsilon \\
3_2 5_\epsilon
\end{array} \right\} \)

\[
(\lambda xy. \text{case } y \ [6 \rightarrow \text{case } x \ [\ldots], \ldots])4 \rightarrow \lambda y. \text{case } y \ [\ldots]
\]

Lazy, stream-like loop of evaluation.
On the form of the plays involved in PCF Böhm trees

They are alternating sequences of moves OPO..., equipped with backward pointers from P moves to some previous O move.

Such plays are called views.

General plays (that will feature in a more synthetic definition of composition of strategies) will also have pointers from the O moves (to some previous P move).
Arenas

An arena $A$ is given by a set of moves $M$, which have a polarity $O$ or $P$ (formally, there is function $\lambda_A : M \to \{O, P\}$)

and by an enabling relation $\vdash$, which is the disjoint union of a subset of $M \times M$ (one writes $m \vdash n$) and of $M$ (one writes $\vdash m$).

If $m \vdash n$, then $m$ and $n$ have opposite polarities, and $\not\vdash n$. If $\vdash m$, then $m$ is an opponent move.
Product of two arenas

Let $A$ and $B$ be arenas. The arena $A \times B$ has as moves all $m_1$ such that $m$ is a move of $A$ and all moves $n_2$ such that $n$ is a move of $B$. Polarities of these moves are as in $A$ and $B$. Enabling is defined as follows:

$$\frac{\vdash_A m}{\vdash m_1} \quad \frac{\vdash_B n}{\vdash n_2} \quad \frac{m \vdash_A a}{m_1 \vdash a_1} \quad \frac{n \vdash_B b}{n_2 \vdash b_2}$$
Function space of two arenas

Let $A$ and $B$ be arenas. The arena $A \rightarrow B$ has as moves all $m_1$ such that $m$ is a move of $A$, with polarity opposite to that in $A$, and all moves $n_2$ such that $n$ is a move of $B$, with the same polarity as in $B$. Enabling is defined as follows:

$$\frac{\vdash_B n \quad \vdash_A m \quad \vdash_B n \quad m \vdash_A a \quad n \vdash_B b}{\vdash n_2 \quad n_2 \vdash m_1 \quad m_1 \vdash a_1 \quad n_2 \vdash b_2}$$
A legal play, or play for short, is a (possibly empty) sequence of moves of alternating polarity which is such that every occurrence of non-initial move is equipped with a pointer to a previous occurrence of a move justifying it. The set of legal plays over an arena $A$ is written $L_A$. (Clearly, a play starts with Opponent, since only opponent moves can be initial.)

A strategy on an arena $A$ is a non-empty set of even-length legal plays, which is closed under even-length prefixes.

Other conditions, like determinism and innocence, will be added later.
An automaton recognising $L_{A \rightarrow B}$

Convention: O and P moves of $A$ ($B$) are written $q, v$ ($q', v'$). Legal plays are all even-length words read by the following automaton (initial state $OO$):
Legal interactions

Let $A, B, C$ be three arenas. A legal interaction, or interaction for short, over these arenas is a sequence $u$ of moves from the three arenas such that

$$u \upharpoonright_{A,B} \in L_{A \rightarrow B}, \quad u \upharpoonright_{B,C} \in L_{B \rightarrow C}, \quad u \upharpoonright_{A,C} \in L_{A \rightarrow C}$$

We write $\text{int}(A, B, C)$ for the set of legal interactions over $A, B, C$.

In this definition, say, $u \upharpoonright_{A,B}$ denotes the subsequence of $u$ consisting only of the moves of $A, B$. One takes care of maintaining the moves of $A, B, C$ all distinct by tagging them if needed.
An automaton recognising legal interactions
Composition of strategies

Let $A, B, C$ be three arenas, and let $\sigma$ (resp. $\tau$) be a strategy of $A \rightarrow B$ (resp. $B \rightarrow C$). The following defines a strategy of $A \rightarrow C$, called the composition of $\sigma$ and $\tau$:

$$\tau \circ \sigma = \{ v \mid \exists u \in \text{int}(A, B, C) \; v = u \upharpoonright_{A,C}, u \upharpoonright_{A,B} \in \sigma, v \upharpoonright_{B,C} \in \tau \}.$$  

(we say that $u$ is a witness of $v$).

In the vocabulary of concurrency theory:

$$(|| \text{ composition}) + \text{ hiding}$$
**Associativity of composition**

If \( u \in \text{int}(A, C, D) \) and \( v \in \text{int}(A, B, C) \) are such that \( u \upharpoonright_{A,C} = v \upharpoonright_{A,C} \), then there is a unique \( w \in \text{int}(A, B, C, D) \) such that \( w \upharpoonright_{A,C,D} = u \) and \( w \upharpoonright_{A,B,C} = v \).

If \( \sigma, \tau, \upsilon \) are strategies of \( A \to B, B \to C, \) and \( C \to D \), respectively, then \( \upsilon \circ (\tau \circ \sigma) \equiv (\upsilon \circ \tau) \circ \sigma \).
Identity strategy

We define

\[ id' = \{ u \in L_{A \to A} \mid v \upharpoonright_1 = v \upharpoonright_2 \text{ for all even prefixes } v \text{ of } u \} \]

We define \( id'' \) as the smallest set of plays closed under the following rules:

- \( \epsilon \in id'' \)
- \( v \in id'\), a O move \( \Rightarrow va_2a_1 \in id'' \)
- \( v \in id'\), a P move \( \Rightarrow va_1a_2 \in id'' \)

We have \( id' = id'' \) (\( id \) for short), and \( id \) is a strategy.
The identity strategy is an identity

Let $A, B, C$ be three arenas and let $u \in \text{int}(A, B, C)$. Then $u$ is a sequence of blocks of the form $mb_1 \ldots b_k n$ where $m$ is an $O (A \rightarrow C)$ move, the $b_i$’s are $B$ moves, and $n$ is a $P (A \rightarrow C)$ move. Moreover, if $u$ is a witness for $\tau \circ \sigma$ (i.e., $u \upharpoonright_{A, B} \in \sigma$, $u \upharpoonright_{B, C} \in \tau$), and if $u = u' mb_1 \ldots b_k n$ where $mb_1 \ldots b_k n$ is as above, then $u'$ is also a witness.

We have always $id \circ \sigma = \sigma$ and $\sigma \circ id = \sigma$. 
Determinism, innocence

A strategy $\sigma$ is called deterministic when

$$smn_1, smn_2 \in \sigma \Rightarrow n_1 = n_2$$

The P view $\langle s \rangle$ of a play $s$ is defined as follows:

$$\langle \varepsilon \rangle = \varepsilon$$

$$\langle sn \rangle = \langle s \rangle n$$  \hspace{1cm} (n P move)

$$\langle sm \rangle = m$$ \hspace{1cm} (m initial)

$$\langle sns'm \rangle = \langle sn \rangle m$$ \hspace{1cm} (m O move, m points to n)

A deterministic strategy is called innocent if

$$s \in \sigma \iff \langle s \rangle \in \sigma$$

Arenas and strategies form a category. It contains as subcategories the categories of arenas and deterministic strategies, and of arenas and innocent strategies (idem for well-bracketed).
Innocence, logically

In a view, $O$, unlike $P$, has to play in the immediate subtype:

$$\lambda x \ldots \lambda y . x M \_1 M \_2$$

- $P =$ head variable $x : A_1 \rightarrow A_2 \rightarrow B$ can be bound far away
- $O =$ initial move of $M_1$ (or $M_2$) has to be in type $A_1$ (or $A_2$)
Fat versus meager

In the official HO semantics, the meaning of a PCF Böhm tree (or term) is made of all plays whose view is in (the transcription of the) tree (as strategy), as stressed in this course. We call the resulting set of plays the fat version. In contrast, we call our preferred version, consisting of views only, the meager version.

**Theorem.** The composition defined on the fat versions through (|| composition) + hiding is the fat version of the composition of the meager versions via the game abstract machine.
Illustrating (|| composition) + hiding

The (unique) legal interaction witnessing the interaction

\[ q_\epsilon[q_1, \leftrightarrow^0] \left\{ \begin{array}{l}
q_{11}[3_{11}, \leftrightarrow^0] \\
4_1[7_\epsilon, \leftrightarrow^1] \\
6_1[9_\epsilon, \leftrightarrow^1]
\end{array} \right. \]

\[ q_1[q_{11}, \leftrightarrow^0] \left\{ \begin{array}{l}
0_{11}[3_1, \leftrightarrow^1] \\
3_{11}[6_1, \leftrightarrow^1]
\end{array} \right. \]

is
(which belongs to the fat version of the strategy on the left)
The key result of Hyland and Ong

The properties characterizing (the injective interpretation of) PCF Böhm trees are

determinism, innocence and well-bracketing

(For AJM, characterization via history-freeness)
A cartesian closed category

We get the canonical currying isos “for free” (the moves are the same, up to retagging):

\[(A \times B) \to C\]

\[
\frac{(A \times B) \to C}{A \to (B \to C)}
\]

Moreover, one can interpret fixpoints (since we were able to read infinite Böhm trees as strategies).

Therefore, we have a model of PCF.
Full abstraction

We write $M =_{obs} N$ iff for all $C$ s.t. $C[M]$ and $C[N]$ are closed and of base type, we have: $C[M] \rightarrow^* c$ iff $C[N] \rightarrow^* c$.

A model is **fully abstract** when

$$\llbracket M \rrbracket = \llbracket N \rrbracket \quad \text{iff} \quad M =_{obs} N$$

The “if” direction is the difficult one
Definable separability

If the model is such that for every distinct \( f, g \) of the same type \( A \) (interpreting some syntactic type) there exists a *definable* \( h \) of type \( A \to \text{bool} \) such that \( hf \neq hg \), then it is fully abstract.

*Compact definability* + *extensionality* imply definable separability.
### Some results and some non-results

<table>
<thead>
<tr>
<th>Language</th>
<th>Model</th>
<th>Def.</th>
<th>FA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCF + por</td>
<td>Cont</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PCF + catch</td>
<td>$SA \approx (G_{inn}/=_{op})$</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PCF</td>
<td>$PCFBT/=_{op}$</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>PCF</td>
<td>$G_{AJM}, G_{HO}, PCFBT$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>PCF + control</td>
<td>$G_{inn}$</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Idealized Algol</td>
<td>$G_{wb}$</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

52
**HO games vs sequential algorithms**

Interpret exponential differently:
- HO games are *repetitive*, bool → bool infinite
- Sequential algorithms have *memory*, bool → bool infinite

Cf. two exponentials of coherence spaces (multiset and set)
Around full abstraction

- **Stable** model *(Berry, reinvented by Girard)* \(\mapsto\) **linear logic**.
- Sequential algorithms led ... to the **categorical abstract machine**, and then to **explicit substitutions**.
- The full abstraction problem boosted also the study of **logical relations** *(Sieber, O’Hearn and Ricky, Bucciarelli)*, and motivated **Bucciarelli-Ehrhard’s and Longley’s extensional accounts of sequentiality**.
The game semantics program

- references (Abramsky, McCusker, Honda)
- control (Laird)
- subtyping (Chroboczek)
- nondeterminism (Harmer), probabilistic choice (Danos and Harmer)
- call-by-value (Honda and Yoshida)
- concurrency (Ghica and Murawski, Laird)
Imperative arenas

- The arena **comm**: run and done
- The arena **var**:
  
  \[
  \text{read} \quad \text{write}(n) \quad (n \in \omega) \\
  \text{OK} \quad n \quad (n \in \omega)
  \]

The strategy **cell** :

\[
\text{write}(0) \text{ OK read 0} \\
\text{write}(0) \text{ OK write}(2) \text{ OK read 2}
\]

reads last written value (not innocent)
Cell discipline enforced by interaction

\[ [(x := 0); (x := x + 1)] = \]

\[ \{ \begin{array}{l}
\text{run}_\epsilon \text{write}(0) \text{OK} \\
\text{read}_1 \text{write}(n + 1) \text{OK} \\
\vdots
\end{array} \] \]

Interaction play with cell:

\[ \{ \begin{array}{l}
\text{run}_\epsilon \text{write}(0) \text{OK} \text{read}_1 0 \text{write}(1) \text{OK} \\
\text{done}_\epsilon
\end{array} \]
Definability through factorization

Every strategy $\sigma : A$ can be written as $(\tau \text{ cell})$ for some *innocent* strategy $\tau : \text{var} \rightarrow A$

Other such results:
- **catch** for non well-bracketing (*Laird*)
- a **dice** strategy for nondeterminism / probabilistic games (*Danos, Harmer*)
- a form of **case** for non-rigidity (a condition dual to well-bracketing) (*Danos, Harmer, Laurent*)
Definable separability

Essentially the same argument for sequential algorithms, and for the IA (non innocent) model

Let $\sigma_1, \sigma_2 : A$, let $m_1 \ldots m_{2n} \in \sigma_1 \setminus \sigma_2$. We define $\tau : A \rightarrow \mathbb{nat}$ as the minimal strategy that contains

$q m_1 \ldots m_{2n} 0$
Game semantics for verification

Up to second-order:

• pointers are useless, i.e., can be reconstructed uniquely (Ghica and McCusker)

• the strategies interpreting second-order IA terms (as sets of words) are *regular languages*.

Applications to decidability results. More results have been obtained for larger fragments (third order: Ong and others)