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.

\[
\begin{align*}
t_1 + Ss_2 &= S(t_1 + s_2) \\
t_1 + SSt_2 &= S(t_1 + St_2) \\
S(t_1 + St_2) &= SS(t_1 + s_2)
\end{align*}
\]
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 ($n \geq 1$, $p \geq 1$):

  $\lambda x_1 \cdots x_n.xM_1 \cdots M_p$

- 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

\[
\omega(M) = \begin{cases} 
\Omega & \text{if } M = \lambda x_1 \cdots x_n.(\lambda x.M)M_1 \cdots M_p \\
\lambda x_1 \cdots x_n.x\omega(M_1) \cdots \omega(M_p) & \text{if } M = \lambda x_1 \cdots x_n.xM_1 \cdots M_p,
\end{cases}
\]

\[
\frac{M_1 \leq N_1 \cdots M_p \leq N_p}{\Omega \leq M} \quad \frac{\lambda x_1 \cdots x_n.xM_1 \cdots M_p \leq \lambda x_1 \cdots x_n.xN_1 \cdots N_p}
\]

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

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

\[
\lambda x_1 \cdots x_n.x M_1 \ldots (\lambda y_1 \cdots y_p.y N_1 \ldots N_p) \ldots M_n
\]
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!).
\textit{\(\eta\)-long Böhm trees}

We restrict ourselves to the simply typed setting, and impose that

\begin{itemize}
  \item each occurrence of a variable must appear in a context where it is applied to all its arguments,
  \item 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.
\end{itemize}
Executing Böhm trees (the abstract machine)

Terms  \( M ::= (\lambda \vec{x}. W) \)  
Environments  \( e ::= \text{nil} \mid B \cdot e \)  
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

\[
\begin{cases}
  e = B_0 \cdot \cdots \cdot B_n \\
  B_i = \langle \vec{x}_i \leftarrow \vec{N}_i \rangle[e_i] \\
  x = x_{ij} \\
  N_{ij} = (\lambda \vec{z}. W)
\end{cases}
\]
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:

- $n : \text{Nat}$ (for all $n \in \omega$)
- $T, F : \text{Bool}$
- $\text{succ, pred} : \text{Nat} \rightarrow \text{Nat}$
- $\text{zero?} : \text{Nat} \rightarrow \text{Bool}$
- $\text{if then else} : \text{Bool} \rightarrow \text{Nat} \rightarrow \text{Nat} \rightarrow \text{Nat}$
- $\text{if then else} : \text{Bool} \rightarrow \text{Bool} \rightarrow \text{Bool} \rightarrow \text{Bool}$

- $\Omega : \sigma$ for all $\sigma$
- $Y : (\sigma \rightarrow \sigma) \rightarrow \sigma$ for all $\sigma$
PCF Böhm trees

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

\[ \frac{v : C}{\Gamma \vdash v : C} \]

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

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

\[
\begin{array}{c}
M_1 \quad \cdots \quad M_p \quad v_1 \rightarrow W_1 \quad \cdots \quad v_k \rightarrow W_k \\
\hline
\lambda \bar{x}.\text{case } y
\end{array}
\]
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 $\text{nat}$ for natural numbers has the following moves: one (initial) $\bigcirc$ move denoted $q$, and a $\bigtriangledown$ move $n$ for each natural number $n$. 

\[
\{ \{\epsilon\} \quad \{\epsilon, qn\} \quad \{ \bot \ n \} \}
\]
Product of arenas

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

$$
\begin{cases}
\{\epsilon\} \\
\{\epsilon, q_1m_1\} \\
\{\epsilon, q_2n_2\} \\
\{\epsilon, q_1m_1, q_2n_2\}
\end{cases}
\begin{cases}
(\bot, \bot) \\
(m, \bot) \\
(\bot, n) \\
(m, n)
\end{cases}
$$
**Function types** \((\text{nat}_1 \rightarrow \text{nat}_\epsilon)\)

\[
q \in q_1 \begin{cases} 0 \ 3 \epsilon \\ 1 \ 4 \epsilon \end{cases} \quad \lambda x. \ case \ x \begin{cases} 0 \rightarrow 3 \\ 1 \rightarrow 4 \end{cases}
\]

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

Interaction with \(\{q_1 0\}\) converges, interaction with \(\{q_1 2\}\) diverges.
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

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

Stuttering strategies 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:

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

\[q_\epsilon q_1 \left\{ \begin{array}{ll}
q_{11} & 4_1 7_\epsilon \\
6_1 9_\epsilon & \end{array} \right. \]
Pointers

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

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

(type ((bool_{111} \rightarrow bool_{11}) \rightarrow bool_1) \rightarrow bool_\epsilon)
\[ Kierstead_1 = q_\epsilon[q_1, 0] \]

\[ Kierstead_2 = q_\epsilon[q_1, 0] \]
Game abstract machine for PCF Böhm trees

\[
\begin{align*}
q_\epsilon q_1 & \begin{cases}
q_{11}^3 & 3_{11}^3 \\
4_{17}^7 & 6_{19}^9 \quad q_1 q_{11} \begin{cases}
0_{11}^3 & 3_{11}^3 \\
3_{11}^6 & 6_{19}^9
\end{cases}
\end{cases}
\end{align*}
\]

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

If \( n' \) points to \( m \), then play \( n \) under \( m' \)
Executing Kierstead against

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

\[
q_1[q_{11}, \iff] \begin{cases}
q_{111}[q_{11}, \iff] & \begin{cases}
q_{111}[F_{111}, \iff]
\end{cases}
T_{11}[T_{111}, \iff]
F_{11}[F_{111}, \iff]
\end{cases}
T_{11}[F_1, \iff]
F_{11}[T_1, \iff]
\]

\[ Kierstead_1 = q_\epsilon[q_1, \iff] \begin{cases}
q_{11}[q_{11}, \iff] & \begin{cases}
q_{11}[F_{111}, \iff]
\end{cases}
T_{11}[T_{111}, \iff]
F_{11}[F_{111}, \iff]
\end{cases}
T_{11}[T_\epsilon, \iff]
F_{11}[F_\epsilon, \iff]
\]
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:

$$q_\epsilon q_1 \begin{cases} q_{11} 0_\epsilon \\ q_{12} 1_\epsilon \\ n_1(n + 2)_\epsilon \end{cases}$$

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

$$\{ \begin{array}{c}
6_2 q_1 \\
2_1 q_1
\end{array} \} \{ \begin{array}{c}
4_1 8 \epsilon \\
2_1 1 \epsilon \\
4_1 5 \epsilon \\
1_1 6 \epsilon
\end{array} \}$$

against $q_1 4_1$ yields $q_\epsilon q_2 \{ \begin{array}{c}
6_2 8 \epsilon \\
3_2 5 \epsilon
\end{array} \}$$

Lazy, stream-like loop of evaluation.
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 $\nvdash n$. 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:

\[
\begin{align*}
\vdash_A m & \quad \vdash_B n \\
\vdash m_1 & \quad \vdash n_2 \\
m \vdash_A a & \quad n \vdash_B b \\
m_1 \vdash a_1 & \quad n_2 \vdash b_2
\end{align*}
\]
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:

$$
\vdash_B n \quad \vdash_A m \quad \vdash_B n \quad m \vdash_A a \quad n \vdash_B b
$$

$$
\vdash_{n_2} \quad n_2 \vdash m_1 \quad m_1 \vdash a_1 \quad n_2 \vdash b_2
$$
Plays and strategies

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.
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.
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 $\tau$:

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

(we say that $u$ is a witness of $v$).
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 \( \upsilon \circ (\tau \circ \sigma) = (\upsilon \circ \tau) \circ \sigma \).
Identity strategy

We define

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

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

\[
\begin{align*}
\epsilon & \in id'' \\
v \in id'', \text{ a } O \text{ move} & \quad \Rightarrow \quad va_2a_1 \in id'' \\
v \in id'', \text{ a } P \text{ move} & \quad \Rightarrow \quad va_1a_2 \in id''
\end{align*}
\]

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 $\sigma \circ id = \sigma$. 
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:

- $\langle \epsilon \rangle = \epsilon$
- $\langle sn \rangle = \langle s \rangle n$ (n P move)
- $\langle sm \rangle = m$ (m initial)
- $\langle sns'm \rangle = \langle sn \rangle m$ (m O move, m points to n)

A deterministic strategy is called innocent if

$$s \in \sigma \Leftrightarrow \langle s \rangle \in \sigma$$

Arenas and strategies form a category. It contains as subcategories the categories of arenas and deterministic strategies, and of arenas and innocent strategies.
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

$$[[M]] = [[N]] \iff M =_{obs} N$$

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 $A \rightarrow \text{bool}$ 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>
HO games vs sequential algorithms

Interpret exponential differently:
• HO games are *repetitive*, $\text{bool} \rightarrow \text{bool}$ infinite
• Sequential algorithms have *memory*, $\text{bool} \rightarrow \text{bool}$ infinite

Cf. two exponentials of coherence spaces (multiset and set)
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)
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 $A_2$)
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**:
  
  read \( \text{write}(n) \) \((n \in \omega)\)
  OK \(n \) \((n \in \omega)\)

The strategy **cell** :

  write(0) OK read 0
  write(0) OK write(2) OK read 2

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

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

\[ \text{run}_\epsilon \text{write}(0)_1 \text{OK}_1 \text{read}_1 \begin{cases} \vdots \\ n_1 \text{write}(n + 1)_1 \text{OK}_1 \text{done}_\epsilon \end{cases} \]

Interaction play with cell:

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

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

Other such results:
- \textbf{catch} for non well-bracketing (Laird)
- \textbf{a dice} strategy for nondeterminism / probabilistic games (Danos, Harmer)
- \textbf{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 $\tau : A \rightarrow \text{nat}$ 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)