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.

$$
\dfrac{\dfrac{}{t_1 + SSt_2 = S(t_1 + St_2)} \qquad \dfrac{\dfrac{}{t_1 + St_2 = S(t_1 + t_2)}}{S(t_1 + St_2) = SS(t_1 + t_2)}}{t_1 + SSt_2 = SS(t_1 + t_2)}
$$

Untyped λ-calculus

The syntax of the untyped λ-calculus (λ-calculus for short) is given by:

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

where x is called a variable, $M_1 M_2$ is called an application, and $\lambda x.M$ is called an abstraction.

β-reduction:

$$\frac{}{(\lambda x.M)N \to M[x \leftarrow N]}$$

$$\frac{M \to M'}{MN \to M'N} \qquad \frac{N \to N'}{MN \to MN'} \qquad \frac{M \to M'}{\lambda x.M \to \lambda x.M'}$$

Normal forms in the λ-calculus

Any λ-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

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

$$\frac{}{\Omega \leq M} \qquad \frac{M_1 \leq N_1 \cdots M_p \leq N_p}{\lambda x_1 \cdots x_n.xM_1 \cdots M_p \leq \lambda x_1 \cdots x_n.xN_1 \cdots N_p}$$

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

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

Böhm trees as strategies

$$\dfrac{M_1 \quad \ldots \quad \dfrac{N_1 \ \ldots \ N_p}{\lambda y_1 \cdots y_p . y} \quad \ldots \quad M_n}{\lambda x_1 \cdots x_n . x}$$

$$\lambda x_1 \cdots x_n . x M_1 \ldots (\lambda y_1 \cdots y_p . y N_1 \ldots N_p) \ldots M_n$$

Simply typed λ-calculus

$$A ::= C \mid A \to A \qquad (C \text{ base type})$$

Thus every type write s uniquely as $\boldsymbol{A_1 \to \cdots \to A_n \to C}$.

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

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : A \to B}$$

$$\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

Böhm trees associated to simply typable terms are finite (see Silvia's lectures!).

η-long Böhm trees

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	$e ::= nil \parallel B{\bullet}e$
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.

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

$$\text{where} \begin{cases} e = B_0{\bullet}\cdots{\bullet}B_n & B_i = \langle\vec{x_i} \leftarrow \vec{N_i}\rangle[e_i] \\ x = x_{ij} & N_{ij} = (\lambda\vec{z}.W) \end{cases}$$

Executing Böhm trees (illustration)

$2'$	$u(\lambda x.u(\lambda y.x))$	$\langle u \overset{2}{\leftarrow} (\lambda r.r(r(z)))\rangle[\ ] = \rho_0$
$3'$	$r(r(z))$	$\langle r \overset{3}{\leftarrow} (\lambda x.u(\lambda y.x))\rangle[\rho_0] = \rho_1$
$4'$	$u(\lambda y.x)$	$\langle x \leftarrow (r(z))\rangle[\rho_1]\bullet(\rho_0 = \langle u \overset{4}{\leftarrow} (\lambda r.r(r(z)))\rangle[\ ]) = \rho_2$
$5'$	$r(r(z))$	$\langle r \overset{5}{\leftarrow} (\lambda y.x)\rangle[\rho_2] = \rho_3$
$6'$	x	$\langle y \leftarrow (r(z))\rangle[\rho_3]\bullet(\rho_2 = \langle x \overset{6}{\leftarrow} (r(z))\rangle[\rho_1]\bullet\rho_0) = \rho_4$
$7'$	$r(z)$	$\langle\rangle[\rho_4]\bullet(\rho_1 = \langle r \overset{7}{\leftarrow} (\lambda x.u(\lambda y.x))\rangle[\rho_0]) = \rho_5$
$8'$	$u(\lambda y.x)$	$\langle x \leftarrow (z)\rangle[\rho_5]\bullet(\rho_0 = \langle u \overset{8}{\leftarrow} (\lambda r.r(r(z)))\rangle[\ ]) = \rho_6$
$9'$	$r(r(z))$	$\langle r \overset{9}{\leftarrow} (\lambda y.x)\rangle[\rho_6] = \rho_7$
$10'$	x	$\langle y \leftarrow (r(z))\rangle[\rho_7]\bullet(\rho_6 = \langle x \overset{10}{\leftarrow} (z)\rangle[\rho_5]\bullet\rho_0) = \rho_8$
$11'$	z	$\langle\rangle[\rho_8]\bullet\rho_5$

The language PCF

PCF is simply typed λ-calculus, with the following constants

$$
\begin{array}{lll}
n & : Nat & (n \in \omega) \\
T, F & : Bool & \\
succ, pred & : Nat \to Nat & \\
zero? & : Nat \to Bool & \\
\textit{if} \quad \textit{then} \quad \textit{else} & : Bool \to Nat \to Nat \to Nat & \\
\textit{if} \quad \textit{then} \quad \textit{else} & : Bool \to Bool \to Bool \to Bool & \\
& & \\
\Omega & : A & \text{for all } A \\
Y & : (A \to A) \to A & \text{for all } A
\end{array}
$$

Operational semantics of PCF

$$(\lambda x.M)N \to M[x \leftarrow N] \qquad YM \to M(YM)$$

$$zero?(0) \to T \qquad zero?(n+1) \to F$$

$$succ(n) \to n+1 \qquad pred(n+1) \to n$$

$$\textit{if } T \textit{ then } N \textit{ else } P \to N \qquad \textit{if } F \textit{ then } N \textit{ else } P \to P$$

$$\frac{M \to M'}{MN \to M'N} \qquad \frac{M \to M'}{\textit{if } M \textit{ then } N \textit{ else } P \to \textit{if } M' \textit{ then } N \textit{ else } P}$$

$$\frac{M \to M'}{f(M) \to f(M')} \quad (\text{for } f \in \{succ, pred, zero?\})$$

PCF Böhm trees

$$M ::= \lambda\vec{x} : \vec{A}.W \qquad W ::= v \parallel \textsf{case}\ yM_1 \cdots M_n\ [v_1 \to W_1, \ldots, v_k \to W_k]$$

$$\frac{v : C}{\Gamma \vdash v : C}$$

$$\frac{\begin{array}{c}\Gamma\bullet y = A_1 \to \cdots \to A_n \to C \quad v_1, \ldots, v_k : C \\ \Gamma \vdash M_1 : A_1\ \cdots\ \Gamma \vdash M_n : A_n \quad \Gamma \vdash W_1 : C_1\ \cdots\ \Gamma \vdash W_k : C_1\end{array}}{\Gamma \vdash \textsf{case}\ yM_1 \cdots M_n\ [v_1 \to W_1, \ldots, v_k \to W_k] : C_1}$$

$$\frac{\Gamma, \vec{x} : \vec{A} \vdash W : C}{\Gamma \vdash (\lambda\vec{x} : \vec{A}.W) : \vec{A} \to C}$$

Note that a body W is always of base type C.

PCF Böhm trees as strategies

$$\frac{M_1 \quad \cdots \quad M_p \quad v_1 \to W_1 \quad \cdots \quad v_k \to W_k}{\lambda \vec{x}.\mathtt{case}\ y}$$

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 **nat** for natural numbers has the following moves: one (initial) O move denoted q, and a P move n for each natural number n.

$$\begin{cases} \{\epsilon\} \\ \{\epsilon, qn\} \end{cases} \qquad\qquad \begin{cases} \perp \\ n \end{cases}$$

Product of arenas

The arena **nat** $\times$ **nat** is made of two disjoints copies of nat.

$$\left\{\begin{array}{l}\{\epsilon\}\\ \{\epsilon, q_1 m_1\}\\ \{\epsilon, q_2 n_2\}\\ \{\epsilon, q_1 m_1, q_2 n_2\}\end{array}\right. \qquad \left\{\begin{array}{l}(\perp,\perp)\\ (m,\perp)\\ (\perp,n)\\ (m,n)\end{array}\right.$$

Function types ($\mathsf{nat}_1 \rightarrow \mathsf{nat}_\epsilon$)

$$q_\epsilon q_1 \begin{cases} 0_1 3_\epsilon \\ 3_1 6_\epsilon \end{cases} \qquad \lambda x.\,\mathsf{case}\,x \begin{cases} 0 \to 3 \\ 3 \to 6 \end{cases}$$

Note the inversion of polarity for $\mathbf{nat}_1$. Also, $q_\epsilon \vdash q_1$.

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

Plays and (deterministic) strategies

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

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

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 σ:

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

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

Stuttering strategies *do not* exist in the earlier model of sequential algorithms of PCF (Berry-Curien, late 1970).

A higher-order type

$(\mathsf{nat}_{11} \to \mathsf{nat}_{1}) \to \mathsf{nat}_{\epsilon}$

$$h = \lambda f.\,\mathsf{case}\ f(3)[4 \to 7, 6 \to 9]$$

As a strategy:

$$\lambda f.\,\mathsf{case}\ f \begin{cases} (3) \\ 4 \to 7 \\ 6 \to 9 \end{cases} \qquad q_\epsilon q_1 \begin{cases} q_{11} 3_{11} \\ 4_1 7_\epsilon \\ 6_1 9_\epsilon \end{cases}$$

Pointers

$$Kierstead_1 = \lambda f.\mathsf{case}\ f(\lambda x.\mathsf{case}\ f(\lambda y.\mathsf{case}\ x))$$

$$Kierstead_2 = \lambda f.\mathsf{case}\ f(\lambda x.\mathsf{case}\ f(\lambda y.\mathsf{case}\ y))$$

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

(type $((\mathsf{bool}_{111} \to \mathsf{bool}_{11}) \to \mathsf{bool}_1) \to \mathsf{bool}_\epsilon$) **AMBIGUOUS!**

de Bruijn

The solution to this ambiguity is to import the de Bruijn technology of pointers! Recall that in de Bruijn notation, bound variables are replaced by numbers recording "how far they are bound", e.g.

$$\lambda x.x(\lambda y.yx) \quad \rightsquigarrow \quad \lambda.0(\lambda.01)$$

We do the same here to disambiguate q_{111} (next slide).

$$Kierstead_1 = q_\epsilon[q_1, \overset{0}{\hookleftarrow}] \begin{cases} q_{11}[q_1, \overset{1}{\hookleftarrow}] \begin{cases} q_{11}\underline{[q_{111}, \overset{1}{\hookleftarrow}]} \begin{cases} T_{111}[T_{11}, \overset{1}{\hookleftarrow}] \\ F_{111}[F_{11}, \overset{1}{\hookleftarrow}] \end{cases} \\ T_1[T_{11}, \overset{1}{\hookleftarrow}] \\ F_1[F_{11}, \overset{1}{\hookleftarrow}] \end{cases} \\ T_1[T_\epsilon, \overset{1}{\hookleftarrow}] \\ F_1[F_\epsilon, \overset{1}{\hookleftarrow}] \end{cases}$$

$$Kierstead_2 = q_\epsilon[q_1, \overset{0}{\hookleftarrow}] \begin{cases} q_{11}[q_1, \overset{1}{\hookleftarrow}] \begin{cases} q_{11}\underline{[q_{111}, \overset{0}{\hookleftarrow}]} \begin{cases} T_{111}[T_{11}, \overset{1}{\hookleftarrow}] \\ F_{111}[F_{11}, \overset{1}{\hookleftarrow}] \end{cases} \\ T_1[T_{11}, \overset{1}{\hookleftarrow}] \\ F_1[F_{11}, \overset{1}{\hookleftarrow}] \end{cases} \\ T_1[T_\epsilon, \overset{1}{\hookleftarrow}] \\ F_1[F_\epsilon, \overset{1}{\hookleftarrow}] \end{cases}$$

Inserting the implicit pointers in our previous examples

$h = \lambda f.\, \mathsf{case}\, f(3)[4 \to 7, 6 \to 9]$ $\qquad$ $\lambda x.\, \mathsf{case}\, x[0 \to 3, 3 \to 6]$

$$q_\epsilon q_1 \begin{cases} q_{11} 3_{11} \\ 4_1 7_\epsilon \\ 6_1 9_\epsilon \end{cases} \qquad q_1 q_{11} \begin{cases} 0_{11} 3_1 \\ 3_{11} 6_1 \end{cases}$$

$$q_\epsilon[q_1, \overset{0}{\hookleftarrow}] \begin{cases} q_{11}[3_{11}, \overset{0}{\hookleftarrow}] \\ 4_1[7_\epsilon, \overset{1}{\hookleftarrow}] \\ 6_1[9_\epsilon, \overset{1}{\hookleftarrow}] \end{cases} \qquad q_1[q_{11}, \overset{0}{\hookleftarrow}] \begin{cases} 0_{11}[3_1, \overset{1}{\hookleftarrow}] \\ 3_{11}[6_1, \overset{1}{\hookleftarrow}] \end{cases}$$

Game abstract machine (GAM) for PCF Böhm trees

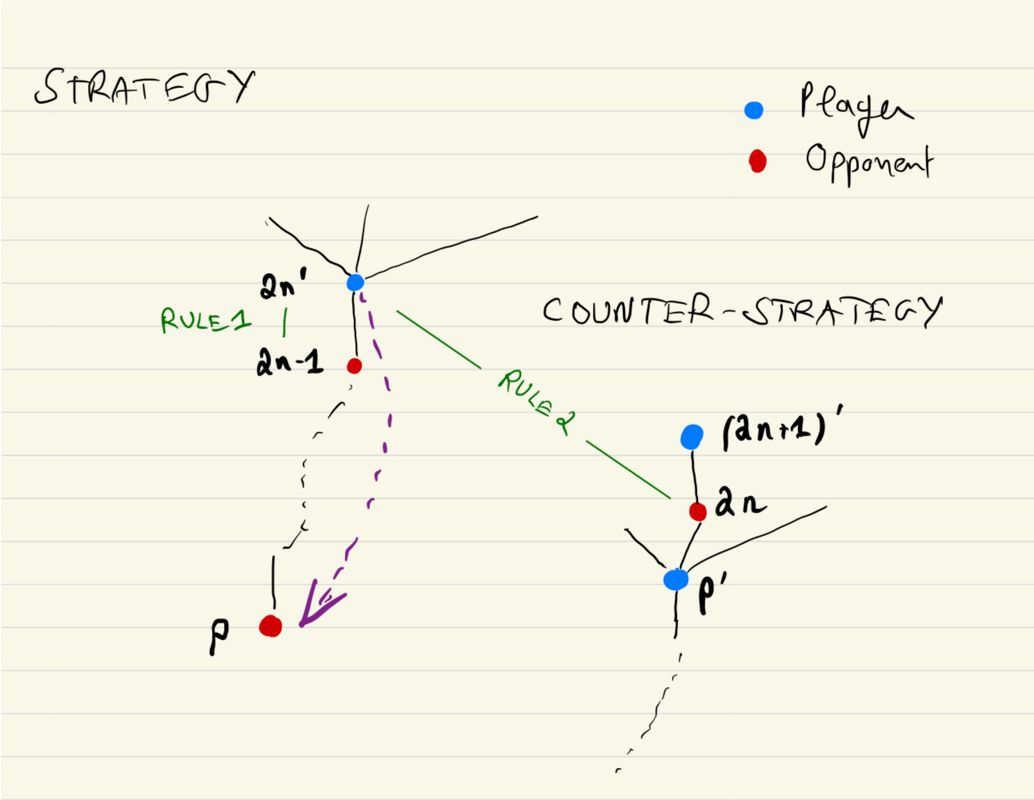
The machine takes as input a strategy σ of type, say, $A \to C$ and a strategy τ of type A (which we call the counter-strategy), and returns a strategy of type C (that is, a value of base type).

The machine explores both σ and τ, alternatively. The successive steps are $\mathbf{1}, \mathbf{2}', \mathbf{2}, \mathbf{3}', \mathbf{3}, \ldots$, with $\mathbf{n}, (\mathbf{n}+\mathbf{1})'$ pointing in σ for n odd and in τ for n even. The machine has two rules:

• Rule 1. At step $\mathbf{1}$, the machine points to the root of σ. At each step $\mathbf{n}$, the machine is pointing to an O move in either σ or τ, and the next step $(\mathbf{n}+\mathbf{1})'$ is played according to what σ or τ (both deterministic) prescribes.

• Rule 2. Once a (Player) step ($\mathbf{q}'$ has been performed, the next step $\mathbf{q}$ is performed on the other side as described next slide (so the machine goes back and forth between σ and τ).

The GAM, pictorially



• If $\mathbf{q}'$ points to $\mathbf{p}$, then the next step $\mathbf{q}$ is played over $\mathbf{p}'$. The name m of the move played at time $\mathbf{q}'$ dictates which branch to choose from $\mathbf{p}'$. If $\mathbf{p}$ is 1, then play $\mathbf{q}$ at the root of τ.

GAM: an example of execution

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

$$\overset{\boxed{2'}}{\langle q_\epsilon, \mathbf{1}\rangle}[q_1, \overset{0}{\hookleftarrow}] \begin{cases} \overset{\boxed{4'}}{\langle q_{11}, \mathbf{3}\rangle}[3_{11}, \overset{0}{\hookleftarrow}] \\ \overset{\boxed{6'}}{\langle 6_1, \mathbf{5}\rangle}[9_\epsilon, \overset{1}{\hookleftarrow}] \end{cases}$$

$$\overset{\boxed{3'}}{\langle q_1, \mathbf{2}\rangle}[q_{11}, \overset{0}{\hookleftarrow}] \begin{cases} \overset{\boxed{5'}}{\langle 3_{11}, \mathbf{4}\rangle}[6_1, \overset{1}{\hookleftarrow}] \end{cases}$$

Executing Kierstead against

$$\lambda g.\texttt{case}\ g(\texttt{case}\ gT\ [T \to T, F \to F])\ [T \to F, F \to T]$$

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

$$\mathit{Kierstead}_1 = q_\epsilon[q_1, \overset{0}{\hookleftarrow}] \begin{cases} q_{11}[q_1, \overset{1}{\hookleftarrow}] \begin{cases} q_{11}\underline{[q_{111}, \overset{1}{\hookleftarrow}]} \begin{cases} T_{111}[T_{11}, \overset{1}{\hookleftarrow}] \\ F_{111}[F_{11}, \overset{1}{\hookleftarrow}] \end{cases} \\ T_1[T_{11}, \overset{1}{\hookleftarrow}] \\ F_1[F_{11}, \overset{1}{\hookleftarrow}] \end{cases} \\ \\ T_1[T_\epsilon, \overset{1}{\hookleftarrow}] \\ F_1[F_\epsilon, \overset{1}{\hookleftarrow}] \end{cases}$$

$$\langle q_\epsilon, \mathbf{1}\rangle[q_1, \overset{0}{\hookleftarrow}] \left\{\begin{array}{l}
\langle q_{11}, \mathbf{3}\rangle[q_1, \overset{1}{\hookleftarrow}] \left\{\begin{array}{l}
\langle q_{11}, \mathbf{5}\rangle[q_{111}, \overset{1}{\hookleftarrow}] \left\{ \langle T_{111}, \mathbf{15}\rangle[T_{11}, \overset{1}{\hookleftarrow}] \right. \\
\langle F_1, \mathbf{17}\rangle[F_{11}, \overset{1}{\hookleftarrow}]
\end{array}\right. \\
\langle q_{11}, \mathbf{7}\rangle[q_1, \overset{1}{\hookleftarrow}] \left\{\begin{array}{l}
\langle q_{11}, \mathbf{9}\rangle[q_{111}, \overset{1}{\hookleftarrow}] \left\{ \langle F_{111}, \mathbf{11}\rangle[F_{11}, \overset{1}{\hookleftarrow}] \right. \\
\langle T_1, \mathbf{13}\rangle[T_{11}, \overset{1}{\hookleftarrow}]
\end{array}\right. \\
\langle T_1, \mathbf{19}\rangle[T_\epsilon, \overset{1}{\hookleftarrow}]
\end{array}\right.$$

$$\left\{\begin{array}{l}
\langle q_1, \mathbf{2}\rangle[q_{11}, \overset{0}{\hookleftarrow}] \left\{\begin{array}{l}
\langle q_{111}, \mathbf{6}\rangle[q_{11}, \overset{1}{\hookleftarrow}] \left\{\begin{array}{l}
\langle q_{111}, \mathbf{10}\rangle[F_{111}, \overset{0}{\hookleftarrow}] \\
\langle T_{11}, \mathbf{14}\rangle[T_{111}, \overset{1}{\hookleftarrow}]
\end{array}\right. \\
\langle F_{11}, \mathbf{18}\rangle[T_1, \overset{1}{\hookleftarrow}]
\end{array}\right. \\
\langle q_1, \mathbf{4}\rangle[q_{11}, \overset{0}{\hookleftarrow}] \left\{ \langle T_{11}, \mathbf{16}\rangle[F_1, \overset{1}{\hookleftarrow}] \right. \\
\langle q_1, \mathbf{8}\rangle[q_{11}, \overset{0}{\hookleftarrow}] \left\{ \langle F_{11}, \mathbf{12}\rangle[T_1, \overset{1}{\hookleftarrow}] \right.
\end{array}\right.$$

Compare with the stack-free environment machine

$$
\begin{array}{c}\boxed{(}\\ \langle q_\epsilon,\mathbf{1}\rangle\end{array}
\begin{array}{c}\boxed{u}\\ [q_1,\overset{0}{\hookleftarrow}]\end{array}
\left\{\begin{array}{l}
\begin{array}{c}\boxed{(\lambda x}\\ \langle q_{11},\mathbf{3}\rangle\end{array}
\begin{array}{c}\boxed{u}\\ [q_1,\overset{1}{\hookleftarrow}]\end{array}
\left\{\begin{array}{c}\boxed{(\lambda y}\\ \langle q_{11},\mathbf{5}\rangle\end{array}
\begin{array}{c}\boxed{x}\\ [q_{111},\overset{1}{\hookleftarrow}]\end{array}\right.
\\[2ex]
\begin{array}{c}\boxed{(\lambda x}\\ \langle q_{11},\mathbf{7}\rangle\end{array}
\begin{array}{c}\boxed{u}\\ [q_1,\overset{1}{\hookleftarrow}]\end{array}
\left\{\begin{array}{c}\boxed{(\lambda y}\\ \langle q_{11},\mathbf{9}\rangle\end{array}
\begin{array}{c}\boxed{x}\\ [q_{111},\overset{1}{\hookleftarrow}]\end{array}\right.
\end{array}\right.
$$

$$
\left\{\begin{array}{l}
\begin{array}{c}\boxed{(\lambda r}\\ \langle q_1,\mathbf{2}\rangle\end{array}
\begin{array}{c}\boxed{r}\\ [q_{11},\overset{0}{\hookleftarrow}]\end{array}
\left\{\begin{array}{c}\boxed{(}\\ \langle q_{111},\mathbf{6}\rangle\end{array}
\begin{array}{c}\boxed{r}\\ [q_{11},\overset{0}{\hookleftarrow}]\end{array}
\left\{\begin{array}{c}\boxed{(}\\ \langle q_{111},\mathbf{10}\rangle\ [z,\overset{-}{\hookleftarrow}]\end{array}\right.\right.
\\[2ex]
\begin{array}{c}\boxed{(\lambda r}\\ \langle q_1,\mathbf{4}\rangle\end{array}
\begin{array}{c}\boxed{r}\\ [q_{11},\overset{0}{\hookleftarrow}]\end{array}
\\
\begin{array}{c}\boxed{(\lambda r}\\ \langle q_1,\mathbf{8}\rangle\end{array}
\begin{array}{c}\boxed{r}\\ [q_{11},\overset{0}{\hookleftarrow}]\end{array}
\end{array}\right.
$$

Non-linearity

The last two examples feature non-linear terms (multiple occurrences of f, and multiple occurrences of u, r).

Each time the machine revisits a node that was already visited, it has to open a new copy of the strategy. (In the last example, see steps $\mathbf{2}$, $\mathbf{4}$ and $\mathbf{8}$, and $\mathbf{3}$ and $\mathbf{7}$.)

Steps versus nodes versus moves

One should not confuse

- the successive **steps** of the machine (in bold), that are numbers $\mathbf{n}$ or $\mathbf{n}'$,
- the **nodes** (or positions, or occurrences) in the trees of the strategy and counter-strategy,
- the **moves** of the relevant arenas.

• In a strategy, nodes are decorated by moves. The same move can decorate several nodes (cf. stuttering), but locally all the moves decorating the children of a P move are labelled by different moves.

• In a GAM execution, there are additionally decorations on nodes by numbers and primed numbers, that act as time stamps (and may induce copy creation, cf. previous slide).

Spelling out Rule 2 of the GAM in full detail

"If $\mathbf{q}'$ points to $\mathbf{p}$, then the next step $\mathbf{q}$ is played over $\mathbf{p}'$" means, slowly:

At step $\mathbf{q}'$, the machine has visited a P node, labelled by a move m in, say, σ. This node is equipped with a pointer to an O node in (a copy of a subtree of) σ, which was visited at time $\mathbf{p}$. Then, at the next step $\mathbf{q}$, the machine will visit a node β which is among the children of the node α of (a copy ... of) τ visited by the machine at time $\mathbf{p}'$. The name m of the move played at time $\mathbf{q}'$ dictates which child to choose: the one which is labelled by m!

Well-bracketing

This strategy of $(\mathsf{nat}_{11} \times \mathsf{nat}_{12} \to \mathsf{nat}_1) \to \mathsf{nat}_\epsilon$ 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_ϵ may be answered while q_1 and q_{11} are still open.

In the well-bracketed discipline, along any play questions must be answered obeying a stack discipline.

Strong (or partial) evaluation

If we want now to let $\sigma : A \to B$ and $\tau : A$ to interact (with B not of base type), then we need to relaunch (but not reboot) the machine again and again.

$$q_\epsilon q_2 \begin{cases} 6_2 q_1 \begin{cases} 4_1 8_\epsilon \\ 2_1 1_\epsilon \end{cases} \\ 3_2 q_1 \begin{cases} 4_1 5_\epsilon \\ 1_1 6_\epsilon \end{cases} \end{cases} \quad \text{against} \quad q_1 4_1 \quad \text{yields} \quad q_\epsilon q_2 \begin{cases} 6_2 8_\epsilon \\ 3_2 5_\epsilon \end{cases}$$

$$(\lambda xy.\mathsf{case}\ y\ [6 \to \mathsf{case}\ x\ [\ldots], \ldots])4 \quad \to \quad \lambda y.\mathsf{case}\ y\ [\ldots]$$

To produce the branch $q_\epsilon q_2 6_2 8_\epsilon$, "we" (= the Opponent in B) have to provide 6_2 and then the machine can continue.

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.

The arena **nat** for natural numbers

The set of moves is $\{q\} \cup \mathbb{N}$, with $\lambda(q) = O$ and $\lambda(n) = P$, and we set

$$\begin{aligned} &\vdash q \\ &q \vdash n \end{aligned}$$

Similarly for **bool**.

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:

$$\frac{\vdash_A m}{\vdash m_1} \qquad \frac{\vdash_B n}{\vdash n_2} \qquad \frac{m \vdash_A a}{m_1 \vdash a_1} \qquad \frac{n \vdash_B b}{n_2 \vdash b_2}$$

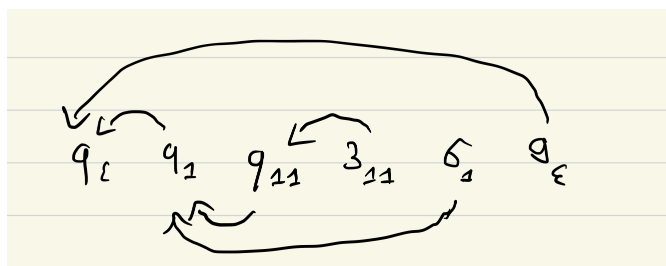
Function space of two arenas

Let A and B be arenas. The arena $A \to B$ 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:

$$\frac{\vdash_B n}{\vdash n_2} \quad \frac{\vdash_A m \qquad \vdash_B n}{n_2 \vdash m_1} \quad \frac{m \vdash_A a}{m_1 \vdash a_1} \quad \frac{n \vdash_B b}{n_2 \vdash b_2}$$

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. (A play starts with Opponent, since only opponent moves can be initial.) A legal play looks like this:



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, can be added.

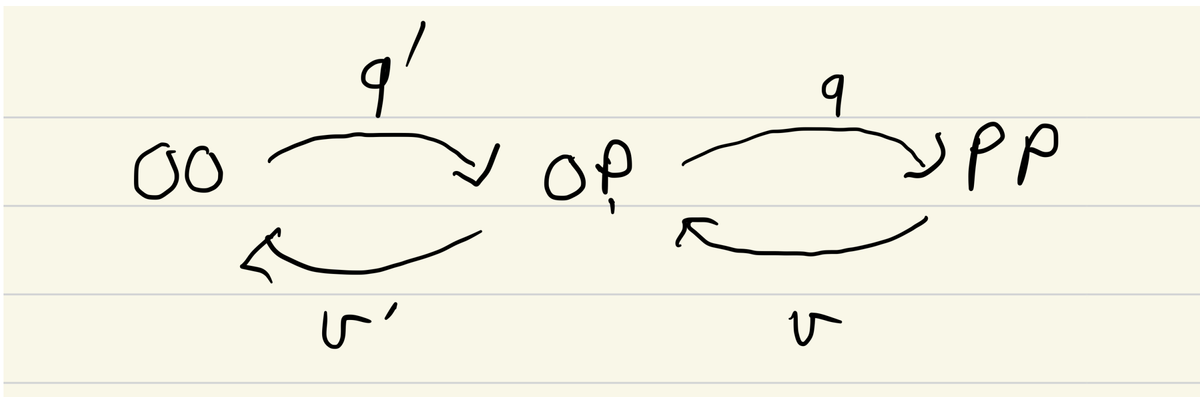
Strategies as trees

We note that any set of words over any alphabet can be organised as a forest (each word is a branch): try! It is a forest, in general.

Now, for the arenas interpreting PCF types, we note that they all have a unique intial move: this is because the initial move of the arena interpreting $A_1 \to \ldots \to A_n \to C$ has as only initial move the unique initial move of (the arena interpreting) C. So we have trees.

An automaton recognising $L_{A \to B}$

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



(If one insists on final states, then taking OO and PP as final states implements the even-length constraint!)

Additionally, the word must be equipped with pointers respecting enablings!

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

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

We write $int(A, B, C)$ 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

$$
\begin{tikzcd}
OOO \arrow[r, bend left, "q''"] & OOP \arrow[l, bend left, "r''"] \arrow[r, bend left, "q'"] & OPP \arrow[l, bend left, "r'"] \arrow[r, bend left, "q"] & PPP \arrow[l, bend left, "r"]
\end{tikzcd}
$$

Composition of strategies

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

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

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

In the vocabulary of concurrency theory:

(|| composition) + hiding

Associativity of composition

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

(Here, $int(A, B, C, D)$ is defined in a similar way as $int(A, B, C)$, by taking restrictions to $A \to B$, $B \to C$, $C \to D$, and $A \to D$.)

Proposition. If σ, τ, υ 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

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

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

$$\frac{}{\epsilon \in id''} \qquad \frac{v \in id'', a\ O \text{ move}}{va_2a_1 \in id''} \qquad \frac{v \in id'', a\ P \text{ move}}{va_1a_2 \in id''}$$

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 $u \in int(A, B, C)$. Then u is a sequence of blocks of the form $mb_1 \ldots b_k n$ where m is an O $(A \to C)$ move, the b_i's are B moves, and n is a P $(A \to C)$ 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 $u = u' m b_1 \ldots b_k n$ 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

- Recall that a strategy σ is called deterministic when

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

- The P view $\ulcorner s \urcorner$ of a play s is defined as follows:

$$\begin{array}{ll}
\ulcorner \epsilon \urcorner = \epsilon & \\
\ulcorner sn \urcorner = \ulcorner s \urcorner n & (n \text{ P move}) \\
\ulcorner sm \urcorner = m & (m \text{ initial}) \\
\ulcorner sns'm \urcorner = \ulcorner sn \urcorner m & (m \text{ O move}, m \text{ points to } n)
\end{array}$$

A deterministic strategy is called innocent if

$$s \in \sigma \Leftrightarrow \ulcorner s \urcorner \in \sigma$$

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).

Visibility

In fact, the definition of view just given is sloppy: what to do with the pointers?

Hyland and Ong further impose a visibility condition on plays, which guarantees that you do not lose your (remaining) pointers while defining the view recursively (details omitted).

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 $x : A_1 \to A_2 \to B$ 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.

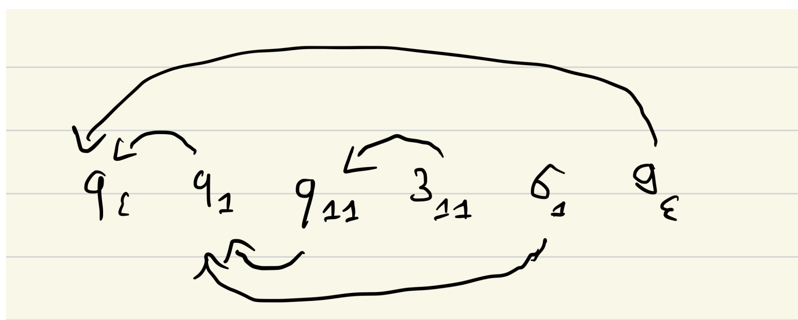
Theorem. The composition defined on the fat versions through ($||$ composition) + hiding is the fat version of the composition of the meager versions via the game abstract machine.

Illustrating ($\|$ composition) + hiding

The (unique) legal interaction witnessing the interaction

$$q_\epsilon[q_1, \stackrel{0}{\hookleftarrow}] \begin{cases} q_{11}[3_{11}, \stackrel{0}{\hookleftarrow}] \\ 4_1[7_\epsilon, \stackrel{1}{\hookleftarrow}] \\ 6_1[9_\epsilon, \stackrel{1}{\hookleftarrow}] \end{cases} \qquad q_1[q_{11}, \stackrel{0}{\hookleftarrow}] \begin{cases} 0_{11}[3_1, \stackrel{1}{\hookleftarrow}] \\ 3_{11}[6_1, \stackrel{1}{\hookleftarrow}] \end{cases}$$

is



which belongs to the fat version of the strategy on the left:

$$\ulcorner u \urcorner = q_\epsilon[q_1, \stackrel{0}{\hookleftarrow}]6_1[9_\epsilon, \stackrel{1}{\hookleftarrow}]$$

The key result of Hyland and Ong

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

determinism, innocence and well-bracketing

(For AJM, characterisation via history-freeness)

A cartesian closed category

• For products, we need

$$\frac{(A \to B_1) \times (A \to B_2)}{A \to (B_1 \times B_2)}$$

indeed, a strategy in $A \to (B_1 \times B_2)$ is a forest which can be split in two forests (according to where the roots come from).

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

$$\frac{(A \times B) \to C}{A \to (B \to C)}$$

• 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 $M =_{op} N$ iff for all C s.t. $C[M]$ and $C[N]$ are closed and of base type, we have:

$$C[M] \longrightarrow^{\star} c \text{ iff } C[N] \longrightarrow^{\star} c$$

.

A model is **fully abstract** when

$$[\![M]\!] = [\![N]\!] \quad \text{iff} \quad M =_{op} N$$

The “if” direction is the difficult one

The full abstraction problem for PCF

In the late 1970's arose the question of finding a fully abstract model of PCF.

• Milner built one as a "term model quotiented by the operational equivalence". The question was then: can we describe this model by other means, as functions of some sorts between suitable domains?

• Candidate 1. Scott continuous functions f (any single piece of the output $f(x)$ may be computed using a finite part of the input x, which one can take be minimal). But Scott pointed out the problematic parallel disjunction satisfying

$$por(\perp, T) = T \qquad por(T, \perp) = T$$

which is not definable in PCF. But adding it to the syntax, Plotkin showed that Scott model "becomes" fullly abstract.

The stable model of PCF

• Candidate 2. Gérard Berry "killed" *por* by introducing stable functions (for a fixed x and a fixed piece of $f(x)$, such a minimal input is unique, and thus minimum).

But he noticed the problematic character of the function $Gustave$ satisfying

$$\begin{array}{l} Gustave(T, F, \perp) = T \\ Gustave(F, \perp, T) = T \\ Gustave(\perp, T, F) = T \end{array}$$

• The next candidates in the list were sequential algorithms, and then HO/AJM.

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.

Proof. If $[\![M]\!] \neq [\![N]\!]$, let h be given by our assumption, and let P be such that $[\![P]\!] = h$, $v_1 = h([\![M]\!])$, $v_2 = h([\![N]\!])$, and $C = P[]$. Then $C[M] \to^* v_1$ and $C[M] \to^* v_2$, and hence $M \not=_{op} N$.

Compact definability + *extensionality* imply definable separability.

Some results and some non-results

Language	Model	Def.	FA
PCF + **por**	$Cont$	**Yes**	**Yes**
PCF + **catch**	$SA \approx (\mathbf{G}_{inn}/=_{op})$	**Yes**	**Yes**
PCF	$PCFBT/=_{op}$	**Yes**	**Yes**
PCF	$\mathbf{G}_{AJM}, \mathbf{G}_{HO}, PCFBT$	**Yes**	*No*
PCF + *control*	$\mathbf{G}_{inn}$	**Yes**	*No*
Idealised Algol	$\mathbf{G}_{wb}$	**Yes**	**Yes**

HO games vs sequential algorithms

Interpret exponential differently:

- HO games are *repetitive*, bool$\rightarrow$bool infinite
- Sequential algorithms have *memory*, bool$\rightarrow$bool finite

For the linear logicians: 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**:

$$\begin{array}{ll} \text{read} & \text{write}(n) \ \ (n \in \omega) \\ \text{OK} & n \ \ (n \in \omega) \end{array}$$

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

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

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

Interaction play with cell:

$$\mathsf{run}_\epsilon\ \mathsf{write}(0)_1\ \mathsf{OK}_1\ \mathsf{read}_1\ 0_1\ \mathsf{write}(1)_1\ \mathsf{OK}_1\ \mathsf{done}_\epsilon$$

Definability through factorisation

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

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 model

Let $\sigma_1, \sigma_2 : A$, let $m_1 \ldots m_{2n} \in \sigma_1 \setminus \sigma_2$. We define $\tau : A \to \mathsf{nat}$ 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)

Abstract interpretation

For decidability, we need finiteness of alphabets. A finite approximation of nat or var is specified by a *finite* partition π of $\mathbb{N}$.

n, write(n) are now $[n]_\pi$, write($[n]_\pi$).

Now, strategies interpreting terms with abstracted types can be *nondeterministic!*

Example of non-determinism arising from abstract interpretation

Suppose that $=$ receives the type $\mathtt{nat}_{\pi_1} \times \mathtt{nat}_{\pi_2} \to \mathtt{bool}$. Then its interpretation contains all the plays $qq_1[n]_{\pi_1}q_2[n]_{\pi_2}T$ and all the plays $qq_1[n]_{\pi_1}q_2[m]_{\pi_2}F$ (for $m \neq n$). Then, say, as soon as the equivalence class of n is not a singleton in either π_1 or π_2, then $n = n$ may execute nondeterministically to T or F. To be completely specific, suppose that $[2]_{\pi_1} = \{2\}$ and $[2]_{\pi_2} = \{0, 2\}$, then $=$ contains both $qq_1[2]_{\pi_1}q_2[2]_{\pi_2}T$ and $qq_1[2]_{\pi_1}q_2[2]_{\pi_2}F$.

A model checking loop (Ghica et al)

For checking a safety property (add a P move abort):

1. Evaluate $[\![M]\!]_\pi$ with respect to some abstract interpretation

2. If $[\![M]\!]_\pi$ does not contain the move abort, conclude that M is **safe**.

a. if all the occurrences of **abort** have been reached nondeterministically, then refine the abstract interpretation accordingly, and go back to step 1.

b. Otherwise, conclude that M is **unsafe**.

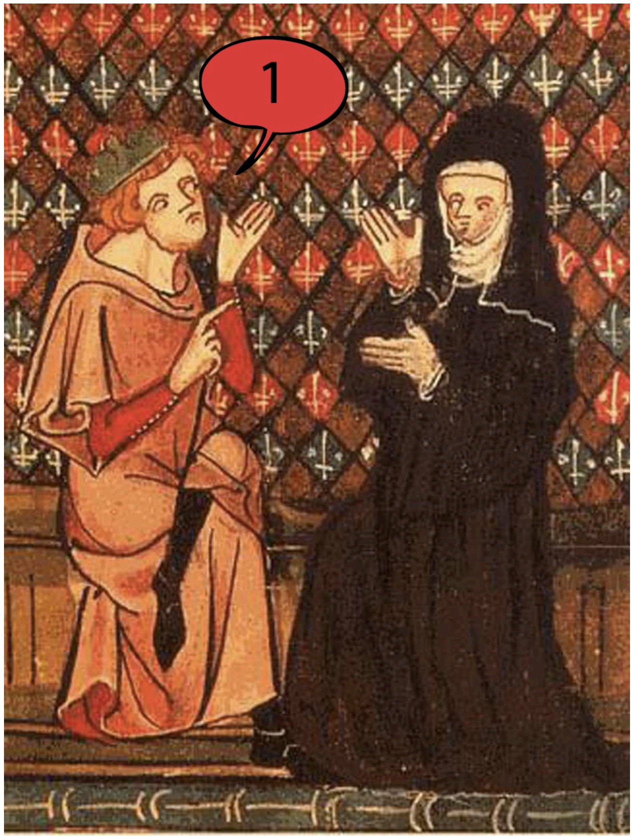
Categorical combinatorics of scheduling and synchronization

Paul-André Melliès

Institut de Recherche en Informatique Fondamentale (IRIF)
CNRS & Université Paris Diderot

ACM Symposium on Principles of Programming Languages
Cascais ⫝̸ POPL'19 ⫝̸ 16 $\longrightarrow$ 19 January 2019

Understanding logic in space and time



What are the principles at work in a dialogue?

Template games

Categorical combinatorics of synchronization

The category of polarities

We introduce the category

$$\pm_{\mathrm{game}}$$

freely generated by the graph

$$\begin{tikzcd} \langle\ominus\rangle \arrow[r, shift right, "P"'] & \langle\oplus\rangle \arrow[l, shift right, "O"'] \end{tikzcd}$$

the category $\pm_{\mathrm{game}}$ will play a fundamental role in the talk

Template games

First idea:

Define a **game** as a category A equipped with a functor

$$\begin{tikzcd} A \arrow[d, "\lambda_A"] \\ \pm_{\mathrm{game}} \end{tikzcd}$$

to the category $\pm_{\mathrm{game}}$ freely generated by the graph

$$\begin{tikzcd} \langle \ominus \rangle \arrow[r, shift right, "P"'] & \langle \oplus \rangle \arrow[l, shift right, "O"'] \end{tikzcd}$$

Inspired by the notion of **coloring** in graph theory

Positions and trajectories

It is convenient to use the following terminology

$$\begin{array}{ccc} \text{objects} & \leftrightarrow & \text{positions} \\ \text{morphisms} & \leftrightarrow & \text{trajectories} \end{array}$$

and to see the category A as an **unlabelled** transition system.

The polarity functor

The polarity functor

$$\lambda_A \quad : \quad A \longrightarrow \pm_{\text{game}}$$

assigns a polarity $\oplus$ or $\ominus$ to every position of the game A.

Definition. A position $a \in A$ is called

Player	when its polarity	$\lambda_A(a) = \oplus$	is positive
Opponent	when its polarity	$\lambda_A(a) = \ominus$	is negative

Opponent moves

Definition. An **Opponent move**

$$m \quad : \quad a^{\oplus} \longrightarrow b^{\ominus}$$

is a trajectory of the game A transported to the edge

$$O \quad : \quad \langle \oplus \rangle \longrightarrow \langle \ominus \rangle$$

of the template category $\pm_{\mathrm{game}}$.

Player moves

Definition. A **Player move**

$$m \quad : \quad a^{\ominus} \longrightarrow b^{\oplus}$$

is a trajectory of the game A transported to the edge

$$P \quad : \quad \langle \ominus \rangle \longrightarrow \langle \oplus \rangle$$

of the template category $\pm_{\mathrm{game}}$.

Silent trajectories

Definition. A **silent move**

$$m \quad : \quad a \longrightarrow b$$

is a trajectory of the game A transported to an identity morphism

$$id_{\langle \oplus \rangle} \quad : \quad \langle \oplus \rangle \longrightarrow \langle \oplus \rangle$$

$$id_{\langle \ominus \rangle} \quad : \quad \langle \ominus \rangle \longrightarrow \langle \ominus \rangle$$

of the template category $\pitchfork_{\mathrm{game}}$.

The template of strategies

The template of strategies

In order to describe the strategies between two games

$$\sigma \quad : \quad A \longrightarrow\!\!\!\!\!\!\!\!\!\!+\quad B$$

we introduce the **template of strategies**

$$\text{⅄}_{\text{strat}}$$

defined as the category freely generated by the graph

$$\begin{tikzcd} \langle \ominus, \ominus \rangle \arrow[r, shift right, "O_s"'] & \langle \oplus, \ominus \rangle \arrow[l, shift right, "P_s"'] \arrow[r, shift right, "P_t"'] & \langle \oplus, \oplus \rangle \arrow[l, shift right, "O_t"'] \end{tikzcd}$$

The template of strategies

Each of the four labels

$$O_s \qquad P_s \qquad O_t \qquad P_t$$

describes a specific kind of Opponent and Player move

O_s	:	Opponent move	played at	the source game
P_s	:	Player move	played at	the source game
O_t	:	Opponent move	played at	the target game
P_t	:	Player move	played at	the target game

which may appear on the interactive trajectory played by a strategy

$$\sigma \quad : \quad A \longrightarrow\!\!\!\!\!\!\!\!+\;\;\;\; B.$$

The template of strategies

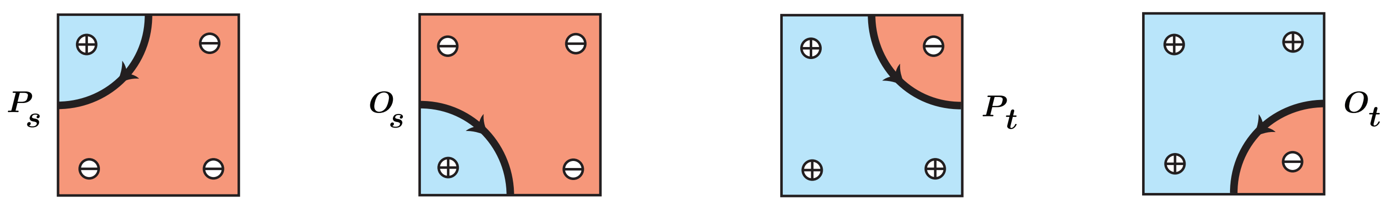
The four generators

$$\begin{tikzcd} \langle \ominus, \ominus \rangle \arrow[r, shift right, "O_s"'] & \langle \oplus, \ominus \rangle \arrow[l, shift right, "P_s"'] \arrow[r, shift right, "P_t"'] & \langle \oplus, \oplus \rangle \arrow[l, shift right, "O_t"'] \end{tikzcd}$$

of the category

$$\pitchfork_{\mathrm{strat}}$$

may be depicted as follows:

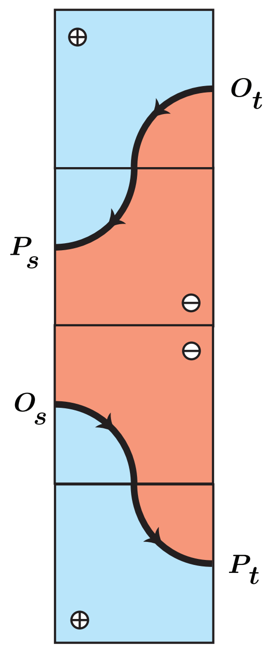


The template of strategies

In that graphical notation, the sequence

$$O_t \cdot P_s \cdot O_s \cdot P_t$$

is depicted as



The template of strategies

The category $\pitchfork_{\text{strat}}$ comes equipped with a span of functors

$$\pitchfork_{\text{game}} \xleftarrow{s=(1)} \pitchfork_{\text{strat}} \xrightarrow{t=(2)} \pitchfork_{\text{game}}$$

defined as the projection $s = (1)$ on the first component:

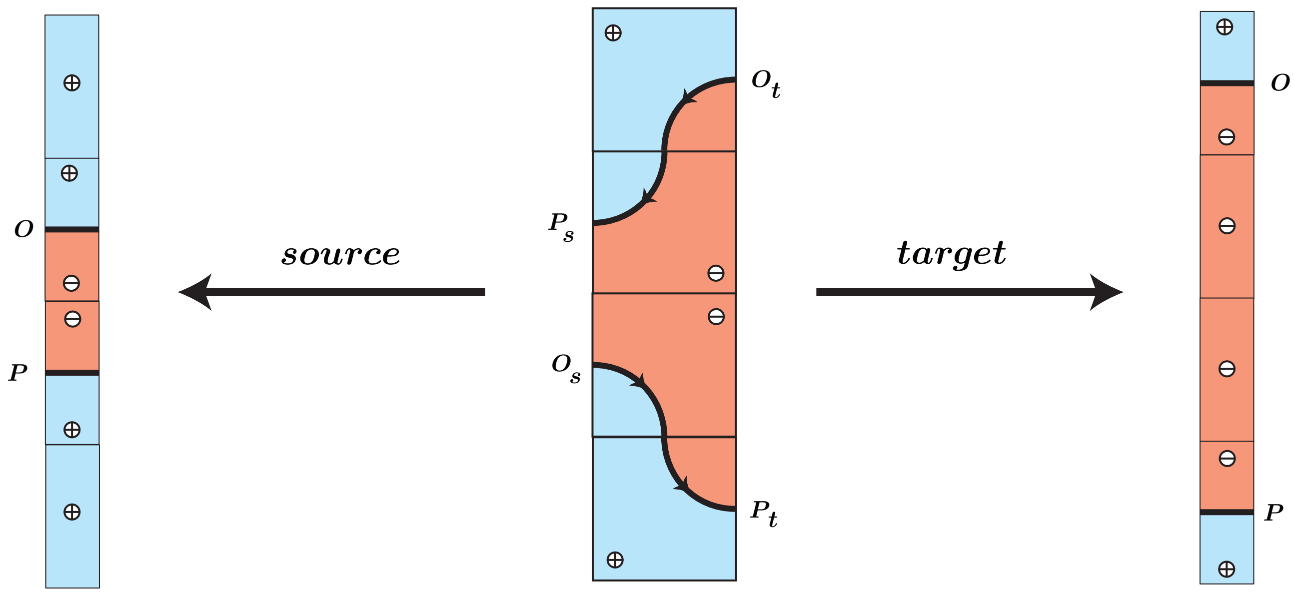
$$\begin{array}{rcl} \langle \ominus, \ominus \rangle & \mapsto & \langle \ominus \rangle \\ \langle \oplus, \ominus \rangle \, , \, \langle \oplus, \oplus \rangle & \mapsto & \langle \oplus \rangle \end{array} \qquad \begin{array}{c} O_s \mapsto P \qquad P_s \mapsto O \\ O_t \, , \, P_t \mapsto id_{\langle \oplus \rangle} \end{array}$$

and as the projection $t = (2)$ on the second component:

$$\begin{array}{rcl} \langle \oplus, \oplus \rangle & \mapsto & \langle \oplus \rangle \\ \langle \ominus, \ominus \rangle \, , \, \langle \oplus, \ominus \rangle & \mapsto & \langle \ominus \rangle \end{array} \qquad \begin{array}{c} O_t \mapsto O \qquad P_t \mapsto P \\ O_s \, , \, P_s \mapsto id_{\langle \ominus \rangle} \end{array}$$

The template of strategies

The two functors s and t are illustrated below:



Strategies between games

Second idea:

Define a **strategy** between two games

$$\sigma \quad : \quad A \longrightarrow\!\!\!\!\!\!\!\!\!\!+\quad B$$

as a **span of functors**

$$A \xleftarrow{\ s\ } S \xrightarrow{\ t\ } B$$

together with a **scheduling functor**

$$S \xrightarrow{\ \lambda_\sigma\ } \pitchfork_{\mathrm{strat}}$$

Strategies between games

making the diagram below commute

$$\begin{tikzcd}
A \arrow[d, "\lambda_A"'] & S \arrow[l, "s"'] \arrow[r, "t"] \arrow[d, "\lambda_\sigma"] & B \arrow[d, "\lambda_B"] \\
\mp_{\mathrm{game}} & \mp_{\mathrm{strat}} \arrow[l, "s"'] \arrow[r, "t"] & \mp_{\mathrm{game}}
\end{tikzcd}$$

Key idea:

Every trajectory $s \in S$ induces a pair of trajectories $s_A \in A$ and $s_B \in B$.

The functor λ_σ describes how s_A and s_B are scheduled together by σ.

Support of a strategy

Terminology. The category S defining the span

$$A \xleftarrow{\ s\ } S \xrightarrow{\ t\ } B$$

is called the **support** of the strategy

$$\sigma \quad : \quad A \longrightarrow B$$

Basic intuition:

$$\boxed{\text{« the support } S \text{ contains the trajectories played by } \sigma \text{ »}}$$

A typical scheduling $B \cdot A \cdot A \cdot B$

A trajectory $s \in S$ of the strategy σ with schedule

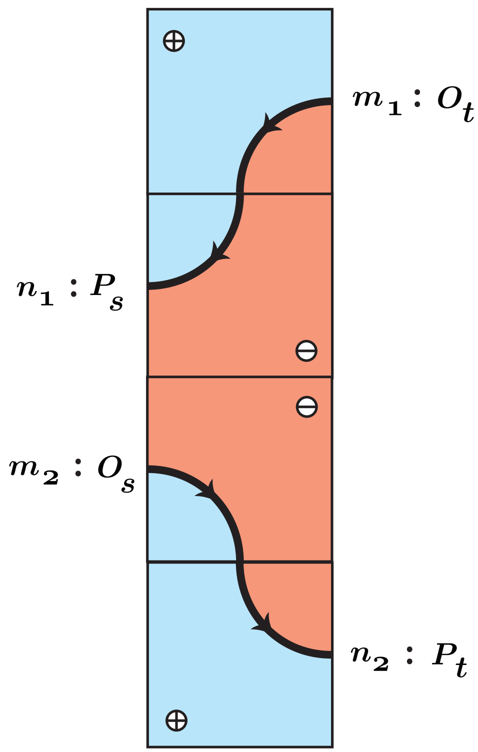
$$\langle \oplus, \oplus \rangle \xrightarrow{O_t} \langle \oplus, \ominus \rangle \xrightarrow{P_s} \langle \ominus, \ominus \rangle \xrightarrow{O_s} \langle \ominus, \oplus \rangle \xrightarrow{P_t} \langle \oplus, \oplus \rangle$$

is traditionally depicted as

	A	$\xrightarrow{\sigma}$	B
first move m_1 of polarity O_t			m_1
second move n_1 of polarity P_s	n_1		
third move m_2 of polarity O_s	m_2		
fourth move n_2 of polarity P_t			n_2

A typical scheduling $B \cdot A \cdot A \cdot B$

Thanks to the approach, one gets the more informative picture:



Simulations

Definition: A **simulation** between strategies

$$\theta \;:\; \sigma \Longrightarrow \tau \;:\; A \mathrel{\longrightarrow\!\!\!\!\!\!\!\!\!+\;\;\;} B$$

is a **functor** from the support of σ to the support of τ

$$\theta \;:\; S \longrightarrow T$$

making the three triangles commute

$$\begin{tikzcd} S \arrow[rr, "\theta"] \arrow[dr, "s"'] & & T \arrow[dl, "s"] \\ & \pm\mathrm{game} & \end{tikzcd} \qquad \begin{tikzcd} S \arrow[rr, "\theta"] \arrow[dr, "\lambda_\sigma"'] & & T \arrow[dl, "\lambda_\tau"] \\ & \pm\mathrm{strat} & \end{tikzcd} \qquad \begin{tikzcd} S \arrow[rr, "\theta"] \arrow[dr, "t"'] & & T \arrow[dl, "t"] \\ & \pm\mathrm{game} & \end{tikzcd}$$

The category of strategies and simulations

Suppose given two games A and B.

The category **Games** (A, B) has **strategies** between A and B

$$\sigma, \tau \quad : \quad A \longrightarrow\!\!\!\!\!\!\!\!\!+\;\;\;\; B$$

as objects and **simulations** between strategies

$$\theta \; : \; \sigma \Longrightarrow \tau \; : \; A \longrightarrow\!\!\!\!\!\!\!\!\!+\;\;\;\; B$$

as morphisms.

The bicategory **Games**

A bicategory of games, strategies and simulations

The bicategory **Games** of games and strategies

At this stage, we want to turn the family of categories

$$\textbf{Games}\,(A, B)$$

into a **bicategory**

$$\textbf{Games}$$

of games and strategies.

The bicategory **Games** of games and strategies

To that purpose, we need to define a composition functor

$$\circ_{A,B,C} \; : \; \textbf{Games}\,(B,C) \times \textbf{Games}\,(A,B) \longrightarrow \textbf{Games}\,(A,C)$$

which composes a pair of strategies

$$\sigma \quad : \quad A \longrightarrow\!\!\!\!\!\!\!\!\!\!+\;\;\;\;\; B \qquad\qquad \tau \quad : \quad B \longrightarrow\!\!\!\!\!\!\!\!\!\!+\;\;\;\;\; C$$

into a strategy

$$\sigma \circ_{A,B,C} \tau \quad : \quad A \longrightarrow\!\!\!\!\!\!\!\!\!\!+\;\;\;\;\; C$$

Composition of strategies

The construction starts by putting the pair of functorial spans side by side:

$$
\begin{tikzcd}
 & S \arrow[dl, "s"'] \arrow[d, "\lambda_\sigma"'] \arrow[dr, "t"] & & T \arrow[dl, "s"'] \arrow[d, "\lambda_\tau"] \arrow[dr, "t"] & \\
A \arrow[d, "\lambda_A"'] & \pitchfork_{\mathrm{strat}} \arrow[dl, "s"] \arrow[dr, "t"'] & B \arrow[d, "\lambda_B"] & \pitchfork_{\mathrm{strat}} \arrow[dl, "s"] \arrow[dr, "t"'] & C \arrow[d, "\lambda_C"] \\
\pitchfork_{\mathrm{game}} & & \pitchfork_{\mathrm{game}} & & \pitchfork_{\mathrm{game}}
\end{tikzcd}
$$

Fine, but how shall one carry on and perform the composition?

The template of interactions

Third idea:

We define the **template of interactions**

$$\text{ⵜ}_{\mathrm{int}}$$

as the category obtained by the pullback diagram below

$$\begin{tikzcd}
& & \text{ⵜ}_{\mathrm{int}} \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & & \\
& \text{ⵜ}_{\mathrm{strat}} \arrow[dl, "s"'] \arrow[dr, "t"'] \arrow[rr, phantom, "pb"] & & \text{ⵜ}_{\mathrm{strat}} \arrow[dl, "s"] \arrow[dr, "t"] & \\
\text{ⵜ}_{\mathrm{game}} & & \text{ⵜ}_{\mathrm{game}} & & \text{ⵜ}_{\mathrm{game}}
\end{tikzcd}$$

The template of interactions

Somewhat surprisingly, the category

$$\pm_{\mathrm{int}}$$

is simple to describe, as the **free category** generated by the graph

$$\langle \ominus, \ominus, \ominus \rangle \underset{O_s}{\overset{P_s}{\leftrightarrows}} \langle \oplus, \ominus, \ominus \rangle \underset{P|O}{\overset{O|P}{\leftrightarrows}} \langle \oplus, \oplus, \ominus \rangle \underset{P_t}{\overset{O_t}{\leftrightarrows}} \langle \oplus, \oplus, \oplus \rangle$$

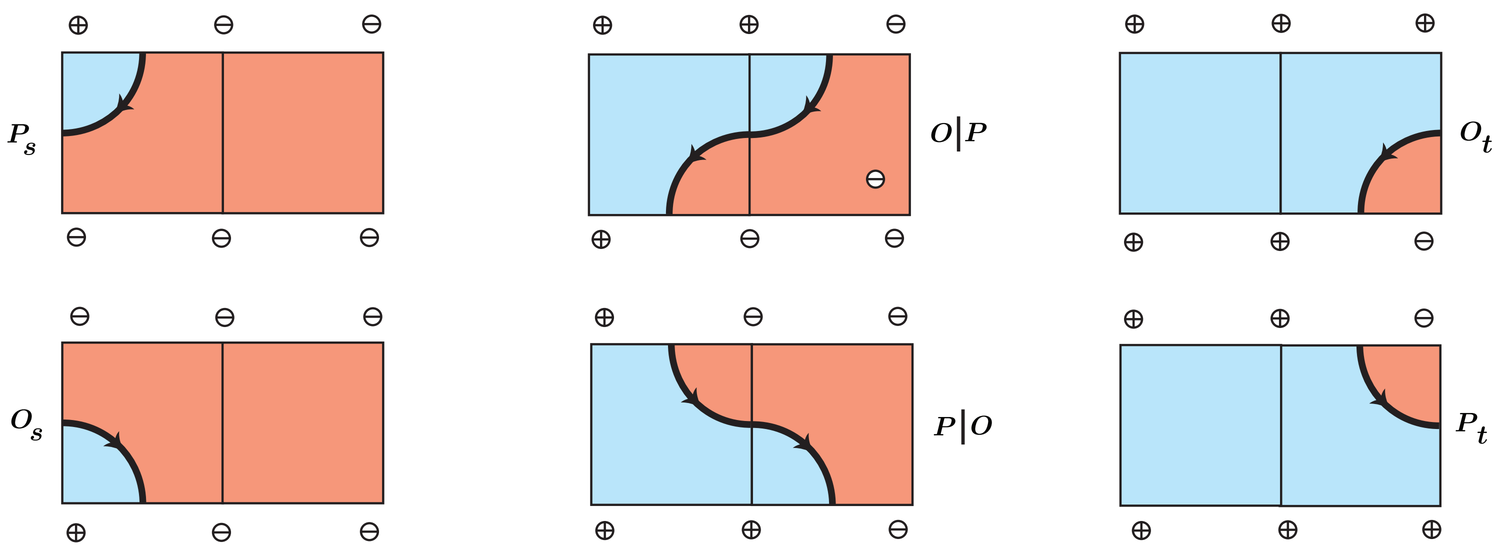
with four states or positions.

The template of interactions

The six generators

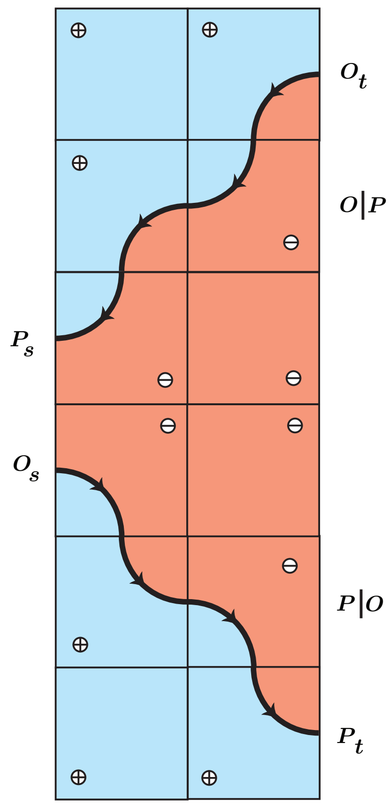
$$\langle\ominus,\ominus,\ominus\rangle \underset{O_s}{\overset{P_s}{\leftrightarrows}} \langle\oplus,\ominus,\ominus\rangle \underset{P|O}{\overset{O|P}{\leftrightarrows}} \langle\oplus,\oplus,\ominus\rangle \underset{P_t}{\overset{O_t}{\leftrightarrows}} \langle\oplus,\oplus,\oplus\rangle$$

may be depicted as follows:



The template of interactions

A typical sequence of interactions is thus depicted as follows:



Key observation

The template $\pitchfork_{\mathrm{int}}$ of interactions comes equipped with a functor

$$\mathit{hide} \quad : \quad \pitchfork_{\mathrm{int}} \longrightarrow \pitchfork_{\mathrm{strat}}$$

which makes the diagram below commute:

$$\begin{tikzcd}
\pitchfork_{\mathrm{strat}} \arrow[d, "(1)"'] & \pitchfork_{\mathrm{int}} \arrow[l, "(12)"'] \arrow[r, "(23)"] \arrow[d, "\mathit{hide}"] & \pitchfork_{\mathrm{strat}} \arrow[d, "(2)"] \\
\pitchfork_{\mathrm{game}} & \pitchfork_{\mathrm{strat}} \arrow[l, "s=(1)"'] \arrow[r, "t=(2)"] & \pitchfork_{\mathrm{game}}
\end{tikzcd}$$

and thus defines a map of span.

Key observation

The functor

$$hide \quad : \quad \pitchfork_{\mathrm{int}} \longrightarrow \pitchfork_{\mathrm{strat}}$$

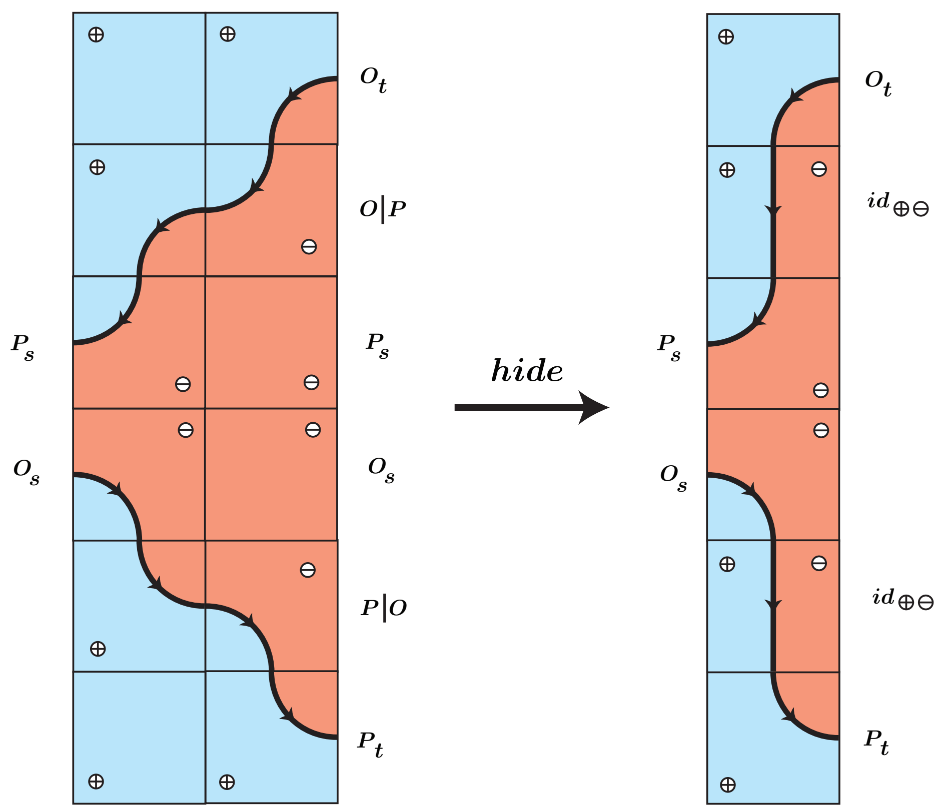
is defined by **projecting** the positions of the interaction category

$$\langle\, \varepsilon_1\, ,\, \varepsilon_2\, ,\, \varepsilon_3 \,\rangle$$

on their first and third components:

$$\begin{array}{rcl@{\qquad}rcl}
\langle \ominus, \ominus, \ominus \rangle & \mapsto & \langle \ominus, \ominus \rangle & O_s \mapsto O_s & & P_s \mapsto P_s \\
\langle \oplus, \ominus, \ominus \rangle\, ,\, \langle \oplus, \oplus, \ominus \rangle & \mapsto & \langle \oplus, \ominus \rangle & O|P\, ,\, P|O & \mapsto & id_{\langle \oplus, \ominus \rangle} \\
\langle \oplus, \oplus, \oplus \rangle & \mapsto & \langle \oplus, \oplus \rangle & O_s \mapsto O_s & & P_s \mapsto P_s
\end{array}$$

Illustration



Composition of strategies

$$
\begin{tikzcd}
 & & S \times_B T \arrow[dl, "\pi_1"'] \arrow[d, dashed, "\lambda_\sigma \parallel \lambda_\tau"] \arrow[dr, "\pi_2"] & & \\
 & S \arrow[dl, "s"'] \arrow[d, "\lambda_\sigma"'] & \pm_{\mathrm{int}} \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] \arrow[dd, "hide" description] & T \arrow[d, "\lambda_\tau"] \arrow[dr, "t"] & \\
A \arrow[d, "\lambda_A"'] & \pm_{\mathrm{strat}} \arrow[dl, "s"'] & & \pm_{\mathrm{strat}} \arrow[dr, "t"] & C \arrow[d, "\lambda_C"] \\
\pm_{\mathrm{game}} & & \pm_{\mathrm{strat}} \arrow[ll, "s"'] \arrow[rr, "t"] & & \pm_{\mathrm{game}}
\end{tikzcd}
$$

Composition of strategies

This definition of composition implements the slogan that

$$\boxed{\text{composition} \quad = \quad \text{synchronization} \;+\; \text{hiding}}$$

What about identities?

There exists a functor

$$\mathit{copycat} \quad : \quad \ddagger_{\mathrm{game}} \longrightarrow \ddagger_{\mathrm{strat}}$$

which makes the diagram commute:

$$\begin{tikzcd}
 & \ddagger_{\mathrm{game}} \arrow[dl, "id"'] \arrow[dr, "id"] \arrow[dd, "copycat"] & \\
\ddagger_{\mathrm{game}} & & \ddagger_{\mathrm{game}} \\
 & \ddagger_{\mathrm{strat}} \arrow[ul, "s=(1)"] \arrow[ur, "t=(2)"'] &
\end{tikzcd}$$

and thus defines a morphism of spans.

What about identities?

The functor

$$copycat \quad : \quad \pm_{\mathrm{game}} \longrightarrow \pm_{\mathrm{strat}}$$

is defined by **duplicating** the positions of the polarity category

$$\langle \varepsilon \rangle$$

in the following way:

$$\begin{array}{rcl} \langle \ominus \rangle & \mapsto & \langle \ominus, \ominus \rangle \\ \langle \oplus \rangle & \mapsto & \langle \oplus, \oplus \rangle \end{array} \qquad \begin{array}{rcl} O & \mapsto & O_t \cdot P_s \\ P & \mapsto & O_s \cdot P_t \end{array}$$

A synchronous copycat strategy

The functor

$$copycat \quad : \quad \pm_{\mathrm{game}} \longrightarrow \pm_{\mathrm{strat}}$$

transports the edge

$$\langle \ominus \rangle \xleftarrow{\;O\;} \langle \oplus \rangle$$

to the trajectory consisting of two moves

$$\langle \ominus, \ominus \rangle \xleftarrow{\;P_s\;} \langle \oplus, \ominus \rangle \xleftarrow{\;O_t\;} \langle \oplus, \oplus \rangle$$

A synchronous copycat strategy

The functor

$$\mathit{copycat} \quad : \quad \pm_{\mathrm{game}} \longrightarrow \pm_{\mathrm{strat}}$$

transports the edge

$$\langle \ominus \rangle \xrightarrow{\ P\ } \langle \oplus \rangle$$

to the trajectory consisting of two moves

$$\langle \ominus, \ominus \rangle \xrightarrow{\quad O_s \quad} \langle \oplus, \ominus \rangle \xrightarrow{\quad P_t \quad} \langle \oplus, \oplus \rangle$$

The identity strategy

Given a game A, the copycat strategy

$$cc_A \quad : \quad A \longrightarrow A$$

is defined as the functorial span

$$A \xleftarrow{\ \textit{identity}\ } A \xrightarrow{\ \textit{identity}\ } A$$

together with the scheduling functor

$$\lambda_{cc_A} \;=\; A \xrightarrow{\ \lambda_A\ } \pitchfork_{\text{game}} \xrightarrow{\ \textit{copycat}\ } \pitchfork_{\text{strat}}$$

Identity strategy

$$
\begin{tikzcd}
 & & A \arrow[ddll, "identity"'] \arrow[d, "\lambda_A"] \arrow[ddrr, "identity"] & & \\
 & & \pm\mathrm{game} \arrow[ddll, "id"'] \arrow[dd, "copycat"] \arrow[ddrr, "id"] & & \\
A \arrow[d, "\lambda_A"'] & & & & A \arrow[d, "\lambda_A"] \\
\pm\mathrm{game} & & \pm\mathrm{strat} \arrow[ll, "s"'] \arrow[rr, "t"] & & \pm\mathrm{game}
\end{tikzcd}
$$

Discovery of an unexpected principle

Key observation: the categories

$$\pitchfork[0] = \pitchfork_{\mathrm{game}} \qquad \pitchfork[1] = \pitchfork_{\mathrm{strat}} \qquad \pitchfork[2] = \pitchfork_{\mathrm{int}}$$

and the span of functors

$$\pitchfork[0] \xleftarrow{\ s\ } \pitchfork[1] \xrightarrow{\ t\ } \pitchfork[0]$$

define an **internal category** in Cat with composition and identity

$$\pitchfork[2] \xrightarrow{\ \mathit{hide}\ } \pitchfork[1] \qquad \pitchfork[0] \xrightarrow{\ \mathit{copycat}\ } \pitchfork[1]$$

As an immediate consequence...

Theorem A. The construction just given defines a **bicategory**

Games

of games, strategies and simulations.

Main technical result of the paper

Theorem B. The bicategory

Games

of games, strategies and simulations is **symmetric monoidal.**

Main technical result of the paper

Theorem C. The bicategory

Games

of games, strategies and simulations is **star-autonomous.**

All these results are based on the same recipe!

One constructs an **internal category** of tensorial schedules

$$\pitchfork^{\otimes}$$

together with a pair of **internal functors**

$$\pitchfork \times \pitchfork \xleftarrow{\quad pick \quad} \pitchfork^{\otimes} \xrightarrow{\quad pince \quad} \pitchfork$$

All these results are based on the same recipe!

One constructs an **internal category** of cotensorial schedules

$$\perp^{\wp}$$

together with a pair of **internal functors**

$$\perp \times \perp \xleftarrow{\ \mathit{pick}\ } \perp^{\wp} \xrightarrow{\ \mathit{pince}\ } \perp$$

All these results are based on the same recipe!

One constructs an **internal functor**

$$reverse \quad : \quad \pitchfork^{op} \longrightarrow \pitchfork$$

which reverses the polarity of every position and move

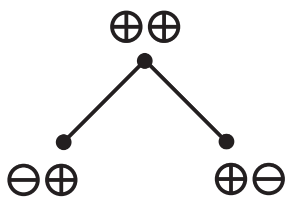
$$\begin{array}{ccc} \oplus & \mapsto & \ominus \\ \ominus & \mapsto & \oplus \end{array} \qquad\qquad \begin{array}{ccc} O & \mapsto & P \\ P & \mapsto & O \end{array}$$

The pick functor

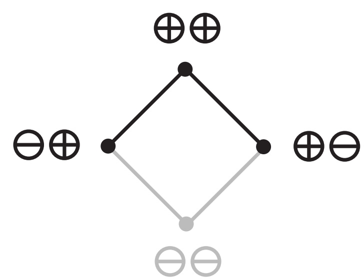
The internal functor

$$pick \quad : \quad \pm^{\otimes} \longrightarrow \pm \times \pm$$

is defined at dimension 0 by the functor:



$$\xrightarrow{pick\,[0]}$$

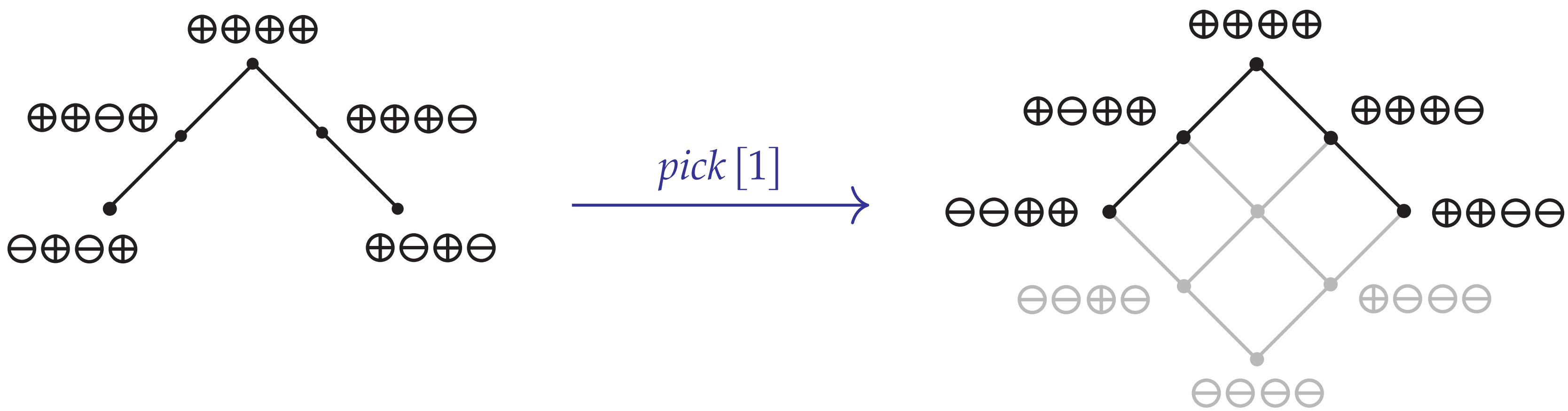


The pick functor

The internal functor

$$\mathit{pick} \quad : \quad \pm^{\otimes} \longrightarrow \pm \times \pm$$

is defined at dimension 1 by the functor:

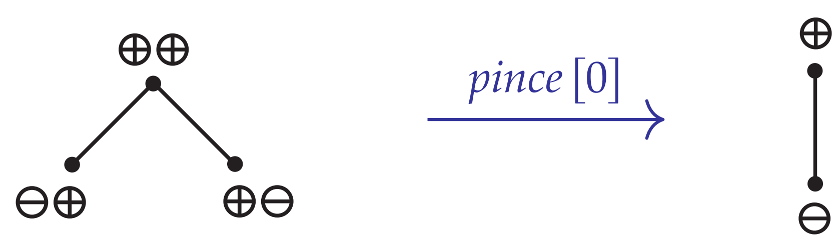


The pince functor

The internal functor

$$pince \quad : \quad \pitchfork^{\otimes} \longrightarrow \pitchfork$$

is defined at dimension 0 by the functor:

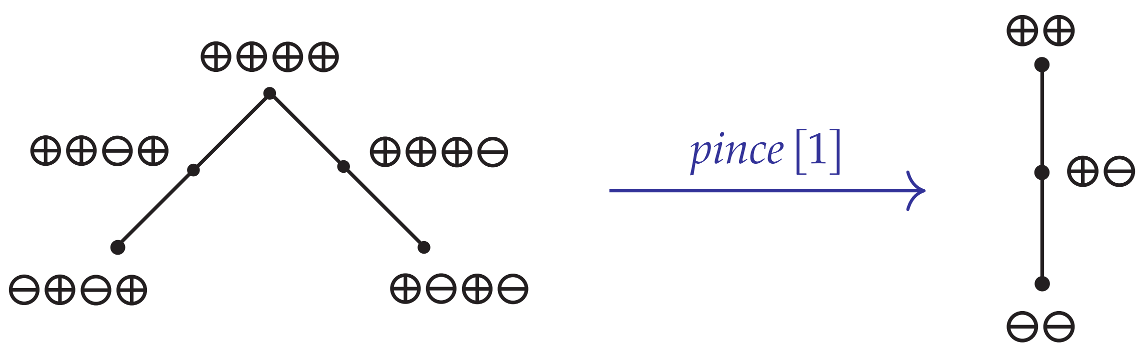


The pince functor

The internal functor

$$pince \quad : \quad \pitchfork^{\otimes} \longrightarrow \pitchfork$$

is defined at dimension 1 by the functor:



Conclusion and future work

▷ games played on **categories** with **synchronous copycats**

▷ an easy recipe to construct **new game semantics**

▷ **three templates** considered in the paper:

$\pitchfork_{\mathrm{alt}}$	alternating games and strategies
$\pitchfork_{\mathrm{conc}}$	concurrent games and strategies
$\pitchfork_{\mathrm{span}}$	functorial spans with no scheduling

► same basic principles in **concurrent separation logic**

► a model of **differential linear logic** based on **homotopy theory**

Selected bibliography

[1] Pierre Castellan and Nobuko Yoshida.
Two Sides of the Same Coin: Session Types and Game Semantics.
POPL'19 – **Capabilities and Session Types session this afternoon!**

[2] Clovis Eberhart and Tom Hirschowitz.
What's in a Game? A Theory of Game Models.
LICS 2018

[3] Russ Harmer, Martin Hyland and PAM.
Categorical Combinatorics for Innocent Strategies.
LICS 2007

[4] PAM and Samuel Mimram.
Asynchronous Games: Innocence Without Alternation.
CONCUR 2007

[5] PAM and Léo Stefanesco.
An Asynchronous Soundness Theorem for Concurrent Separation Logic.
LICS 2018

[6] Sylvain Rideau and Glynn Winskel.
Concurrent Strategies.
LICS 2011

The distributivity law of linear logic

A game semantics of linear logic

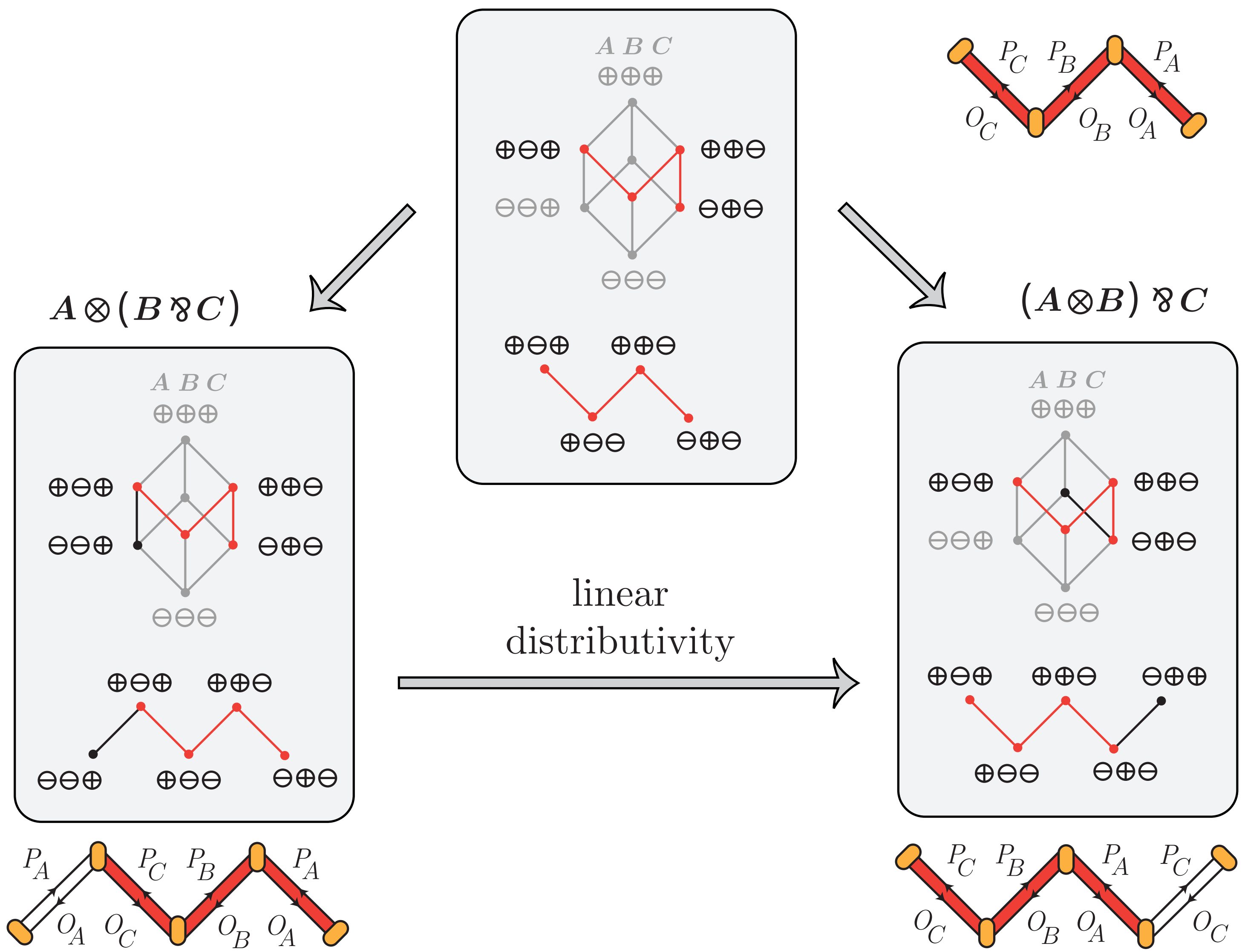
The distributivity law of linear logic

The main ingredient of linear logic

$$\kappa_{A,B,C} \quad : \quad A \otimes (B \parr C) \longrightarrow (A \otimes B) \parr C$$

cannot be interpreted in traditional game semantics.

When one interprets it in template games, here is what one gets...



$A \otimes (B \parr C)$
$(A \otimes B) \parr C$
linear
distributivity

The template of interactions

How the category $\pitchfork_{\mathrm{int}}$ is computed as a pullback

The template of interactions

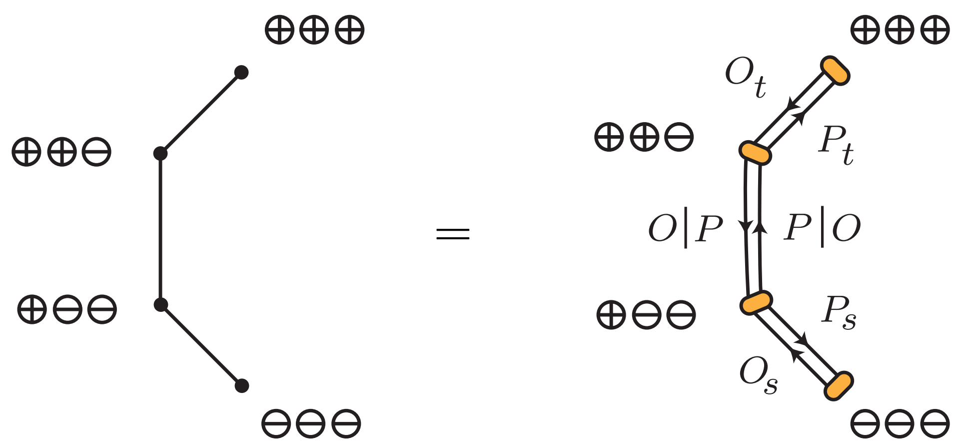
We find illuminating to depict the canonical functor

$$\pitchfork_{\mathrm{int}} \xrightarrow{(1223)} \pitchfork_{\mathrm{strat}} \times \pitchfork_{\mathrm{strat}}$$

induced by the pullback diagram in the following way:

The template of interactions

In order to fully appreciate the diagram, one needs to “fatten” it



in such a way as to recover the template of interactions

$$\langle \ominus,\ominus,\ominus \rangle \underset{O_s}{\overset{P_s}{\leftrightarrows}} \langle \oplus,\ominus,\ominus \rangle \underset{P|O}{\overset{O|P}{\leftrightarrows}} \langle \oplus,\oplus,\ominus \rangle \underset{P_t}{\overset{O_t}{\leftrightarrows}} \langle \oplus,\oplus,\oplus \rangle$$

The template of concurrent games

Templates of concurrent games as commutative monoids

The template of games

The category

$$\pitchfork_{\mathrm{conc}}[0]$$

is generated by the graph

$$\begin{tikzcd} P \arrow[loop left] & \langle * \rangle & O \arrow[loop right] \end{tikzcd}$$

together with the additional equation

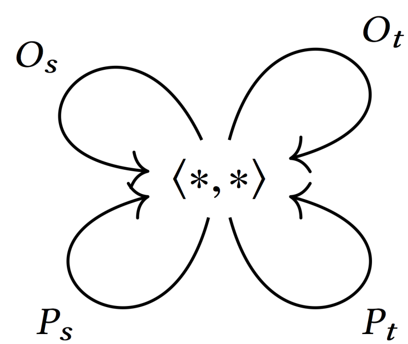
$$O \cdot P = P \cdot O$$

The template of strategies

The category

$$\pm_{\mathrm{conc}}[1]$$

is generated by the graph



together with the six elementary equations

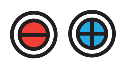
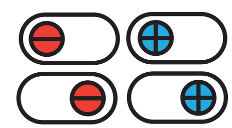
$$\begin{array}{ccc} O_s \cdot P_s = P_s \cdot O_s & O_s \cdot P_t = P_t \cdot O_s & O_s \cdot O_t = O_t \cdot O_s \\ O_t \cdot P_s = P_s \cdot O_t & O_t \cdot P_t = P_t \cdot O_t & P_s \cdot P_t = P_t \cdot O_s \end{array}$$

The templates of games and strategies

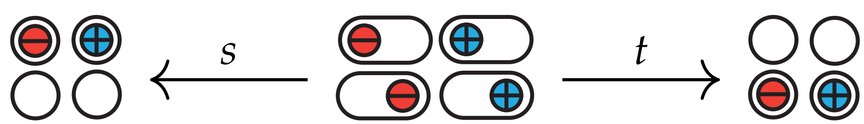
The two templates

$$\pitchfork_{\mathrm{conc}}[0] \qquad\qquad \pitchfork_{\mathrm{conc}}[1]$$

are **commutative monoids** generated by the sets of moves:

$\pitchfork_{\mathrm{conc}}[0] \quad = $  $\qquad\qquad \pitchfork_{\mathrm{conc}}[1] \quad = $ 

The representation is nice to describe the source and target functors:



The template of interactions

When one computes the pullback

$$
\begin{tikzcd}
 & & \pm_{\mathrm{conc}}[2] \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & & \\
 & \pm_{\mathrm{conc}}[1] \arrow[dl, "s"'] \arrow[dr, "t"'] & pb & \pm_{\mathrm{conc}}[1] \arrow[dl, "s"] \arrow[dr, "t"] & \\
\pm_{\mathrm{conc}}[0] & & \pm_{\mathrm{conc}}[0] & & \pm_{\mathrm{conc}}[0]
\end{tikzcd}
$$

one obtains the commutative monoid:

$$\pm_{\mathrm{conc}}[2] \quad =$$