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.

$$\frac{t_1 + St_2 = S(t_1 + St_2)}{t_1 + St_2 = S(t_1 + St_2)} \quad \frac{t_1 + St_2 = S(t_1 + t_2)}{S(t_1 + St_2) = SS(t_1 + t_2)}$$
$$\frac{t_1 + SSt_2 = SS(t_1 + t_2)}{S(t_1 + t_2)}$$

Untyped λ -calculus

The syntax of the untyped λ -calculus (λ -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.

 β -reduction:

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

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

Normal forms in the λ -calculus

Any λ -term has exactly one of the following two forms:

• head normal form $(n \ge 1, p \ge 1)$:

 $\lambda x_1 \cdots x_n \cdot x M_1 \cdots M_p$

• head redex $(n \ge 0, p \ge 1)$:

 $\lambda x_1 \cdots x_n . (\lambda x.M) M_1 \cdots M_p$

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

Böhm trees

$$\begin{split} \omega(M) = \begin{cases} \Omega & \text{if } M = \lambda x_1 \cdots x_n . (\lambda x.P) M_1 \cdots M_p \\ \lambda x_1 \cdots x_n . x \omega(M_1) \cdots \omega(M_p) & \text{if } M = \lambda x_1 \cdots x_n . x M_1 \cdots M_p \\ \end{array} \\ \hline M_1 \leq N_1 \cdots M_p \leq N_p \\ \hline \overline{\Omega \leq M} & \overline{\lambda x_1 \cdots x_n . x M_1 \cdots M_p \leq \lambda x_1 \cdots x_n . x N_1 \cdots N_p} \\ \end{split}$$
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

$$\frac{M_1 \dots M_p}{\lambda y_1 \cdots y_p \cdot y} \dots M_n}{\lambda x_1 \cdots x_n \cdot x}$$

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

Simply typed λ -calculus

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

Thus every type write s uniquely as $A_1 \rightarrow \cdots \rightarrow A_n \rightarrow C$.

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

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

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

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 $e ::= nil \mid B \bullet e$
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.

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

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

Executing Böhm trees (illustration)

$$\begin{array}{lll} 2' & u(\lambda x.u(\lambda y.x)) & \langle u \stackrel{2}{\leftarrow} (\lambda r.r(r(z)))\rangle[\] = \rho_{0} \\ 3' & r(r(z)) & \langle r \stackrel{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 \stackrel{4}{\leftarrow} (\lambda r.r(r(z)))\rangle[\]) = \rho_{2} \\ 5' & r(r(z)) & \langle r \stackrel{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 \stackrel{6}{\leftarrow} (r(z))\rangle[\rho_{1}] \bullet \rho_{0}) = \rho_{4} \\ 7' & r(z) & \langle \rangle[\rho_{4}] \bullet(\rho_{1} = \langle r \stackrel{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 \stackrel{8}{\leftarrow} (\lambda r.r(r(z)))\rangle[\]) = \rho_{6} \\ 9' & r(r(z)) & \langle r \stackrel{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 \stackrel{10}{\leftarrow} (z)\rangle[\rho_{5}] \bullet \rho_{0}) = \rho_{8} \\ 11' & z & \langle \rangle[\rho_{8}] \bullet \rho_{5} \end{array}$$

The language PCF

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

n			: Nat	$(n\in\omega)$
T, .	F		: Bool	
succ, pred			: $Nat \rightarrow Nat$	
zero?			$: Nat \rightarrow Bool$	
if	then	else	: $Bool \rightarrow Nat \rightarrow Nat \rightarrow Nat$	
if	then	else	: $Bool \rightarrow Bool \rightarrow Bool \rightarrow Bool$	
Ω			: A	for all A
Y			$: (A \to A) \to A$	for all A

Operational semantics of PCF

 $\begin{array}{ll} (\lambda x.M)N \to M[x \leftarrow N] & YM \to M(YM) \\ zero?(0) \to T & zero?(n+1) \to F \\ succ(n) \to n+1 & pred(n+1) \to n \\ if \ T \ then \ N \ else \ P \to N & if \ F \ then \ N \ else \ P \to P \end{array}$

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

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

PCF Böhm trees

$$\begin{split} M &::= \lambda \vec{x} : \vec{A}.W \qquad W ::= v \mid \operatorname{case} y M_1 \cdots M_n \ [v_1 \to W_1, \dots, v_k \to W_k] \\ & \frac{v : C}{\Gamma \vdash v : C} \\ \\ & \Gamma \bullet y = A_1 \to \dots \to A_n \to C \quad v_1, \dots, v_k : C \\ & \frac{\Gamma \vdash M_1 : A_1 \ \dots \ \Gamma \vdash M_n : A_n \quad \Gamma \vdash W_1 : C_1 \ \dots \ \Gamma \vdash W_k : C_1}{\Gamma \vdash \operatorname{case} y M_1 \cdots M_n \ [v_1 \to W_1, \dots, v_k \to W_k] : C_1} \\ & \frac{\Gamma, \vec{x} : \vec{A} \vdash W : C}{\Gamma \vdash (\lambda \vec{x} : \vec{A}.W) : \vec{A} \to C} \end{split}$$

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

PCF Böhm trees as strategies

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

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.

$$\left\{\begin{array}{l} \{\epsilon\} \\ \{\epsilon, qn\} \end{array}\right. \qquad \left\{\begin{array}{l} \bot \\ n \end{array}\right.$$

Product of arenas

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

$$\begin{cases} \{\epsilon\} \\ \{\epsilon, q_1 m_1\} \\ \{\epsilon, q_2 n_2\} \\ \{\epsilon, q_1 m_1, q_2 n_2\} \end{cases} \qquad \begin{cases} (\bot, \bot) \\ (m, \bot) \\ (\bot, n) \\ (m, n) \end{cases}$$

Function types $(nat_1 \rightarrow nat_{\epsilon})$

$$q_{\epsilon}q_{1} \begin{cases} 0_{1}3_{\epsilon} \\ 3_{1}6_{\epsilon} \end{cases} \lambda x. \text{ case } x \begin{cases} 0 \to 3 \\ 3 \to 6 \end{cases}$$

Note the inversion of polarity for **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 descirbed by the set of its even-length branches, which are called plays:

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

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

 λx . case $x [4 \rightarrow \text{case } x [3 \rightarrow 2]]$ 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

 $(\mathsf{nat}_{11} \rightarrow \mathsf{nat}_1) \rightarrow \mathsf{nat}_{\epsilon}$

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

As a strategy:

$$\lambda f. \operatorname{case} f \begin{cases} (3) \\ 4 \to 7 \\ 6 \to 9 \end{cases} \quad q_{\epsilon} q_{1} \begin{cases} q_{11} 3_{11} \\ 4_{1} 7_{\epsilon} \\ 6_{1} 9_{\epsilon} \end{cases}$$

Pointers

 $Kierstead_1 = \lambda f. case f(\lambda x. case f(\lambda y. case x))$ $Kierstead_2 = \lambda f. case f(\lambda x. case f(\lambda y. case y))$

$$q_{\epsilon}q_{1} \begin{cases} q_{11}q_{1} \\ q_{11}q_{1} \\ q_{11}q_{1} \\ T_{1}T_{11} \\ F_{1}F_{11} \\ F_{1}F_{11} \\ T_{1}T_{\epsilon} \\ F_{1}F_{\epsilon} \end{cases}$$

 $(\mathsf{type} ((\mathsf{bool}_{111} \rightarrow \mathsf{bool}_{11}) \rightarrow \mathsf{bool}_{1}) \rightarrow \mathsf{bool}_{\epsilon}) \text{ } \textbf{AMBIGUOUS!}$

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).

$$Kierstead_{I} = q_{\epsilon}[q_{1}, \stackrel{0}{\leftarrow}] \begin{cases} q_{11}[q_{1}, \stackrel{1}{\leftarrow}] \\ q_{11}[q_{1}, \stackrel{1}{\leftarrow}] \\ T_{1}[T_{11}, \stackrel{1}{\leftarrow}] \\ T_{1}[T_{11}, \stackrel{1}{\leftarrow}] \\ F_{1}[F_{11}, \stackrel{1}{\leftarrow}] \\ F_{1}[F_{11}, \stackrel{1}{\leftarrow}] \\ F_{1}[F_{11}, \stackrel{1}{\leftarrow}] \\ F_{1}[F_{\epsilon}, \stackrel{1}{\leftarrow}] \\ F_{1}[F_{\epsilon}, \stackrel{1}{\leftarrow}] \\ F_{1}[F_{\epsilon}, \stackrel{1}{\leftarrow}] \\ Kierstead_{2} = q_{\epsilon}[q_{1}, \stackrel{0}{\leftarrow}] \begin{cases} q_{11}[q_{1}, \stackrel{1}{\leftarrow}] \\ q_{11}[q_{1}, \stackrel{1}{\leftarrow}] \\ T_{1}[T_{\epsilon}, \stackrel{1}{\leftarrow}] \\ F_{1}[F_{11}, \stackrel{1}{\leftarrow}] \\ F_{1}[F_{\epsilon}, \stackrel{1}{\leftarrow}] \\ F_{1}[F_{\epsilon}, \stackrel{1}{\leftarrow}] \\ F_{1}[F_{\epsilon}, \stackrel{1}{\leftarrow}] \end{cases} \end{cases}$$

Inserting the implicit pointers in our previous examples

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

$$q_{\epsilon}q_{1} \begin{cases} q_{11}3_{11} \\ 4_{1}7_{\epsilon} \\ 6_{1}9_{\epsilon} \end{cases} \qquad q_{1}q_{11} \begin{cases} 0_{11}3_{1} \\ 3_{11}6_{1} \end{cases}$$

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

Game abstract machine (GAM) for PCF Böhm trees

The machine takes as input a strategy σ of type, say, $A \rightarrow C$ 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 $1, 2', 2, 3', 3, \ldots$, with n, (n + 1)' pointing in σ for n odd and in τ for n even. The machine has two rules:

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

• Rule 2. Once a (Player) step (q' has been performed, the next step 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 q' points to p, then the next step q is played over p'. The name m of the move played at time q' dictates which branch to choose from p'. If p is 1, then play q at the root of τ .

GAM: an example of execution

$$q_{\epsilon}[q_{1}, \stackrel{0}{\leftarrow}] \begin{cases} q_{11}[3_{11}, \stackrel{0}{\leftarrow}] \\ 4_{1}[7_{\epsilon}, \stackrel{1}{\leftarrow}] \\ 6_{1}[9_{\epsilon}, \stackrel{1}{\leftarrow}] \end{cases} q_{1}[q_{11}, \stackrel{0}{\leftarrow}] \begin{cases} 0_{11}[3_{1}, \stackrel{1}{\leftarrow}] \\ 3_{11}[6_{1}, \stackrel{1}{\leftarrow}] \end{cases} \text{ interaction} \\ (q_{\epsilon}, 1)[q_{1}, \stackrel{0}{\leftarrow}] \begin{cases} 2' \\ \langle q_{11}, 3\rangle[3_{11}, \stackrel{0}{\leftarrow}] \\ \langle 6_{1}, 5\rangle[9_{\epsilon}, \stackrel{1}{\leftarrow}] \end{cases} \end{cases} q_{1}[q_{11}, \stackrel{0}{\leftarrow}] \begin{cases} (q_{11}, 3)[3_{11}, \stackrel{0}{\leftarrow}] \\ (\delta_{1}, 5)[9_{\epsilon}, \stackrel{1}{\leftarrow}] \end{cases}$$

Executing Kierstead against

 $\lambda g. \text{case } g(\text{case } gT [T \to T, F \to F]) [T \to F, F \to T]$ $q_{1}[q_{11}, \stackrel{0}{\leftarrow}] \left\{ \begin{array}{c} q_{111}[q_{11}, \stackrel{1}{\leftarrow}] \\ q_{111}[q_{11}, \stackrel{1}{\leftarrow}] \\ T_{11}[T_{111}, \stackrel{1}{\leftarrow}] \\ F_{11}[F_{111}, \stackrel{1}{\leftarrow}] \\ F_{11}[F_{111}, \stackrel{1}{\leftarrow}] \\ F_{11}[T_{111}, \stackrel{1}{\leftarrow}] \end{array} \right.$ $Kierstead_{1} = q_{\epsilon}[q_{1}, \stackrel{0}{\leftarrow}] \begin{cases} q_{11}[q_{1}, \stackrel{1}{\leftarrow}] \\ q_{11}[q_{1}, \stackrel{1}{\leftarrow}] \\ T_{1}[T_{11}, \stackrel{1}{\leftarrow}] \\ F_{111}[F_{11}, \stackrel{1}{\leftarrow}] \\ F_{111}[F_{11}, \stackrel{1}{\leftarrow}] \\ F_{11}[F_{11}, \stackrel{1}{\leftarrow}] \\ F_{1}[F_{\epsilon}, \stackrel{1}{\leftarrow}] \\ F_{1}[F_{\epsilon}, \stackrel{1}{\leftarrow}] \end{cases}$

$$\langle q_{\epsilon}, 1 \rangle [q_{1}, \stackrel{0}{\leftarrow}] \begin{cases} \langle q_{11}, 3 \rangle [q_{1}, \stackrel{1}{\leftarrow}] \\ \langle q_{11}, 7 \rangle [q_{1}, \stackrel{1}{\leftarrow}] \\ \langle q_{11}, 7 \rangle [q_{1}, \stackrel{1}{\leftarrow}] \\ \langle q_{11}, 9 \rangle [q_{111}, \stackrel{1}{\leftarrow}] \\ \langle q_{11}, 9 \rangle [q_{111}, \stackrel{1}{\leftarrow}] \\ \langle T_{1}, 13 \rangle [T_{11}, \stackrel{1}{\leftarrow}] \\ \langle T_{11}, 10 \rangle [F_{111}, \stackrel{1}{\leftarrow}] \\ \langle T_{11}, 10 \rangle [F_{111}, \stackrel{0}{\leftarrow}] \\ \langle F_{11}, F_{11}, \stackrel{1}{\leftarrow}] \\ \langle F_{11}, F_{11}, \stackrel{1}{\leftarrow}] \\ \langle F_{11}, F_{11}, 10 \rangle [F_{11}, \stackrel{1}{\leftarrow}] \\ \langle F_{11}, F_{11}, \stackrel{1}{\leftarrow}] \\ \langle F_{11}, F_{11},$$

Compare with the stack-free environment machine

$$\begin{split} & \underbrace{\left(\begin{array}{c} u\\ \langle q_{\epsilon}, 1 \rangle [q_{1}, \stackrel{0}{\leftarrow}] \right)}_{\langle q_{\epsilon}, 1 \rangle [q_{1}, \stackrel{0}{\leftarrow}]} \begin{cases} \underbrace{\left(\lambda x \quad u\\ \langle q_{11}, 3 \rangle [q_{1}, \stackrel{1}{\leftarrow}] \right)}_{\langle q_{11}, 3 \rangle [q_{1}, \stackrel{1}{\leftarrow}]} \begin{cases} \underbrace{\left(\lambda y \quad x\\ \langle q_{11}, 5 \rangle [q_{111}, \stackrel{1}{\leftarrow}] \right)}_{\langle q_{11}, 7 \rangle [q_{1}, \stackrel{1}{\leftarrow}]} \\ & \underbrace{\left(\lambda x \quad u\\ \langle q_{11}, 7 \rangle [q_{1}, \stackrel{1}{\leftarrow}] \right)}_{\langle q_{11}, 9 \rangle [q_{111}, \stackrel{1}{\leftarrow}]} \\ & \underbrace{\left(\lambda r \quad r\\ \langle q_{1}, 2 \rangle [q_{11}, \stackrel{0}{\leftarrow}] \right)}_{\langle q_{111}, \stackrel{0}{\leftarrow}]} \begin{cases} \underbrace{\left(r \quad r\\ \langle q_{111}, 6 \rangle [q_{11}, \stackrel{0}{\leftarrow}] \right)}_{\langle q_{111}, 10 \rangle [z, \stackrel{-}{\leftarrow}]} \\ & \underbrace{\left(\lambda r \quad r\\ \langle q_{1}, 4 \rangle [q_{11}, \stackrel{0}{\leftarrow}] \right)}_{\langle q_{11}, 8 \rangle [q_{11}, \stackrel{0}{\leftarrow}]} \end{split}$$

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 2, 4 and 8, and 3 and 7.)

Steps versus nodes versus moves

One should not confuse

- the successive steps of the machine (in bold), that are numbers ${\bf n}$ or ${\bf n}^\prime,$
- 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 $(nat_{11} \times nat_{12} \rightarrow nat_1) \rightarrow nat_{\epsilon}$ 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.

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

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

To produce the branch $q_{\epsilon}q_26_28_{\epsilon}$, "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

 $\begin{array}{c} \vdash q \\ q \vdash n \end{array}$

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:

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

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:

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

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 $A_1 \rightarrow \ldots \rightarrow A_n \rightarrow C$ has as only initial move the unique initial move of (the arena interpreting) C. So we have trees.

An automaton for $L_{A \rightarrow B}$

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

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

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 for legal interactions



These automata play a central role in the recent template games of Melliès (Categorical combinatorics of scheduling and synchronization in game semantics, POPL'19).

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 :

 $\tau \circ \sigma = \{ v \mid \exists u \in int(A, B, C) \ v = u \upharpoonright_{A,C}, u \upharpoonright_{A,B} \in \sigma, v \upharpoonright_{B,C} \in \tau \} .$ (we say that u is a witness of v).

In the vocabulary of concurrency theory:

(|| composition) + hiding

Associativity of composition

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

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

Proposition. If σ, τ, v are strategies of $A \to B, B \to C$, and $C \to D$, respectively, then $v \circ (\tau \circ \sigma) = (v \circ \tau) \circ \sigma$.

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:

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

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 uis a sequence of blocks of the form $mb_1 \dots b_k n$ 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 $u = u'mb_1 \dots b_k n$ where $mb_1 \dots b_k n$ is as above, then u' is also a witness.

We have always $id \circ \sigma = \sigma$ and $\sigma \circ id = \sigma$.

Determinism, innocence

 \bullet Recall that a strategy σ is called deterministic when

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

• The P view $\lceil s \rceil$ of a play s is defined as follows:

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:

 $\lambda x...\lambda y.xM_1M_2$

- 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

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

is



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

$$\lceil u \rceil = q_{\epsilon}[q_1, \stackrel{0}{\leftarrow}]\mathbf{6_1}[9_{\epsilon}, \stackrel{1}{\leftarrow}]$$

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 meager strategy in $A \rightarrow (B_1 \times B_2)$ is a forest which can be split in two forests (according to where the roots of the views come from). (In the non-innocent setting, we have a priori only a monoidal and not a cartesian structure.)

• 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

•

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

• The "only if" direction is a consequence of computational adequacy. If $\vdash P : C$ (closed term of basic type), then

$$\llbracket P \rrbracket = c \quad \text{iff} \quad P \longrightarrow^{\star} c$$

• 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

$$por(\bot, T) = T$$
 $por(T, \bot) = T$

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

A witness of non full abstrraction of the continuous model

Plotkin first proved that the continuous model is not fully abstract. He gave the following terms, both of type

$$(Bool \rightarrow Bool \rightarrow Bool) \rightarrow Bool$$

 $M_1 = \lambda g.$ if P_1 then if P_2 then if P_3 then Ω else T else Ω else Ω $M_2 = \lambda g.$ if P_1 then if P_2 then if P_3 then Ω else F else Ω else Ω where $P_1 = gT\Omega$, $P_2 = g\Omega T$, and $P_3 = gFF$. These terms are designed in such a way that:

$$T = [[M_1]](por) \neq [[M_2]](por) = F$$
.

On the other hand $M_1 =_{op} M_2$.

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

 $Gustave(T, F, \bot) = T$ $Gustave(F, \bot, T) = T$ $Gustave(\bot, T, F) = T$

• 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 $A \rightarrow$ bool such that $hf \neq hg$, then it is fully abstract.

Proof. If $\llbracket M \rrbracket \neq \llbracket N \rrbracket$, let *h* be given by our assumption, and let *P* be such that $\llbracket P \rrbracket = h$, $v_1 = h(\llbracket M \rrbracket)$, $v_2 = h(\llbracket N \rrbracket)$, and C = P[]. Then $C[M] \to^* v_1$ and $C[M] \to^* v_2$, and hence $M \neq_{op} N$.

Compact definability + *extensionality* imply definable separability.

Some full abstraction results and some non-results

Language	Model	Def.	FA
PCF+por	$Cont \\ SA \approx (\mathbf{G}_{inn} /=_{op})$	Yes	Yes
PCF+catch		Yes	Yes
PCF	$(PCFBT/=_{op}) \approx (\mathbf{G}_{AJM}/=_{op}) \approx (\mathbf{G}_{HO}/=_{op})$	Yes	Yes
PCF	$ \begin{array}{l} \mathbf{G}_{AJM}, \mathbf{G}_{HO}, PCFBT \\ \mathbf{G}_{inn} \\ \mathbf{G}_{wb} \end{array} $	Yes	No
PCF+ <i>control</i>		Yes	No
IA		Yes	Yes

HO games vs sequential algorithms

Interpret exponential differently:

- HO games are *repetitive*, bool→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**:

read 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)

Idealised Algol

Idealised Algol (as defined by Abramsky and McCusker) is PCF augmented with the types **comm** and **var**, assignment, and sequencing (and dereferencing).

To keep a cartesian category (cf. previous slide where our proof relied on views), Abramsky and McCusker introduce a new component in the definition of an arena A: in addition to moves, polarity and enabling, one specifies a set P_A of authorized plays, called *valid* plays (hence a subset of the set of legal plays). We omit the details.

Cell discipline enforced by interaction

$$\llbracket (x := 0); (x := x + 1) \rrbracket =$$

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

Interaction play with cell:

 $run_{\epsilon} write(0)_1 OK_1 read_1 O_1 write(1)_1 OK_1 done_{\epsilon}$

Definability through factorisation

Every strategy σ : A can be written as (τ cell) for some *innocent* strategy τ : 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 model

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

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 $\operatorname{nat}_{\pi_1} \times \operatorname{nat}_{\pi_2} \to \operatorname{bool}$. 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 $\llbracket M \rrbracket_{\pi}$ with respect to some abstract interpretation
- 2. If $\llbracket M \rrbracket_{\pi}$ 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**.