# Sémantique des jeux asynchrones et réécriture 2-dimensionnelle 

Soutenance de thèse de doctorat

Samuel Mimram<br>Laboratoire PPS (CNRS - Université Paris Diderot)<br>$1^{\text {er }}$ décembre 2008

## A program is

a text
in a programming language.

```
/* kernel entry point function, called from assembler code */
voi d kernel entry(unsi gned I ong magic,unsi gned I ong addr)
{
    /* we will need a multiboot info pointer */
    multiboot_info_t *mbi;
    memory_map_t *mmap;
    unsi gned l ōng mmap size;
    unsi gned l ong block size = 1<<20; /* assume at least 1MB upper memory */
    static char * argv[]={ "ocaml", NULL };
    value *val;
    /* check the multiboot-compliant magic number */
    if(magic!=MULTIBOOT_BOOTLOADER_MAGIC) return;
    /* now set mbi to the right address */
    mbi = (multiboot_info_t *) addr;
    mmap = (memory_map_t *) mbi->mmap_addr;
    mmap_size = mbi->mmap_length;
    if(CHECK_FLAG(mbi->flags,6))
        while (mmap_size>0)
        {
            if ((void *) mmap->base_addr low == &_begin) {
                block_size = mmap->length_र्रow;
                break;
            }
            mmap_size-=si zeof (*mmap);
            mmap++;
        }
    if (& begin+block size<& bss end) {
        c_printf("not enough memory for funk: %lu upper memory needed",&__bss_end-&_begin);
        while(1);
    }
    /* clean bss */
    memset(&__bss_start,0,&_bss_end-&__bss_start);
    /* gcc seems bugged and move some of the following affectations *before* memset... */
    wmb();
    /* TODO: more accurate value */
    mem_size = (unsi gned | ong)(&_begin+block_size);
    heap = & end;
    heaplimi\overline{t}=&_begin+block_size;
    last_seen = heap + 2 + 4*si zeof (voi d*);
    /* then we can setup the kernel */
    setup_kernel();
    /* We also setup the memory */
    setup_memory(heap + HEAP_OFFSET, heaplimit);
    /* then call the caml startup function */
    caml startup(arqv);
```


## A program is

$$
\begin{gathered}
\text { a text } \\
\text { in a programming language. }
\end{gathered}
$$

We want to give a meaning to this language!

## The Curry-Howard correspondence

- Typing programs can avoid some errors

$$
1: \text { int }
$$

## The Curry-Howard correspondence

- Typing programs can avoid some errors

$$
\text { fun } x \rightarrow x+1 \quad: \quad \text { int } \Rightarrow \text { int }
$$

## The Curry-Howard correspondence

- Typing programs can avoid some errors

$$
\text { fun } x \rightarrow \text { not } x \text { : bool } \Rightarrow \text { bool }
$$

## The Curry-Howard correspondence

- Typing programs can avoid some errors

$$
\text { fun } x \rightarrow \text { not } x: \text { bool } \Rightarrow \text { bool }
$$

- The Curry-Howard correspondence:

$$
\begin{aligned}
\text { type of a program } & =\text { formula } \\
\text { program } & =\text { proof of its type }
\end{aligned}
$$

- Studying programs is the same as studying proofs.


## Denotational semantics

## A model interprets

- a type $A$ as a computation space $\llbracket A \rrbracket$
- a program $f: A \Rightarrow B$ as a transformation $\llbracket f \rrbracket: \llbracket A \rrbracket \rightarrow \llbracket B \rrbracket$

$$
A \Rightarrow B \quad: \quad f \quad \sim>\quad \llbracket f \rrbracket \quad: \quad \llbracket A \rrbracket \rightarrow \llbracket B \rrbracket
$$

## Denotational semantics

## A denotational model interprets

- a type $A$ as a computation space $\llbracket A \rrbracket$
- a program $f: A \Rightarrow B$ as a transformation $\llbracket f \rrbracket: \llbracket A \rrbracket \rightarrow \llbracket B \rrbracket$


$$
\text { denotational semantics }=\text { program invariants }
$$

Here, a program will be modeled by its interactive behavior
i.e. by the way it reacts to information provided by its environment.

Here, a program will be modeled by its interactive behavior
i.e. by the way it reacts to information provided by its environment.

```
(fun x not x)false }\rightsquigarrow tru
(fun x not x)true }\rightsquigarrow fals
```

$\Rightarrow$ Game Semantics!

## Outline

Programs do actions in a particular order...

We study the causality induced by programs in two frameworks:
(1) Asynchronous Games
(2) 2-Dimensional Rewriting

## Part I

## Causality in Asynchronous Games

## Game semantics

- A type $A$ is interpreted as a game.
- A program $f: A$ is interpreted as a strategy playing on the game associated to $A$.


## Event structures

Definition
An event structure $(E, \leq, \#)$ is

- a set $E$ of events (or moves)
- equipped with a partial order $\leq$ of causal dependency
- and a binary relation \# of incompatibility.



## Games

## Definition

A game is

- an event structure ( $M, \leq, \#$ )
- equipped with a polarisation function $\lambda: M \rightarrow\{-1,+1\}$ which indicates if a move is Opponent or Player.


The game bool:


The game bool $\Rightarrow$ bool :


## Strategies

## Definition

A play is a sequence of moves which respects the partial order and the incompatibility relation.

Definition
A strategy is a set of plays.

The strategy not:

$$
\text { bool } \quad \Rightarrow \quad \text { bool }
$$

The strategy not:

$$
\text { bool } \quad \Rightarrow \quad \text { bool }
$$

$q$

The strategy not:


F

The strategy not:

$$
\text { bool } \quad \Rightarrow \quad \text { bool }
$$

The characterizations of definable strategies capture the interactive behavior of programs.

Two series of works laid the foundations of game semantics:

- fully complete models of MLL [AJ,HO]
- fully abstract models of PCF [AJM,HON] imposing conditions on strategies:
- innocence, bracketing, ...
extended later on to model various programming features:
- references, control, non-determinism, ...

How can we extend these results to concurrent languages?

## Concurrent computations

Computations are more and more performed in parallel:

- networks,
- multi-core processors,
- etc.


## Concurrent computations

Computations are more and more performed in parallel:

- networks,
- multi-core processors,
- etc.
which raises new problems:
- synchronization
- shared resources


## Concurrent computations

Computations are more and more performed in parallel:

- networks,
- multi-core processors,
- etc.
which raises new problems:
- synchronization
- shared resources
which raises new questions:
- how can we describe these computations? ( $\pi$-calculus, bigraphs, interaction nets, ...)
- how can we type them?
- how can we model them?

We are going to define a game semantics which captures concurrency in programs and proofs.

## Revisiting game semantics

Our games are

- asynchronous:
sequential plays are replaced by Mazurkiewicz traces
- non-alternating.


## Linear Logic

We consider here MALL formulas (without units):

$$
\begin{array}{cc}
\frac{\vdash \Gamma, A, B}{\vdash \Gamma, A 8 B}(\ngtr) & \frac{\vdash \Gamma_{1}, A \quad \vdash \Gamma_{2}, B}{\vdash \Gamma_{1}, \Gamma_{2}, A \otimes B}(\otimes) \\
\frac{\vdash \Gamma, A \quad \vdash \Gamma, B}{\vdash \Gamma, A \& B}(\&) & \frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}(\oplus)
\end{array}
$$

## Mixing points of view

Asynchronous games provide a model in which we can recover

- trace semantics (games),
- causal semantics (event structures),
- relational semantics,
- concurrent semantics (closure operators),


## Mixing points of view

Asynchronous games provide a model in which we can recover

- trace semantics (games),
- causal semantics (event structures),
- relational semantics,
- concurrent semantics (closure operators),
in which we characterize
- a fully complete model of MLL (without units),
- HO innocent strategies in LL,


## Mixing points of view

Asynchronous games provide a model in which we can recover

- trace semantics (games),
- causal semantics (event structures),
- relational semantics,
- concurrent semantics (closure operators),
in which we characterize
- a fully complete model of MLL (without units),
- HO innocent strategies in LL,
by using ideas coming from
- game semantics,
- concurrency theory,
- linear logic,
- category theory.


## The asynchronous graph of a game



- Position: downward-closed set of compatible moves.
- Play: path from the initial position $\emptyset$.
- Strategy: prefix-closed set of plays.


## The asynchronous graph of a game

$$
\begin{array}{cc}
\circ & \emptyset \\
& \\
& \mapsto
\end{array}
$$

## The asynchronous graph of a game

$$
\mathrm{o} \otimes \mathrm{o}
$$



## The asynchronous graph of a game



## Implementations of conjunction

Left implementation of conjunction:

$$
\begin{array}{cccc}
\text { bool } \otimes \text { bool } & \multimap & \text { bool } \\
& & & q \\
q_{L} & & & \\
V_{L} & & \\
& q_{R} & & \\
& & F_{R} & \\
& & &
\end{array}
$$

## Implementations of conjunction

Right implementation of conjunction:

$$
\begin{array}{cccc}
\text { bool } \otimes \text { bool } & \multimap & \text { bool } \\
& & & q \\
& q_{R} & & \\
& F_{R} & & \\
& q_{L} & & \\
V_{L} & & & \\
& & &
\end{array}
$$

$$
F
$$

## Strategies of conjunction

The game bool $\otimes$ bool $\multimap$ bool contains eight subgraphs:


## Strategies of conjunction

Left implementation of conjunction:


## Strategies of conjunction

Right implementation of conjunction:


## Strategies of conjunction

## Parallel implementation of conjunction:



# We want to understand the behavior of strategies generated by proofs in linear logic. 

We are going to enforce a series of diagrammatic axioms on our strategies.

## An MLL proof

$$
\begin{align*}
& \frac{\vdots}{\frac{\vdots}{A^{*}, A}} \frac{\vdots}{A^{*} \otimes B^{*}, A, B}(\otimes) \\
& \begin{array}{c}
A^{*} \otimes B^{*}, A \ngtr B \\
A^{*} \otimes B^{*}, C^{*},(A \ngtr B) \otimes C \\
C
\end{array}  \tag{8}\\
& \frac{\left(A^{*} \otimes B^{*}\right) \gamma C^{*},(A \varnothing B) \otimes C}{\left(\left(A^{*} \otimes B^{*}\right) \gamma C^{*}\right) \gamma((A \varnothing B) \otimes C)} \tag{8}
\end{align*}
$$

- A formula induces a partial order on its connectives.
- A proof is a way to explore the formula
i.e. a partial order on the moves which refines the partial order of the game.


## The Cube property

- Every partial order induces an asynchronous graph.
- Conversely, every asynchronous graph such that

has its homotopy classes are characterized by a partial order on the moves appearing in the paths.


## Asynchronous games

Definition
A game is a pointed asynchronous graph satisfying the Cube property.

We are going to consider strategies $\sigma$ which

- are positional (or history-free):

i.e. characterized by the subgraph of the game they explore.
- satisfy the Cube property.

These properties are not preserved by composition of strategies!

## Ingenuous strategies

## Definition

A strategy $\sigma: A$ is deterministic when

where $m$ is a Player move.

## Definition

A strategy $\sigma$ is ingenuous when
(1) it satisfies the preceding conditions,
(2) it is deterministic.

## Property

Ingenuous strategies compose and form a *-autonomous category (which is compact closed).

This category still has "too many" strategies!

$$
A \otimes B=A \ngtr B
$$

## Halting positions

In the spirit of the relational model, a strategy $\sigma$ should be characterized by its set $\sigma^{\circ}$ of halting positions.

## Definition

A halting position of a strategy $\sigma$ is a position $x$ such that there is no Player move $m: x \longrightarrow y$ that $\sigma$ can play.

The game bool $\otimes$ bool contains the subgraph:


The pair true $\otimes$ false:


The left biased pair true $\otimes$ false:


## Courteous strategies

## Definition

An ingenuous strategy $\sigma$ is courteous when it satisfies

where $m$ is a Player move.

## Theorem

A courteous ingenuous strategy $\sigma$ is characterized by its set $\sigma^{\circ}$ of halting positions.

## Concurrent strategies

The halting positions of such a strategy $\sigma: A$ are precisely the fixpoints of a closure operator on the positions of $A$.

- We thus recover the model of concurrent strategies.
- A semantical counterpart of the focalization property: strategies can play all their Player moves in one "cluster" of moves.


The operation $(-)^{\circ}$ from the category of games to the category of relations is not functorial!

## Games



This mismatch is essentially due to deadlock situations occurring during the interaction.

## Scheduled strategies: avoiding deadlocks

Composing the right implementation of the conjunction with the left biased pair true $\theta$ false leads to a deadlock.

## Scheduled strategies: avoiding deadlocks

Composing the right implementation of the conjunction with the left biased pair true $\theta$ false leads to a deadlock.


## Scheduled strategies: avoiding deadlocks

Composing the right implementation of the conjunction with the left biased pair true $\otimes$ false leads to a deadlock.


- The culprit is the pair, since it induces dependencies between components of the tensor product bool $\otimes$ bool.
- This can be detected by a dynamic scheduling criterion, which enforces an oriented version of the correctness criterion.

$$
A \otimes B \quad=\quad(A \otimes B \quad \cup \quad A \otimes B)^{\perp}
$$

## Outcome

- We reconstruct the concurrent game model (closure operators).
- We thus obtain a model of MALL, fully complete for MLL (without units).
- Two diagrammatic axioms enable us to characterize HO innocent strategies in this model.

Asynchronous games constitute a rich and unifying framework in which we can study concurrent situations and compare various models.

## Part II

## Causality in String Diagrams

## The structure of causality

- We have studied the structure of interactive traces generated by proofs, but what is the structure of causality between moves?
- We have given an external characterization of strategies generated by proofs (by restricting the space of strategies), can we give an internal characterization by generating those strategies?

We are going to give a presentation of a category of games and strategies in a simple case.

## First-order propositional logic

- Formulas:

$$
A::=\forall x \cdot A|\exists x \cdot A| P(x) \mid \ldots
$$

- Rules:

$$
\frac{\Gamma \vdash P, \Delta}{\Gamma \vdash \forall x . P, \Delta}(\forall)
$$

$$
\frac{\Gamma \vdash P[t / x], \Delta}{\Gamma \vdash \exists x . P, \Delta}(\exists)
$$

(with $x \notin \mathrm{FV}(\Gamma, \Delta)$ )

## Causality in proofs

$$
\frac{\frac{\pi}{\Gamma \vdash A, B, \Delta}}{\Gamma \vdash A, \forall y \cdot B, \Delta}(\forall)(\forall)
$$



## Causality in proofs

$$
\frac{\frac{\pi}{\Gamma \vdash A, B, \Delta}}{\Gamma \vdash A, \forall y \cdot B, \Delta}(\forall)(\forall)
$$



## Causality in proofs



## Causality in proofs

$$
\frac{\frac{\pi}{\Gamma \vdash A[t / x], B, \Delta}}{\frac{\Gamma \vdash A[t / x], \forall y \cdot B, \Delta}{\Gamma \vdash \exists x \cdot A, \forall y \cdot B, \Delta}(\forall)}(\exists) \rightsquigarrow \frac{\frac{\pi}{\Gamma \vdash A[t / x], B, \Delta}}{\frac{\Gamma \vdash \exists x \cdot A, B, \Delta}{\Gamma \vdash \exists x \cdot A, \forall y \cdot B, \Delta}(\exists)}(\forall)
$$



## Causality in proofs

$$
\frac{\pi}{\frac{\pi \vdash A, B[t / y], \Delta}{\Gamma \vdash A, \exists y \cdot B, \Delta}}(\exists)(\forall) \rightsquigarrow \frac{\frac{\pi}{\Gamma \vdash A, B[t / y], \Delta}}{\Gamma \vdash \forall x \cdot A, \exists y \cdot B, \Delta}(\forall)
$$

## Causality in proofs



Only when $x \notin \mathrm{FV}(t)$ !

## Causality in proofs

## Essential causal dependencies induced by proofs are

$$
\forall x \longrightarrow \exists y
$$

where the witness $t$ given for $y$ admits $x$ as free variable.

## Games

A formula

$$
\forall x . \forall y . \exists z . \forall t . P
$$

will be interpreted by the game (= polarized poset)


## Games

A sequent

$$
\exists x . P(x) \vdash \exists y \cdot \exists z . P(y) \wedge P(z)
$$

will be interpreted by the game (= polarized poset)


A proof

$$
\frac{\frac{\overline{P(x) \vdash P(x) \wedge P(x)}}{\frac{P(x) \vdash \exists z \cdot P(x) \wedge P(z)}{P(x) \vdash \exists y \cdot \exists z \cdot P(y) \wedge P(z)}}(\exists)}{\exists x \cdot P(x) \vdash \exists y \cdot \exists z \cdot P(y) \wedge P(z)}(\exists)
$$

will be interpreted by the strategy


## Causal strategies

## game $A=$ partial order on the moves strategy $\sigma=$ relation on moves

## Causal strategies

## game $A=$ partial order on the moves strategy $\sigma=$ relation on moves

A strategy $\sigma$ : A is causal when
(1) if $m \xrightarrow{\sigma} n$ then $m$ Opponent and $n$ Player
(2) the relation $\leq_{A} \cup \sigma$ is acyclic

## Causal strategies

## game $A=$ partial order on the moves strategy $\sigma=$ relation on moves

A strategy $\sigma$ : A is causal when
(1) if $m \xrightarrow{\sigma} n$ then $m$ Opponent and $n$ Player
(2) the relation $\leq_{A} \cup \sigma$ is acyclic


Admits:


## Causal strategies

## game $A=$ partial order on the moves strategy $\sigma=$ relation on moves

A strategy $\sigma$ : A is causal when
(1) if $m \xrightarrow{\sigma} n$ then $m$ Opponent and $n$ Player
(2) the relation $\leq_{A} \cup \sigma$ is acyclic


## Presentations of monoids

## Definition

A presentation of a monoid $M$ is given by

- a set $G$ of generators,
- a set $R \subseteq G^{*} \times G^{*}$ of relations,
such that

$$
M \cong\langle G \mid R\rangle \cong G^{*} / \approx
$$

Example

- $\mathbb{N} \cong\langle a \mid\rangle$
- $\mathbb{N} / 2 \mathbb{N} \cong\langle a \mid a a=1\rangle$
- $\mathbb{N} \times \mathbb{N} \cong\langle a, b \mid a b=b a\rangle$
- etc.


## Presentations of categories

More generally, a polygraph presents an $n$-category by giving

- typed generators of dimension $i$ (for $0 \leq i \leq n$ ),
- typed relations of dimension $n+1$.


## Illustration: the simplicial category

The simplicial category $\Delta$ has

- as objects: sets $[n]=\{0,1, \ldots, n-1\}$ where $n \in \mathbb{N}$,
- as morphisms: weakly increasing functions.


## Illustration: the simplicial category

The simplicial category $\Delta$ has

- as objects: sets $[n]=\{0,1, \ldots, n-1\}$ where $n \in \mathbb{N}$,
- as morphisms: weakly increasing functions.
$0-0$



## Illustration: the simplicial category

The simplicial category $\Delta$ has

- as objects: sets $[n]=\{0,1, \ldots, n-1\}$ where $n \in \mathbb{N}$,
- as morphisms: weakly increasing functions.

It is a category: horizontal composition (०)


## Illustration: the simplicial category

The simplicial category $\Delta$ has

- as objects: sets $[n]=\{0,1, \ldots, n-1\}$ where $n \in \mathbb{N}$,
- as morphisms: weakly increasing functions.

It is a category: horizontal composition (०)


## Illustration: the simplicial category

The simplicial category $\Delta$ has

- as objects: sets $[n]=\{0,1, \ldots, n-1\}$ where $n \in \mathbb{N}$,
- as morphisms: weakly increasing functions.

This category is monoidal: vertical composition $(\otimes)$


## Illustration: the simplicial category

The simplicial category $\Delta$ has

- as objects: sets $[n]=\{0,1, \ldots, n-1\}$ where $n \in \mathbb{N}$,
- as morphisms: weakly increasing functions.

This category is monoidal: vertical composition $(\otimes)$


## Illustration: the simplicial category

The simplicial category $\Delta$ has

- as objects: sets $[n]=\{0,1, \ldots, n-1\}$ where $n \in \mathbb{N}$,
- as morphisms: weakly increasing functions.

This category is monoidal: vertical composition $(\otimes)$


## A theory of monoids

The category $\Delta$ contains two morphisms:

$$
\mu:[2] \rightarrow[1] \quad \text { and } \quad \eta:[0] \rightarrow[1]
$$



## A theory of monoids

The category $\Delta$ contains two morphisms:

$$
\mu:[2] \rightarrow[1] \quad \text { and } \quad \eta:[0] \rightarrow[1]
$$



which satisfy

and


# A theory of monoids 

## Property

The morphisms $\mu$ and $\eta$ generate $\Delta$.


## A theory of monoids

## Property

The morphisms $\mu$ and $\eta$ generate $\Delta$.


## A presentation of the category $\Delta$

The category $\Delta$ is isomorphic to the free monoidal category on the two generators

and $\quad \eta:[0] \rightarrow[1]$
$0-0$
quotiented by the relations

and


## A theory of monoids


$\operatorname{Mon}(\mathcal{C}) \cong \operatorname{StrMonCat}(\Delta, \mathcal{C})$

## A theory of games

| strict monoidal functor Games $\rightarrow \mathcal{C}$ |
| :---: |
| $=$ |
| ????? in $\mathcal{C}$ |

The corresponding theory is a polarized variant of relations.

## The category Games

The category Games is the category whose

- objects are integers

$$
[n]=\{0,1,2, \ldots, n-1\}
$$

equipped with a polarization function

$$
\lambda:[n] \rightarrow\{\forall, \exists\}
$$



## The category Games

The category Games is the category whose

- objects are integers

$$
[n]=\{0,1,2, \ldots, n-1\}
$$

equipped with a polarization function

$$
\lambda:[n] \rightarrow\{\forall, \exists\}
$$

## The category Games

The category Games is the category whose

- objects are integers

$$
[n]=\{0,1,2, \ldots, n-1\}
$$

equipped with a polarization function

$$
\lambda:[n] \rightarrow\{\forall, \exists\}
$$

- morphisms are causal strategies.



## The structure of strings



## The structure of strings



## Presentation of the category Games

The category Games is presented by the polygraph with

- one 0-cell,


## Presentation of the category Games

The category Games is presented by the polygraph with

- one 0-cell,
- two 1-cells $\forall$ and $\exists$,


## Presentation of the category Games

The category Games is presented by the polygraph with

- one 0-cell,
- two 1-cells $\forall$ and $\exists$,
- six 2-cells



## Presentation of the category Games

The category Games is presented by the polygraph with

- one 0-cell,
- two 1-cells $\forall$ and $\exists$,
- six 2-cells

- 3-cells (relations) ensuring that
- $\forall$ is a bicommutative bialgebra,
- $\exists$ is left dual to $\forall$.


## Technical byproducts

From this presentation we can deduce that

- causal strategies compose,
- causal strategies are definable: we only have to show that generators are.


## 2-dimensional rewriting

- The proof is done by showing that every diagram is in relation with a diagram in canonical form and that these canonical forms are in bijection with the morphisms of this category.


## 2-dimensional rewriting

- The proof is done by showing that every diagram is in relation with a diagram in canonical form and that these canonical forms are in bijection with the morphisms of this category.
- This proof is very repetitive and requires to handle numerous cases: it should be automated.


## 2-dimensional rewriting

- The proof is done by showing that every diagram is in relation with a diagram in canonical form and that these canonical forms are in bijection with the morphisms of this category.
- This proof is very repetitive and requires to handle numerous cases: it should be automated.
- We have oriented the presentation of $\operatorname{Mat}(\mathbb{N})$ into a confluent rewriting system (§5.5)



## 2-dimensional rewriting

- The proof is done by showing that every diagram is in relation with a diagram in canonical form and that these canonical forms are in bijection with the morphisms of this category.
- This proof is very repetitive and requires to handle numerous cases: it should be automated.
- We have oriented the presentation of $\operatorname{Mat}(\mathbb{N})$ into a confluent rewriting system (§5.5)

- We have introduced an unification algorithm in order to compute critical pairs of such rewriting systems (§5.4).


## Contributions

We defined an asynchronous non-alternating game semantics

- which takes the concurrency of proofs in account,
- which unifies preexisting semantics of linear logic (§2.3),
- in which we extend the notion of HO innocence (§2.4),
- in which we give an interactive reformulation of the correctness criterion (§2.5).

We gave a presentation of a category of games and strategies

- which reveals the algebraic structure of first-order causal dependencies (§4.2),
- which lays the foundations for a 2-dimensional extension of rewriting theory (§5).

Thanks!

