Sémantique des jeux asynchrones et réécriture 2-dimensionnelle

Soutenance de thèse de doctorat

Samuel Mimram

Laboratoire PPS (CNRS – Université Paris Diderot)

1<sup>er</sup> décembre 2008

A program is

a text

in a programming language.

```
/* kernel entry point function, called from assembler code */
void kernel entry(unsigned long magic.unsigned long addr)
  /* we will need a multiboot info pointer */
  multiboot info t *mbi;
  memory map t *mmap;
  unsigned long mmap size:
  unsigned long 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 low;
        break:
      }
      mmap size-=sizeof(*mmap);
      mmap++;
    3
  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 = (unsigned long)(& begin+block size);
  heap = \& end;
  heaplimit = & begin+block size;
  last seen = heap + 2 + 4*si \text{ 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(argv):
```

#### A program is

a text

in a programming language.

We want to give a meaning to this language!

• Typing programs can avoid some errors

1 : int

• Typing programs can avoid some errors

fun  $x \rightarrow x+1$  : int  $\Rightarrow$  int

• Typing programs can avoid some errors

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

• Typing programs can avoid some errors

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

- The Curry-Howard correspondence:
  - type of a program = formula program = proof of its type
- Studying programs is the same as studying proofs.

## Denotational semantics

#### A model interprets

- a type *A* as a *computation space* [[*A*]]
- a program  $f : A \Rightarrow B$  as a transformation  $\llbracket f \rrbracket : \llbracket A \rrbracket \to \llbracket B \rrbracket$

#### $A \Rightarrow B \qquad : \qquad f \qquad \sim \sim \gg \qquad \llbracket f \rrbracket \qquad : \qquad \llbracket A \rrbracket \rightarrow \llbracket B \rrbracket$

## Denotational semantics

A denotational model interprets

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



denotational semantics = 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*.

$(fun x \rightarrow not x)$ false	$\rightsquigarrow$	true
(fun $x \rightarrow \text{not } x$ ) <b>true</b>	$\rightsquigarrow$	false

#### ⇒ 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, ≤, #)
- equipped with a *polarisation* function λ : M → {-1, +1} which indicates if a move is Opponent or Player.







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.

#### bool $\Rightarrow$ 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? (π-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):

$$\frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \ \mathfrak{P} B}(\mathfrak{P}) \qquad \frac{\vdash \Gamma_1, A \vdash \Gamma_2, B}{\vdash \Gamma_1, \Gamma_2, A \otimes B}(\otimes)$$
$$\frac{\vdash \Gamma, A \vdash \Gamma, B}{\vdash \Gamma, A \& B}(\&) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}(\oplus)$$

# 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 Ø.
- Strategy: prefix-closed set of plays.
### The asynchronous graph of a game



### The asynchronous graph of a game



### The asynchronous graph of a game



### Implementations of conjunction

Left implementation of conjunction:



### Implementations of conjunction

Right implementation of conjunction:

bool  $\otimes$  bool bool \_\_\_\_ q  $q_R$  $F_R$  $q_L$  $V_L$ F

The game bool  $\otimes$  bool  $-\infty$  bool contains eight subgraphs:



Left implementation of conjunction:



Right implementation 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



- 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 $\operatorname{ingenuous}$ 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!

#### $A \otimes B = A \ \mathcal{B} B$

### 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!



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  $\otimes$  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.





### 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 ⊗ bool.
- This can be detected by a dynamic **scheduling criterion**, which enforces an oriented version of the correctness criterion.

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

### 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:

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

Rules:

 $\frac{\Gamma \vdash P, \Delta}{\Gamma \vdash \forall x. P, \Delta} (\forall) \qquad \qquad \frac{\Gamma \vdash P[t/x], \Delta}{\Gamma \vdash \exists x. P, \Delta} (\exists)$ (with  $x \notin FV(\Gamma, \Delta)$ )

### Causality in proofs





### Causality in proofs



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



implies

 $\rightarrow$ 



### Causality in proofs






0

0









Only when  $x \notin FV(t)!$ 

Essential causal dependencies induced by proofs are



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

#### $\exists x.P(x) \vdash \exists y.\exists z.P(y) \land P(z)$

will be interpreted by the game (= polarized poset)



A proof

$$\frac{\frac{1}{P(x) \vdash P(x) \land P(x)}}{\frac{P(x) \vdash \exists z. P(x) \land P(z)}{P(x) \vdash \exists y. \exists z. P(y) \land P(z)}} (\exists) \\ \exists x. P(x) \vdash \exists y. \exists z. P(y) \land P(z)} (\exists)$$

will be interpreted by the strategy



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

**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

- **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



**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  $R \subseteq G^* imes G^*$  of *relations*,

such that

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

#### Example

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

## Presentations of categories

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

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

The simplicial category  $\Delta$  has

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

The simplicial category  $\Delta$  has

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



The simplicial category  $\Delta$  has

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

It is a category: horizontal composition ( $\circ$ )



The simplicial category  $\Delta$  has

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

It is a category: horizontal composition ( $\circ$ )



The simplicial category  $\Delta$  has

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

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



The simplicial category  $\Delta$  has

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

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



The simplicial category  $\Delta$  has

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

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



The category  $\Delta$  contains two morphisms:

 $\mu: [2] \to [1] \qquad \text{and} \qquad \eta: [0] \to [1]$   $0 \longrightarrow 0 \qquad \qquad 0$ 

The category  $\Delta$  contains two morphisms:



#### Property

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



#### Property

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



# A presentation of the category $\Delta$

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



quotiented by the relations







strict monoidal functor  $\Delta \to \mathcal{C}$ 

monoid in  $\ensuremath{\mathcal{C}}$ 

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

## A theory of games

strict monoidal functor **Games**  $\rightarrow C$ = ????? in C

The corresponding theory is a polarized variant of relations.

# The category Games

0 V

The category  $\ensuremath{\textbf{Games}}$  is the category whose

• objects are integers

$$[n] = \{0, 1, 2, \dots, n-1\}$$
 equipped with a polarization function 
$$\lambda : [n] \to \{ \forall, \exists \}$$

# The category Games

The category  $\ensuremath{\textbf{Games}}$  is the category whose

• objects are integers

$$[n] = \{0, 1, 2, \dots, n-1\}$$
 equipped with a polarization function 
$$\lambda : [n] \to \{ \forall, \exists \}$$



## The category Games

The category Games is the category whose

• objects are integers

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

equipped with a polarization function

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

• morphisms are causal strategies.



## The structure of strings



## The structure of strings



The category  $\ensuremath{\textbf{Games}}$  is presented by the polygraph with

• one 0-cell,

The category  $\ensuremath{\textbf{Games}}$  is presented by the polygraph with

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

The category Games is presented by the polygraph with

- one 0-cell,
- two 1-cells ∀ and ∃,
- six 2-cells



The category Games is presented by the polygraph with

- one 0-cell,
- two 1-cells ∀ and ∃,
- six 2-cells



- 3-cells (relations) ensuring that
  - ∀ is a bicommutative bialgebra,
  - ∃ is left dual to ∀.
## Technical byproducts

From this presentation we can deduce that

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

• 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.

- 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**.

- 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 Mat(ℕ) into a confluent rewriting system (§5.5)



- 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 Mat(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 ( $\S5$ ).

# Thanks!