

Asynchronous Games

The concurrent geometry of proofs

Paul-André Melliès

CNRS, Université Paris 7

Rencontres Geocal 2006

Marseille, Vendredi 24 Février

Part I

Geometry of concurrency

A short tutorial

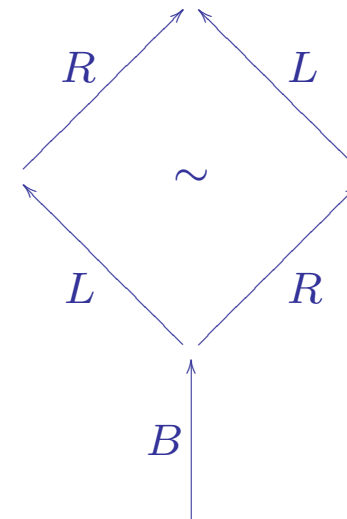
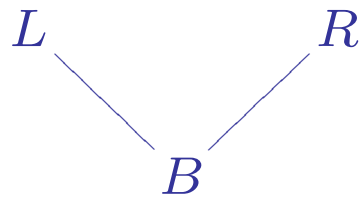
(from causality to geometry, and conversely)

Geometry of concurrency

Event structure



Transition space



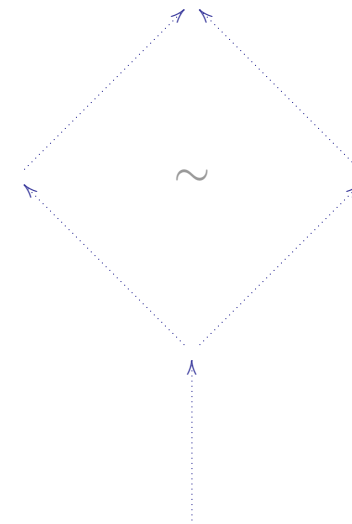
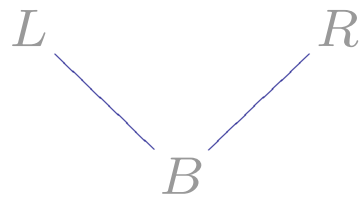
- | | | |
|-----|---|----------------------|
| B | = | Buy a pair of shoes, |
| L | = | Put left shoe on, |
| R | = | Put right shoe on. |

Geometry of concurrency

Event structure



Transition space



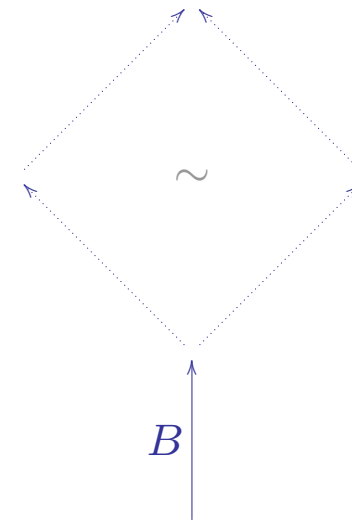
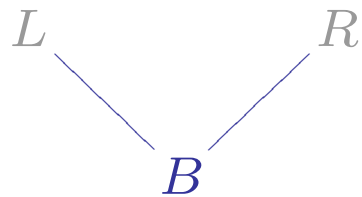
- | | | |
|-----|---|----------------------|
| B | = | Buy a pair of shoes, |
| L | = | Put left shoe on, |
| R | = | Put right shoe on. |

Geometry of concurrency

Event structure



Transition space



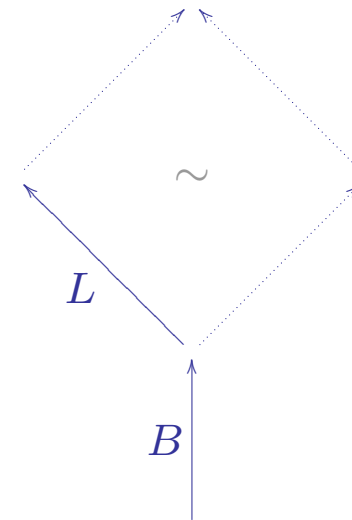
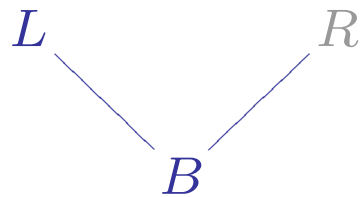
B	=	Buy a pair of shoes,
L	=	Put left shoe on,
R	=	Put right shoe on.

Geometry of concurrency

Event structure



Transition space



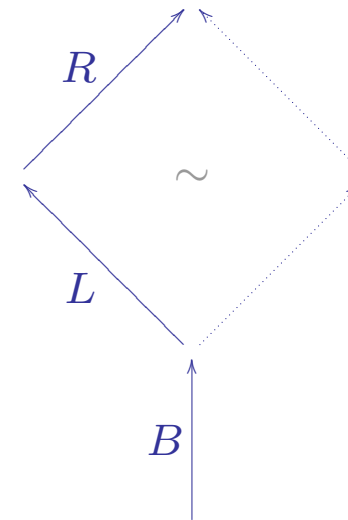
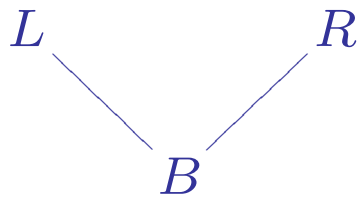
- | | | |
|-----|---|----------------------|
| B | = | Buy a pair of shoes, |
| L | = | Put left shoe on, |
| R | = | Put right shoe on. |

Geometry of concurrency

Event structure



Transition space



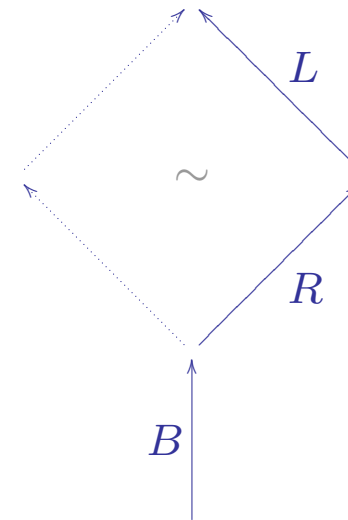
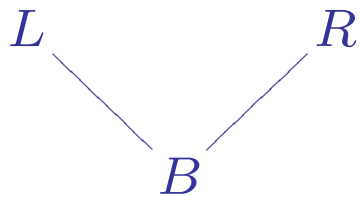
B	=	Buy a pair of shoes,
L	=	Put left shoe on,
R	=	Put right shoe on.

Geometry of concurrency

Event structure

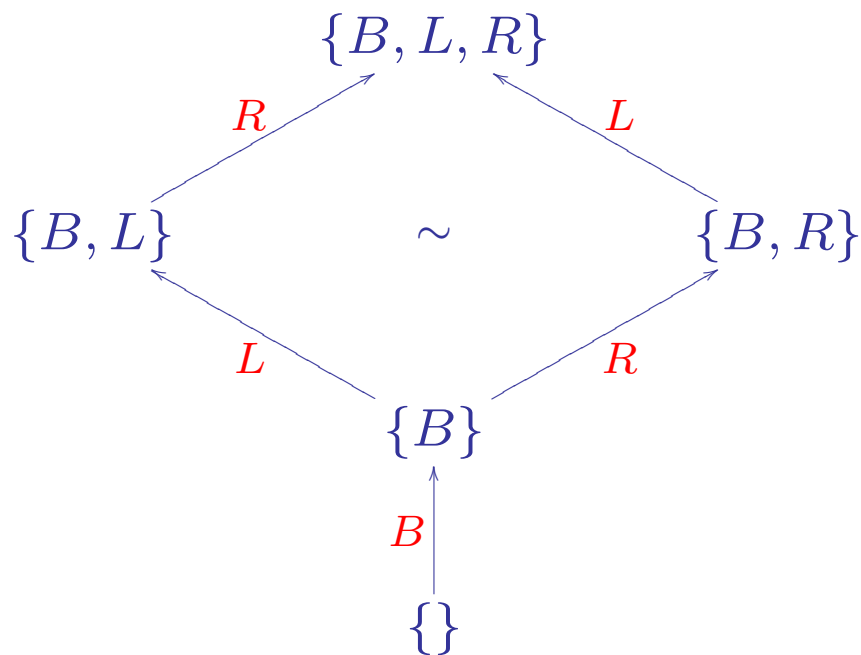


Transition space



B	=	Buy a pair of shoes,
L	=	Put left shoe on,
R	=	Put right shoe on.

A formal definition of the transition space



Node
=
Position
=
Downward-closed
set of events

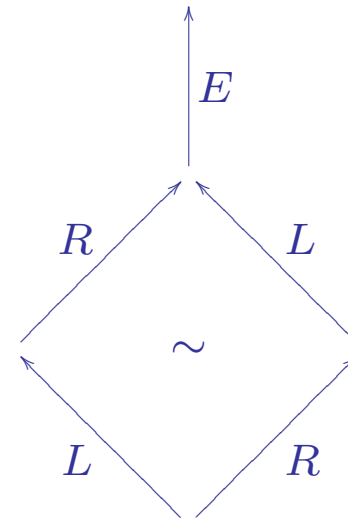
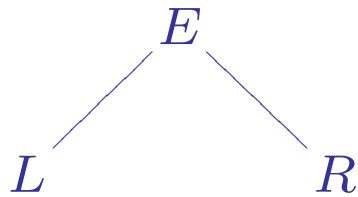
Edge
=
Event

Synchronization

Event structure



Transition space



L	=	Lace left shoe.
R	=	Lace right shoe.
E	=	Exit the store.

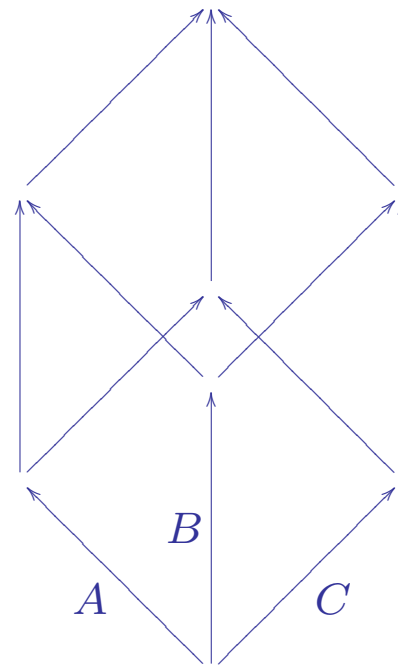
The 3-dimensional cube

Event structure



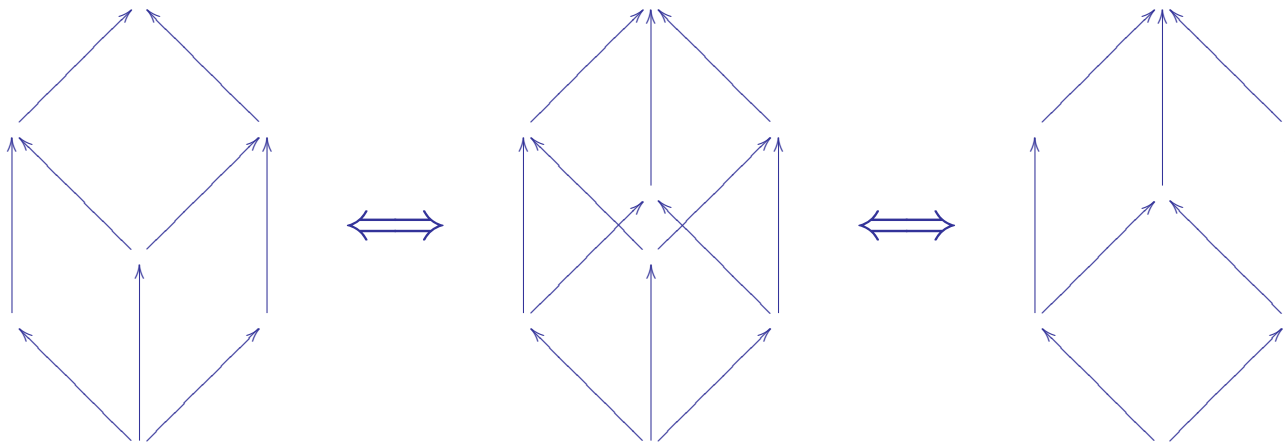
Transition space

A *B* *C*



The 3-dimensional cube property

Property: every transition space generated by an order satisfies:



Conversely...

Parallel transport

Definition: A path

$$x_1 \xrightarrow{t_1} x_2 \xrightarrow{t_2} x_3$$

transports a transition u to a transition v when:

$$\begin{array}{ccccc} y_1 & \longrightarrow & y_2 & \longrightarrow & y_3 \\ \uparrow u & & \sim & & \sim & & \uparrow v \\ x_1 & \xrightarrow{t_1} & x_2 & \xrightarrow{t_2} & x_3 \end{array}$$

Notationally:

$$\begin{array}{ccccc} y_1 & \longrightarrow & y_2 & \longrightarrow & y_3 \\ \uparrow u & & \parallel & & \uparrow v \\ x_1 & \xrightarrow{t_1} & x_2 & \xrightarrow{t_2} & x_3 \end{array}$$

Transition with history

Definition: a transition with history

$$x \xrightarrow{f} y \xrightarrow{u} z$$

consists of a path f extended by a transition u .

Alternatively seen as a non empty path: cf. A^\bullet in Martin Hyland's talk.

Confluence of extraction

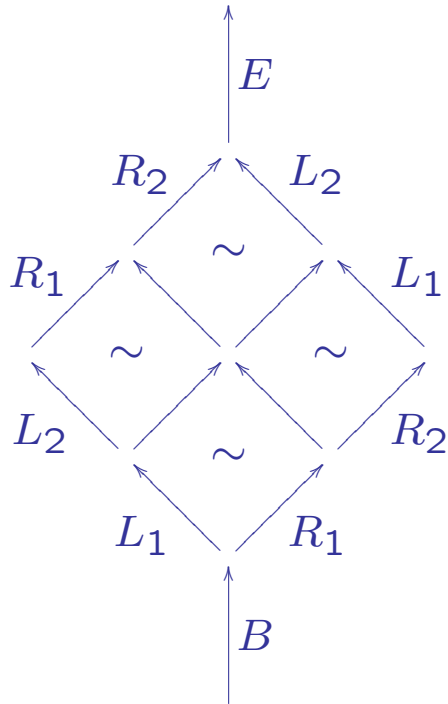
Property: In a transition space satisfying the **3-dimensional cube** property, the extraction procedure

$$x \xrightarrow{f} \cdot \xrightarrow{u} y \quad \rightsquigarrow \quad \dots \quad \rightsquigarrow \quad \dots$$

is **confluent**, and thus normalizes to a **unique** transition with history

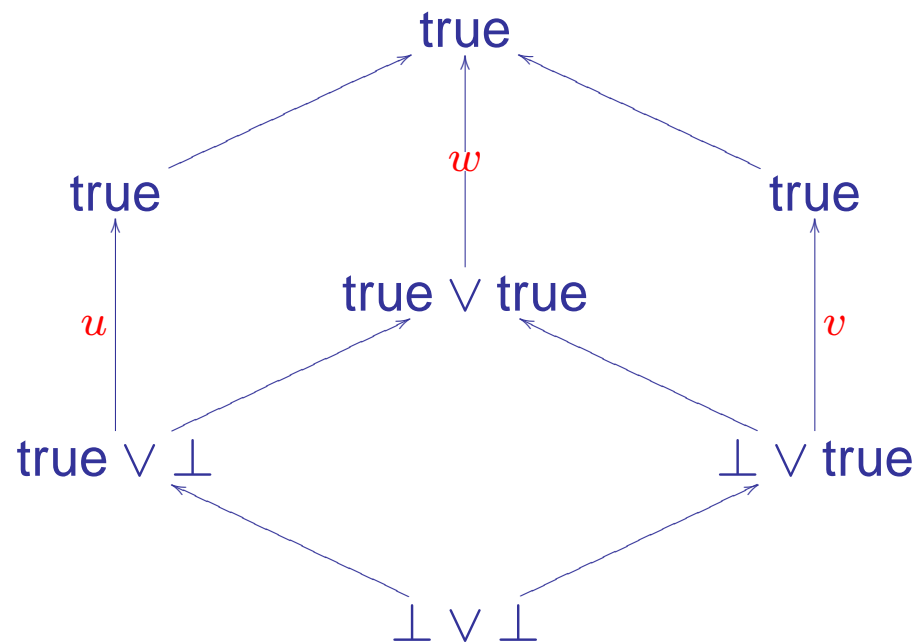
$$x \xrightarrow{g} \cdot \xrightarrow{v} z$$

Extraction: an illustration



$$\begin{aligned}
 B \cdot L_1 \cdot L_2 \cdot R_1 \cdot R_2 &\rightsquigarrow B \cdot L_1 \cdot R_1 \cdot R_2 \\
 &\rightsquigarrow B \cdot R_1 \cdot R_2.
 \end{aligned}$$

A counter-example to confluent extraction: parallel or



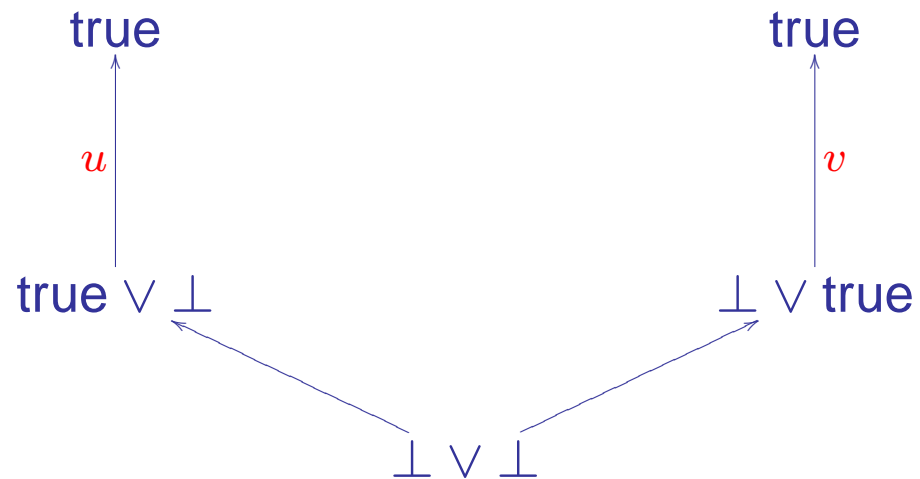
The rewriting rules

$\text{true} \vee x \longrightarrow \text{true}$

$x \vee \text{true} \longrightarrow \text{true}$

$\perp \longrightarrow \text{true}$

A counter-example to confluent extraction: parallel or



The rewriting rules		
$\text{true} \vee x$	\longrightarrow	true
$x \vee \text{true}$	\longrightarrow	true
\perp	\longrightarrow	true

Genericity

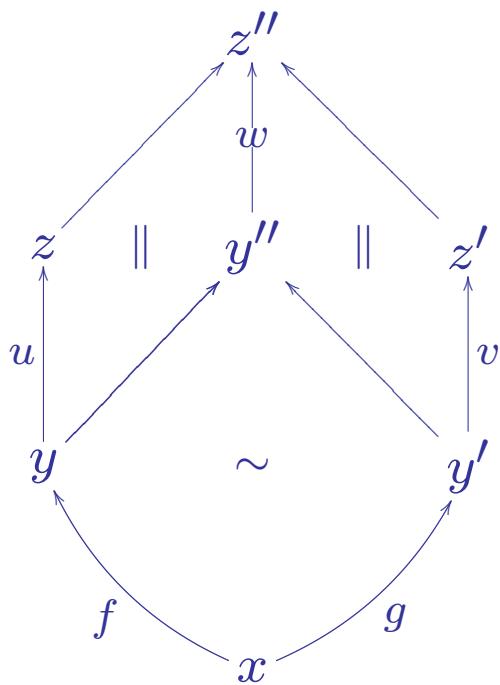
Property: A transition with history

$$x \xrightarrow{f} \cdot \xrightarrow{u} y$$

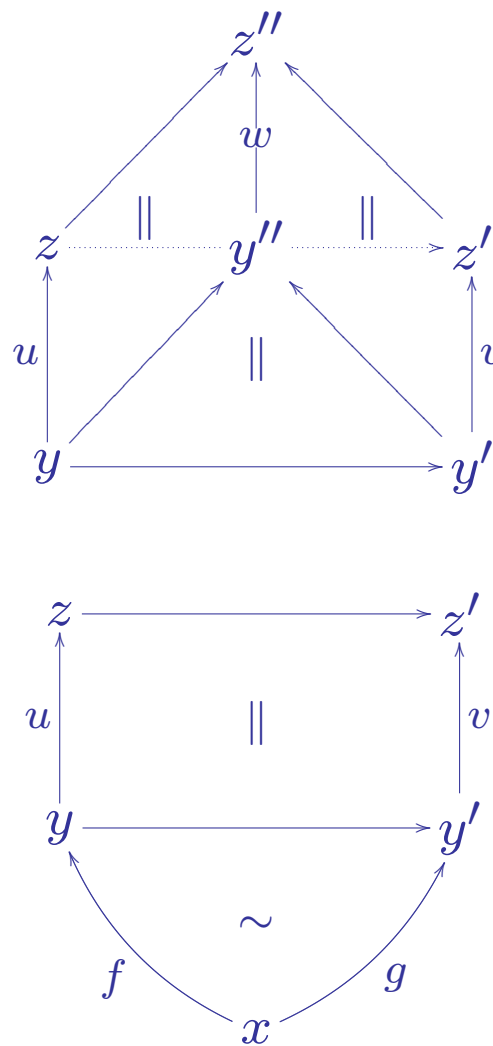
is a **normal form** wrt. \rightsquigarrow if and only if it is **generic**:

Normal form = No extraction step \rightsquigarrow from it.

Every diagram



factors uniquely as



Events

Definition: An **event** is the homotopy class of a normal form

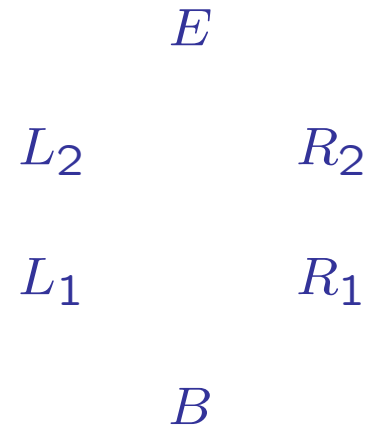
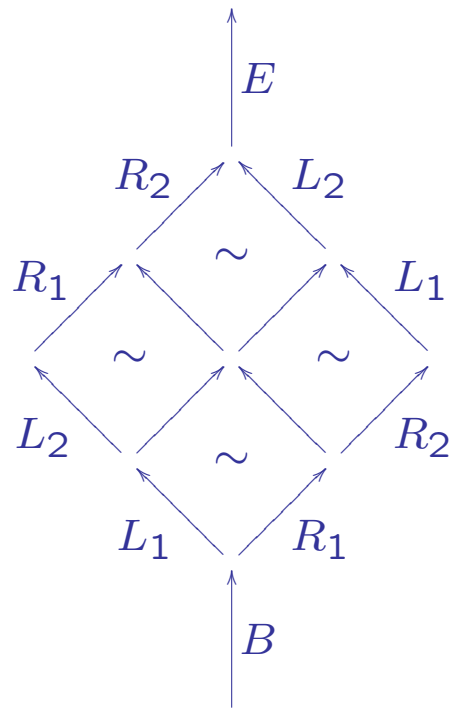
$$x \xrightarrow{f} y \xrightarrow{u} z.$$

Events: an illustration

Transition space

\mapsto

Events



Prefix modulo homotopy

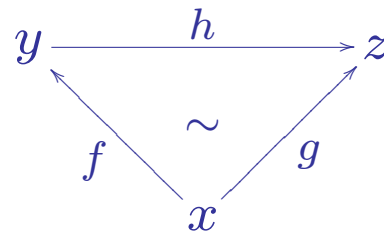
Definition: two coinital paths $f : x \longrightarrow y$ and $g : x \longrightarrow z$ satisfy

$$f \sqsubseteq g$$

when there exists a path

$$h : y \longrightarrow z$$

such that



Prefixing: a partial order on events

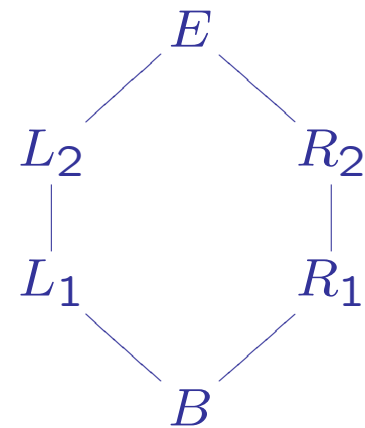
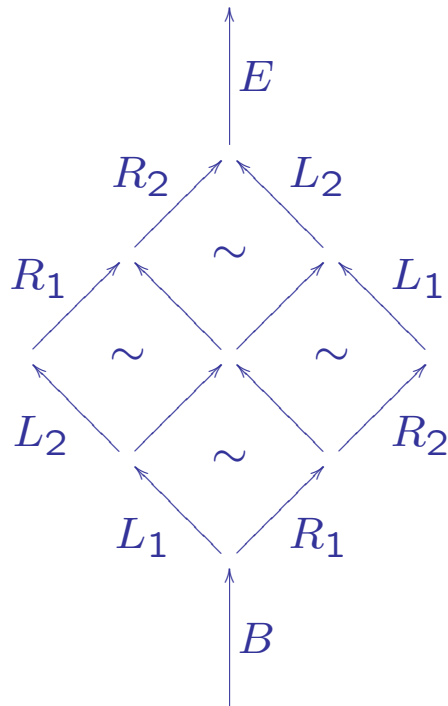
The prefix order \sqsubseteq defines a partial order on events.

The partial order of events: an illustration

Transition path

\mapsto

Partial order on events



The partial order $\llbracket f \rrbracket$ associated to a path f

Every path $f : x \longrightarrow y$ defines the **downward closed set**

$$\llbracket f \rrbracket$$

of all the **events** e satisfying

$$e \sqsubseteq f.$$

Causality deduced from geometry

Main property:

$$f \sim g \iff \llbracket f \rrbracket = \llbracket g \rrbracket.$$

A path $g \sim f$ is precisely a **sequentialization** of the partial order $\llbracket f \rrbracket$.

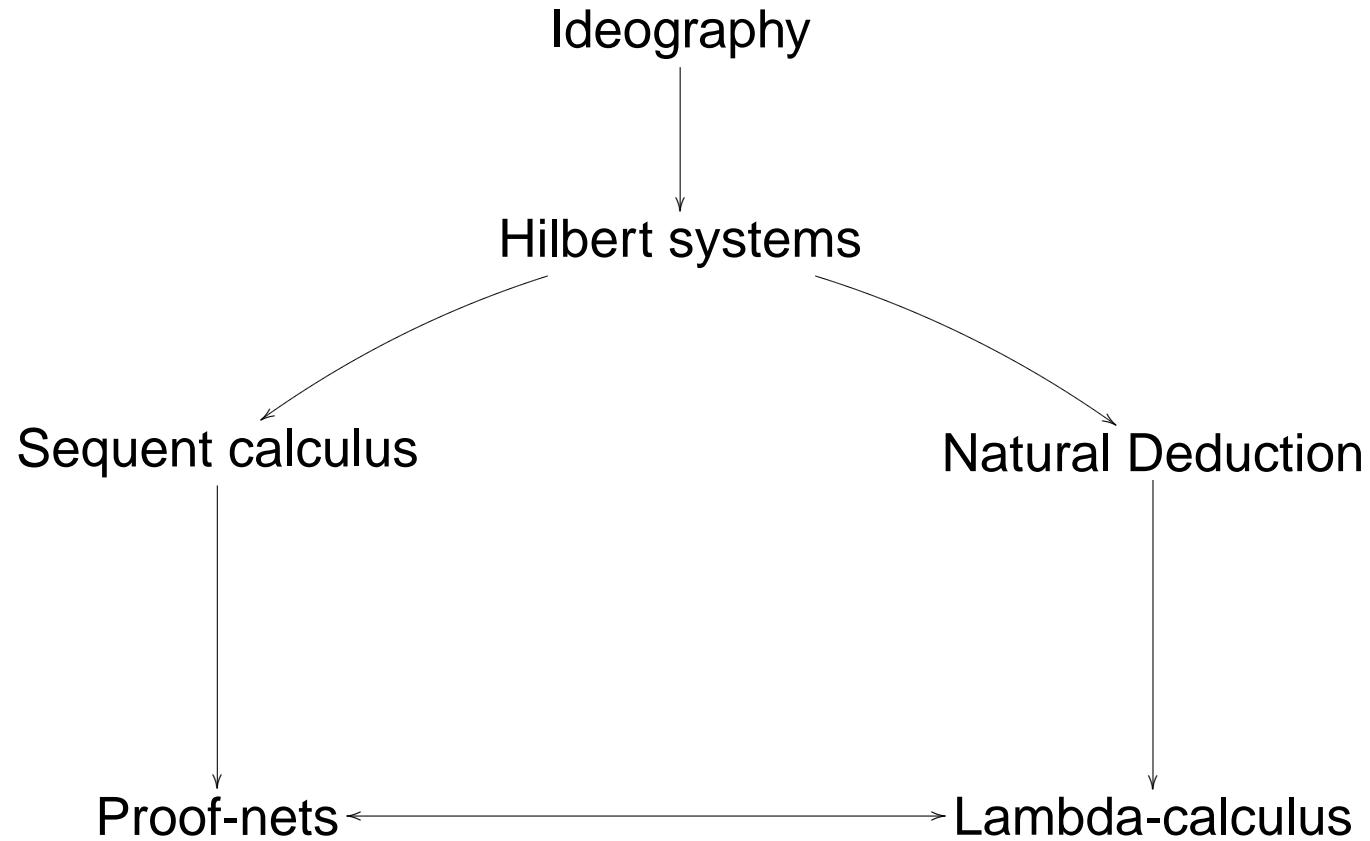
More on this in Samuel Mimram's talk.

Part II

Game semantics

In search of the logical phenomenon

Proof-theory: in search of a notation...



A proof of the drinker's formula in classical logic

$$\begin{array}{c}
 \frac{}{A(x_0) \vdash A(x_0)} \text{Axiom} \\
 \frac{}{A(x_0) \vdash \forall x.A(x), A(x_0)} \text{Right Weakening} \\
 \frac{}{\vdash A(x_0) \Rightarrow \forall x.A(x), A(x_0)} \text{Right } \Rightarrow \\
 \frac{}{B \vdash B} \text{Axiom} \\
 \frac{}{(A(x_0) \Rightarrow \forall x.A(x)) \Rightarrow B \vdash A(x_0), B} \text{Left } \Rightarrow \\
 \frac{}{\forall y.\{(A(y) \Rightarrow \forall x.A(x)) \Rightarrow B\} \vdash A(x_0), B} \text{Left } \forall \\
 \frac{}{\forall y.\{(A(y) \Rightarrow \forall x.A(x)) \Rightarrow B\} \vdash \forall x.A(x), B} \text{Right } \forall \\
 \frac{}{\forall y.\{(A(y) \Rightarrow \forall x.A(x)) \Rightarrow B\}, A(y)_0 \vdash \forall x.A(x), B} \text{Left Weakening} \\
 \frac{}{\forall y.\{(A(y) \Rightarrow \forall x.A(x)) \Rightarrow B\} \vdash A(y)_0 \Rightarrow \forall x.A(x), B} \text{Right } \Rightarrow \\
 \frac{}{B \vdash B} \text{Axiom} \\
 \frac{}{\forall y.\{(A(y) \Rightarrow \forall x.A(x)) \Rightarrow B\}, (A(y)_0 \Rightarrow \forall x.A(x)) \Rightarrow B \vdash B, B} \text{Left } \Rightarrow \\
 \frac{}{\forall y.\{(A(y) \Rightarrow \forall x.A(x)) \Rightarrow B\}, \forall y.\{(A(y) \Rightarrow \forall x.A(x)) \Rightarrow B\} \vdash B, B} \text{Left } \forall \\
 \frac{}{\forall y.\{(A(y) \Rightarrow \forall x.A(x)) \Rightarrow B\} \vdash B, B} \text{Contraction} \\
 \frac{}{\forall y.\{(A(y) \Rightarrow \forall x.A(x)) \Rightarrow B\} \vdash B} \text{Contraction} \\
 \frac{}{\vdash \forall y.\{(A(y) \Rightarrow \forall x.A(x)) \Rightarrow B\} \Rightarrow B} \text{Right } \Rightarrow
 \end{array}$$

Think of music!

106

17

dimin. *p*

dimin. *p*

dimin. *p*

dimin. *p*

dimin. *p*

dimin. *p*

dimin. *p*

dimin. *p*

Music is a **notation** constructed around an acoustic **phenomenon**.

Similarly...

Logic should be a **notation** constructed around a logical **phenomenon**.

But is there something like a logical phenomenon?

The **machine/notation** should not be confused with the **phenomenon** it generates: an oscillator is not a magnetic wave!

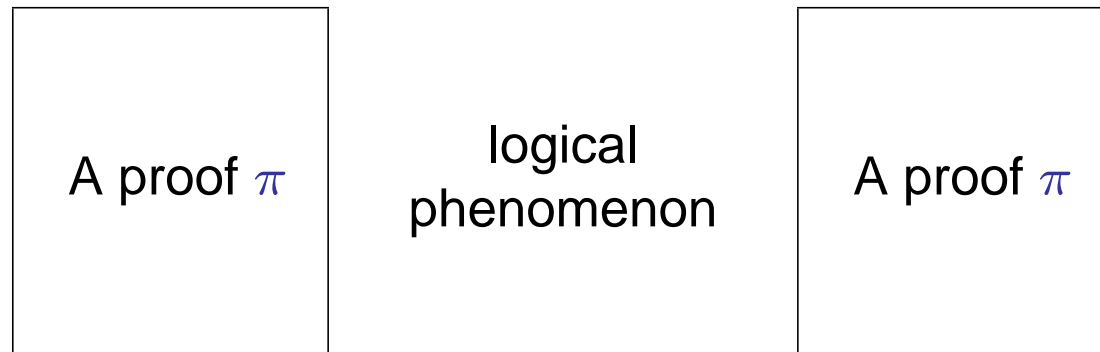
An heuristic: put the machine in a box, and study what emanates from it:



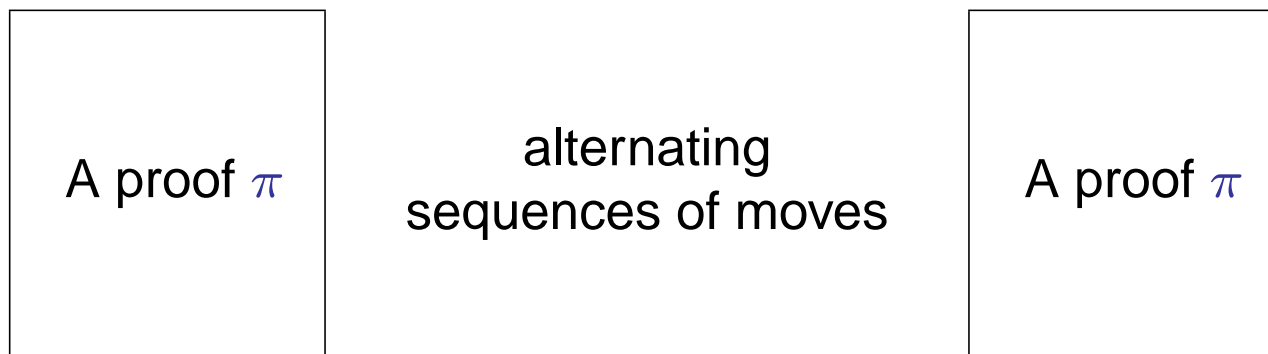
A proof π

Even better!

Capture **two** machines, and study how they **interact**:



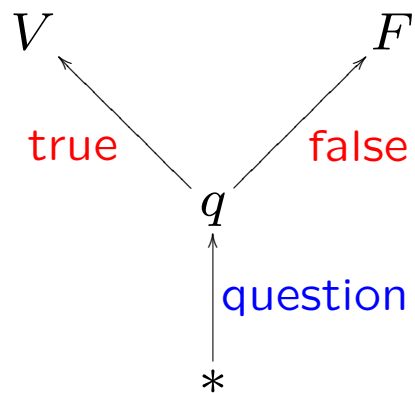
Traditional game semantics



Game semantics: an **interleaving** semantics of proofs.

An interleaving semantics

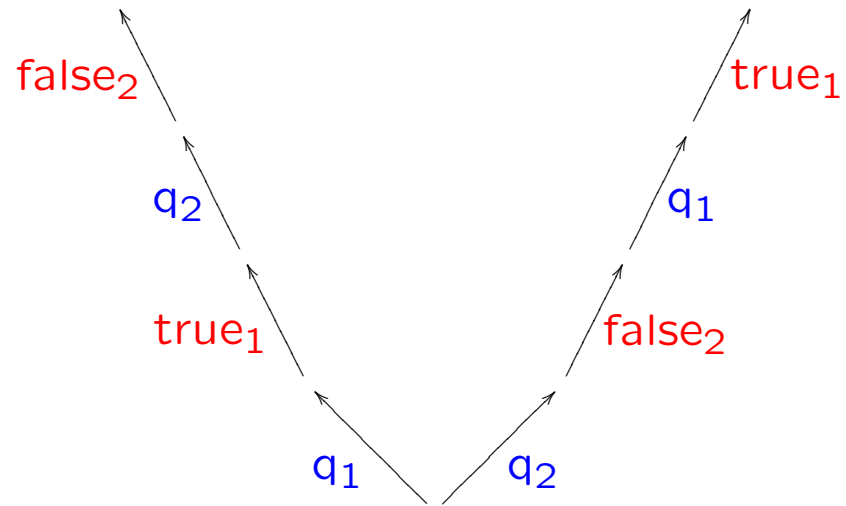
The boolean game \mathbb{B} :



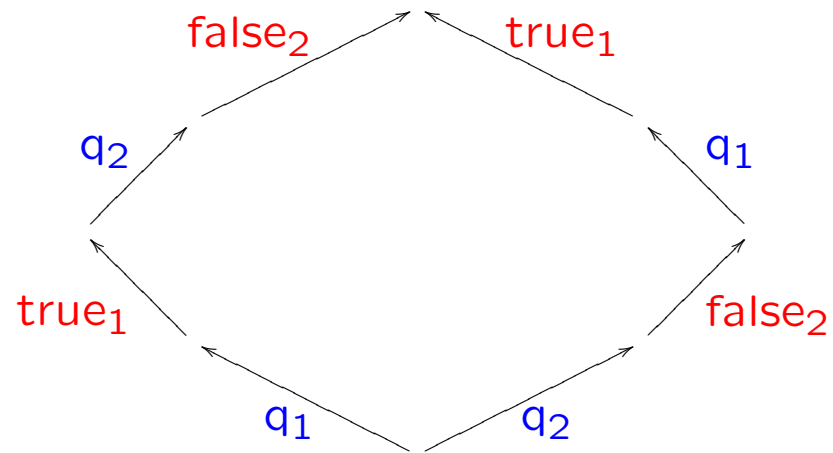
Player in red
Opponent in blue

An interleaving semantics

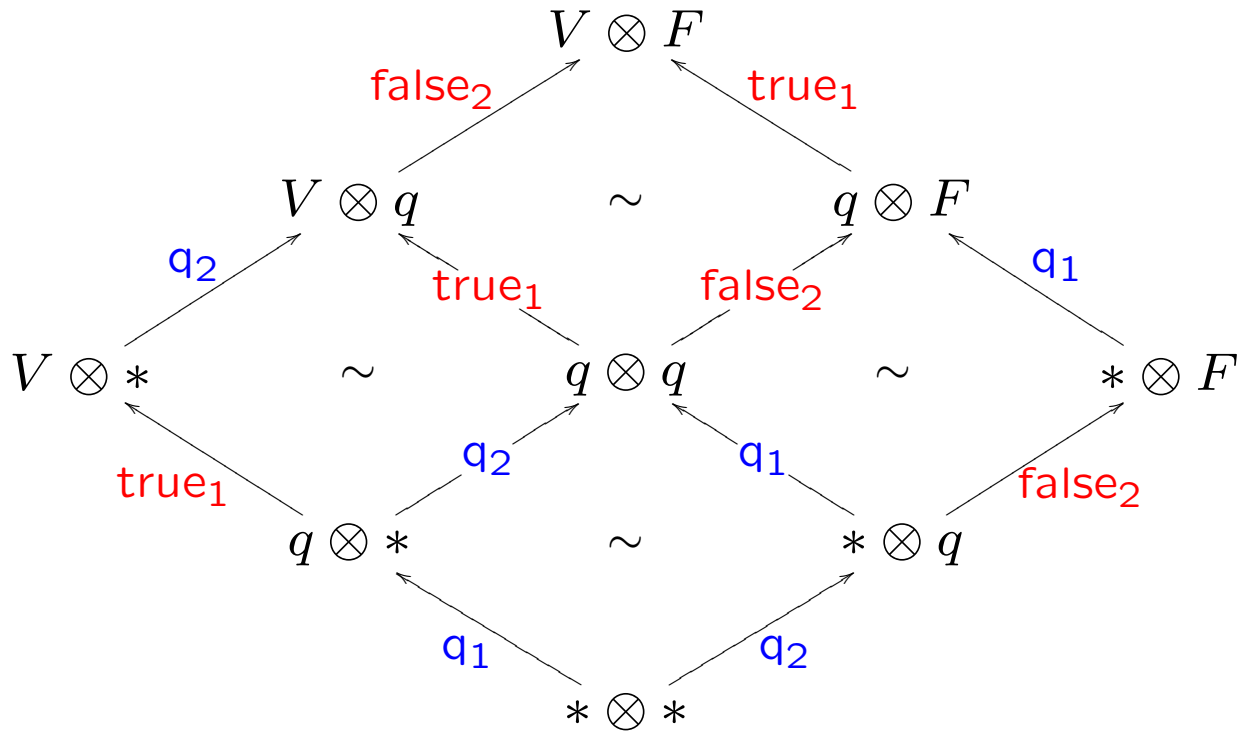
The tensor product of two boolean games \mathbb{B}_1 et \mathbb{B}_2 :



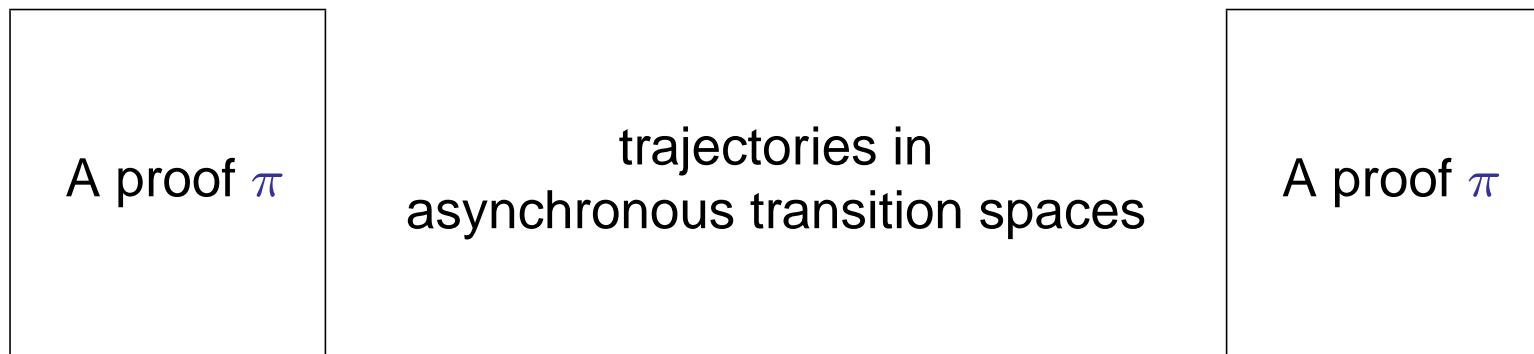
A step towards true concurrency: bend the branches!



True concurrency: tile the diagram!



Asynchronous game semantics



The phenomenon refined: a **truly concurrent** semantics of proofs.

Part III

Innocence in asynchronous games

From trajectories to positions

Asynchronous games

An **asynchronous game** is an event structure equipped with a **polarity** function

$$\lambda : M \longrightarrow \{-1, +1\}$$

indicating whether a move is Player (+1) or Opponent (−1).

Legal plays

A **legal play** is a path

$$* \xrightarrow{m_1} x_1 \xrightarrow{m_2} x_2 \xrightarrow{m_3} \cdots x_{k-1} \xrightarrow{m_k} x_k$$

starting from the empty position $*$ of the transition space, and satisfying:

$$\forall i \in [1, \dots, k], \quad \lambda(m_i) = (-1)^i.$$

So, a legal play is **alternated** and starts by an **Opponent move**.

Strategies

A **strategy** is a set of **legal plays of even length**, such that:

— σ contains **the empty play**,

— σ is **closed under even-length prefix**

$$s \cdot m \cdot n \in \sigma \Rightarrow s \in \sigma,$$

— σ is **deterministic**

$$s \cdot m \cdot n_1 \in \sigma \text{ and } s \cdot m \cdot n_2 \in \sigma \Rightarrow n_1 = n_2.$$

A strategy plays according to the current play.

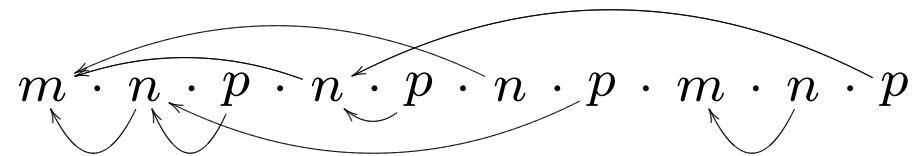
Innocence: strategies with partial information

Full abstraction result [Martin Hyland, Luke Ong, Hanno Nickau, 1994]

Innocence characterizes the interactive behaviour of λ -terms.

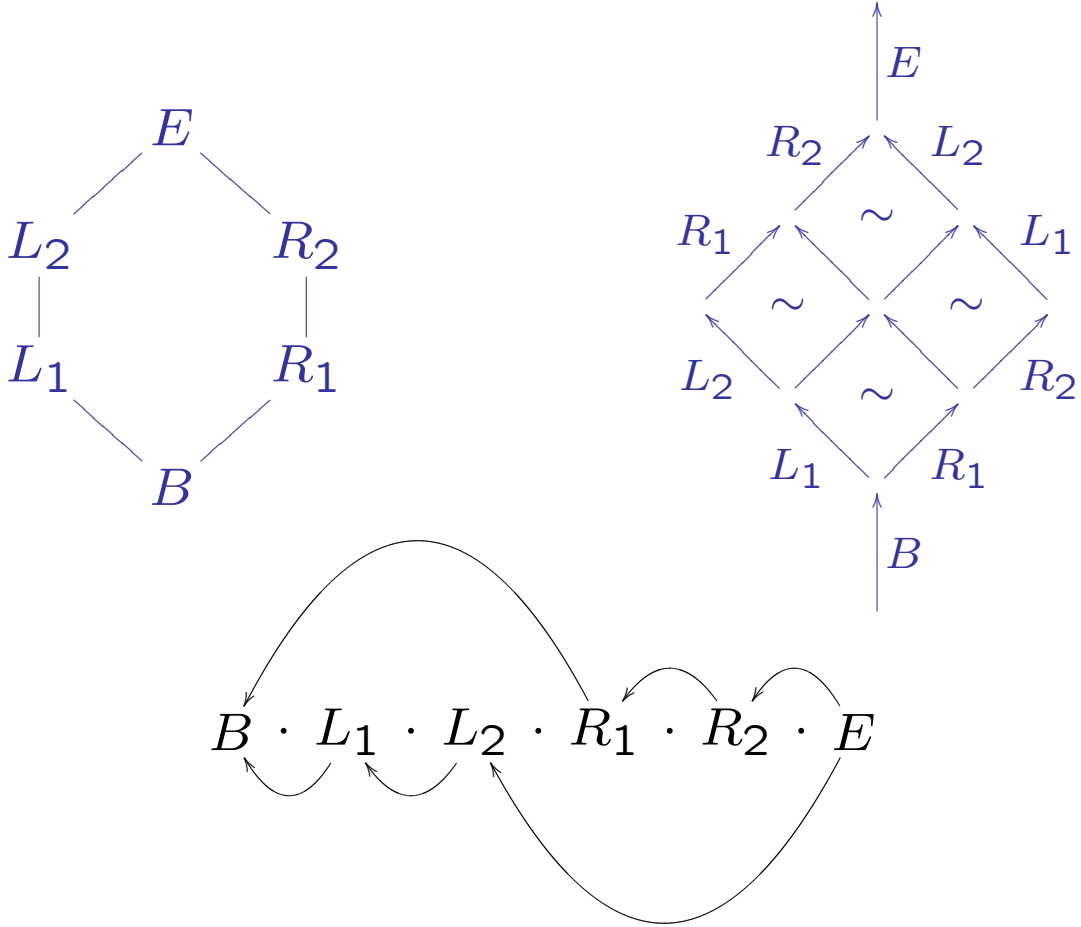
An innocent strategy plays according to the current view.

Where are the pointers in asynchronous games?

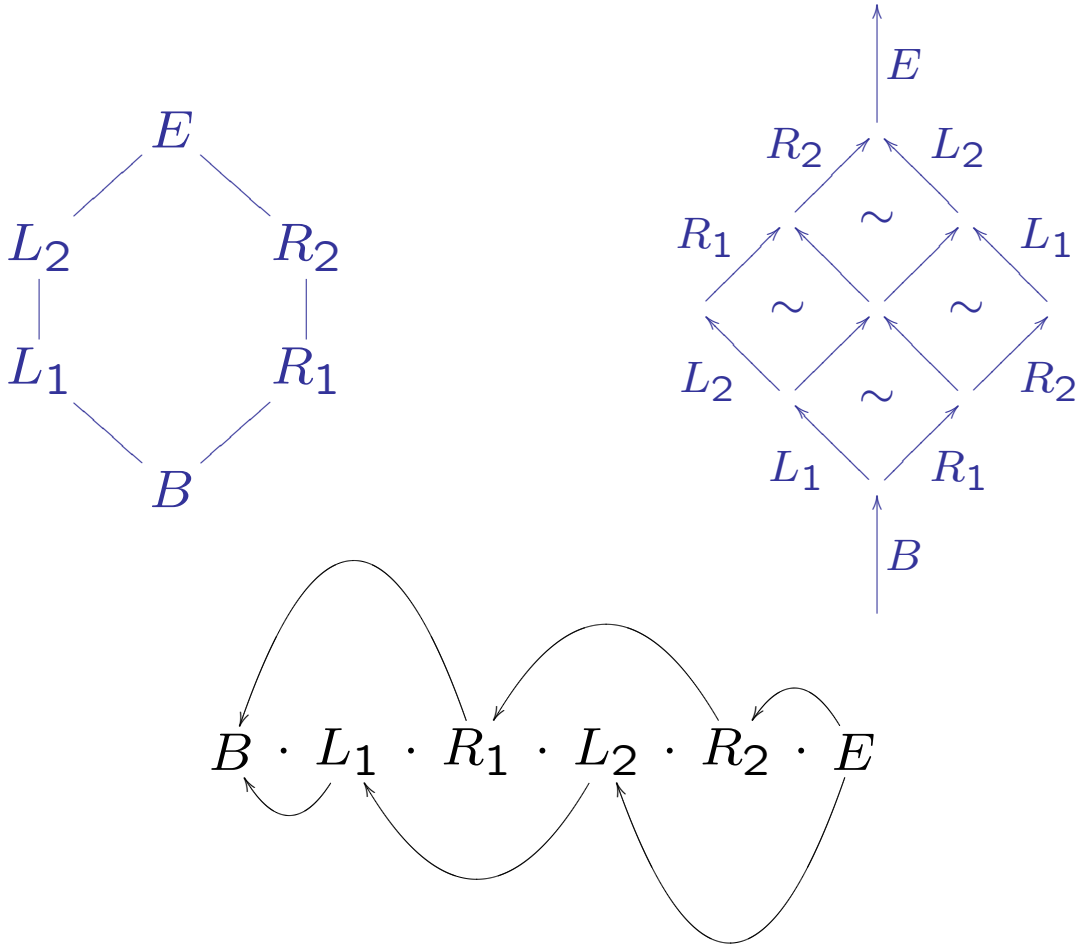


Play = sequence of moves with pointers

Event structure = generalized arena

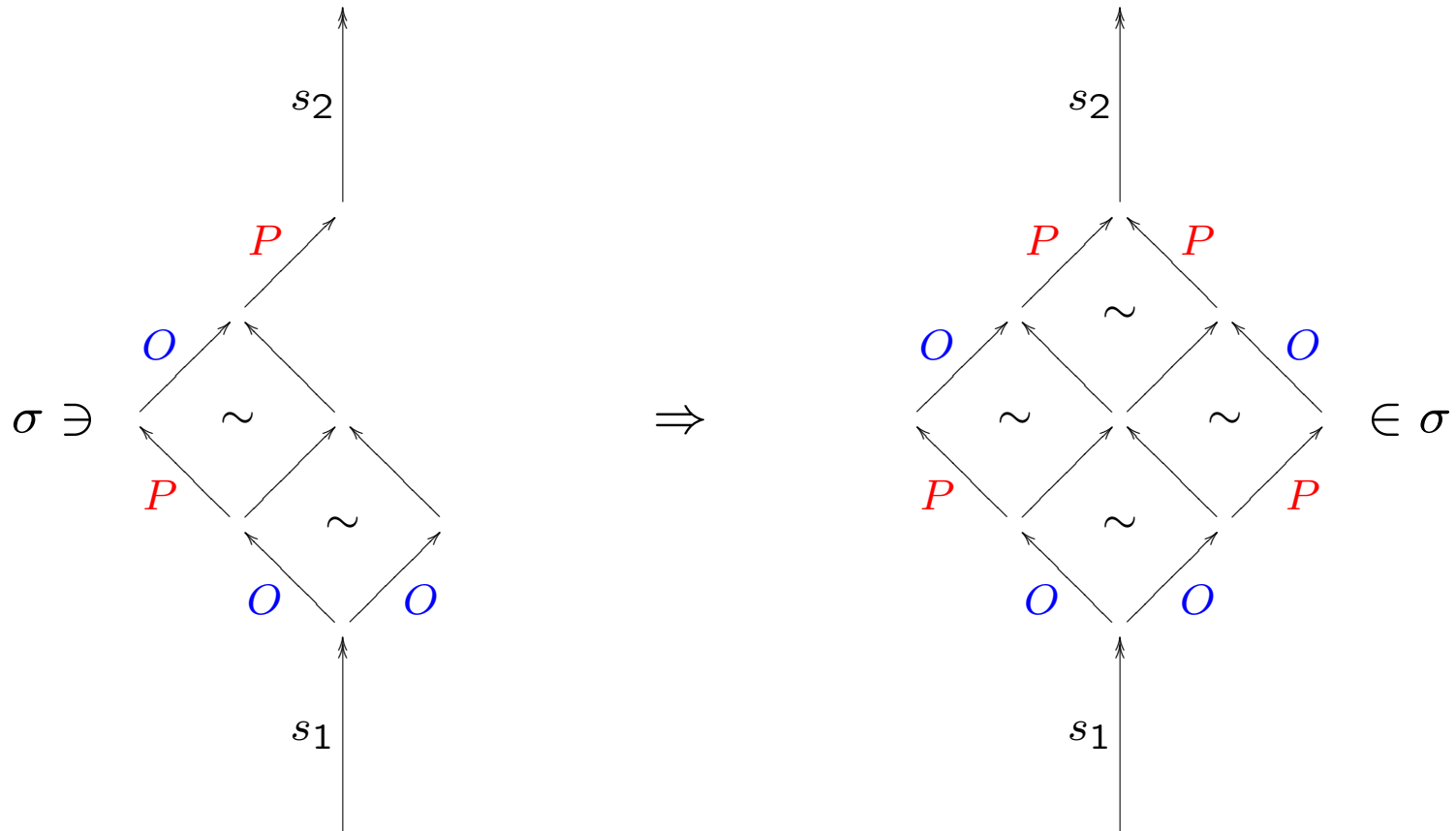


Event structure = generalized arena

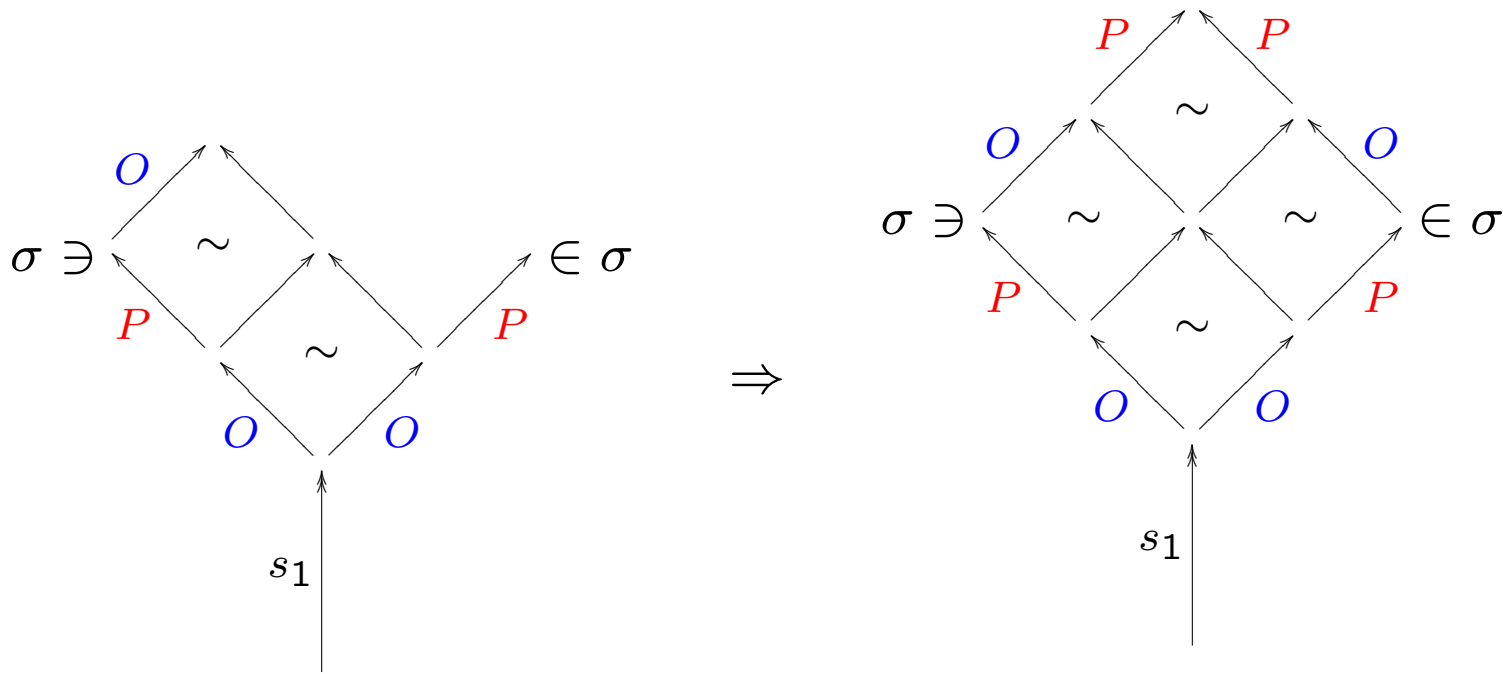


From this follows...

Backward innocence



Forward innocence



Innocent strategies are positional

Definition. A strategy σ is **positional** when for every two plays s_1 and s_2 with same target x :

$$s_1 \in \sigma \text{ and } s_2 \in \sigma \text{ and } s_1 \cdot t \in \sigma \Rightarrow s_2 \cdot t \in \sigma$$

Theorem (by an easy diagrammatic proof)

Every innocent strategy σ is positional

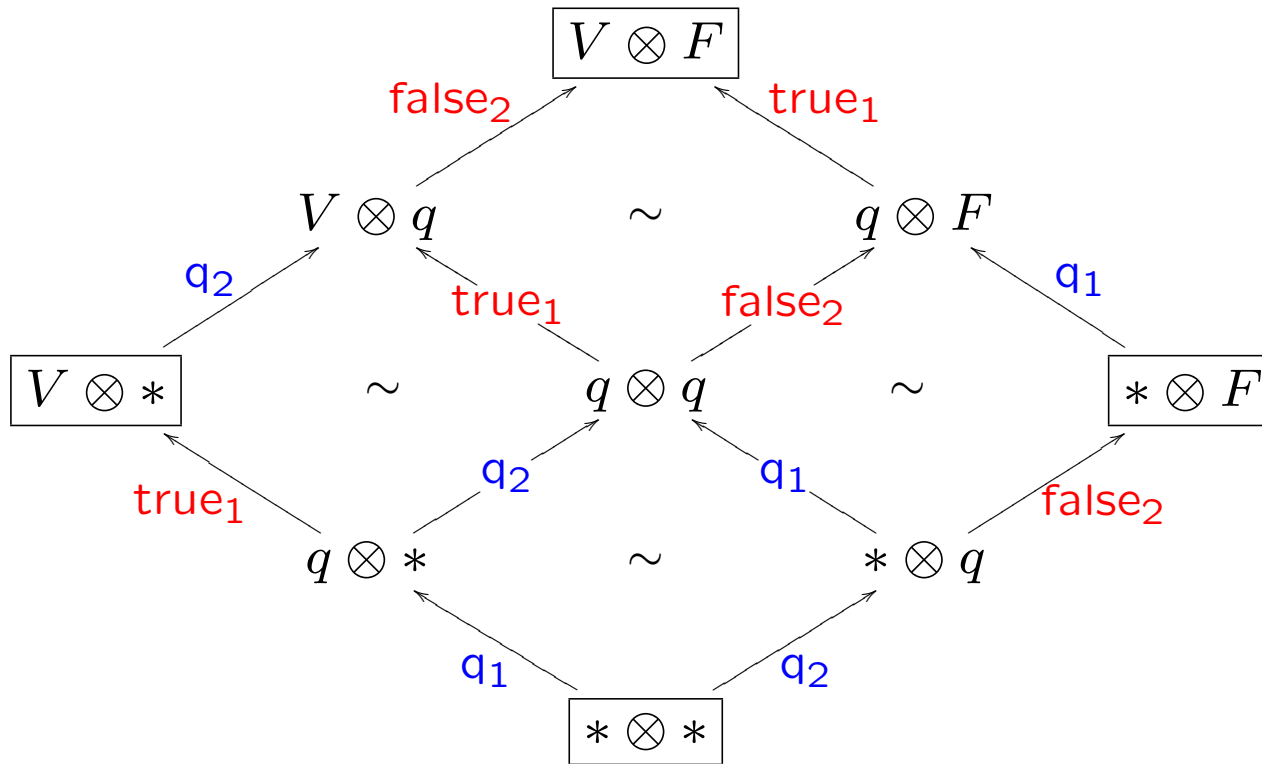
More: An innocent strategy is characterized by the positions it reaches.

A positional characterization of innocence

An innocent strategy σ is a set of **positions** satisfying :

- σ is closed under **union** and **intersection**,
- if $\sigma \ni x \xrightarrow{m} t \xrightarrow{*} z \in \sigma$ and the move m is by **Opponent**, then there exists a unique **Player** move $t \xrightarrow{n} y$ such that $\sigma \ni y \xrightarrow{*} z \in \sigma$.
- the **dual** property (reverse the arrows and the polarities),
- the root position $*$ is element of σ .

An illustration: the strategy $(\text{true} \otimes \text{false})$



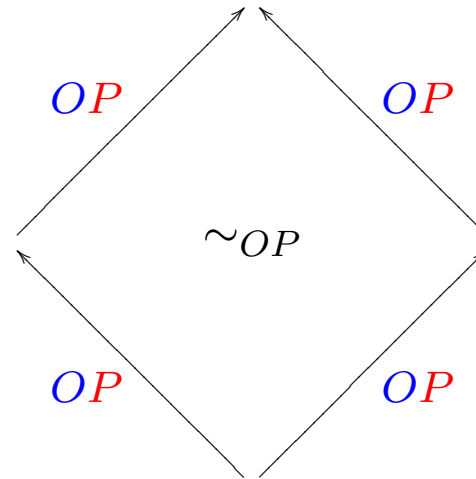
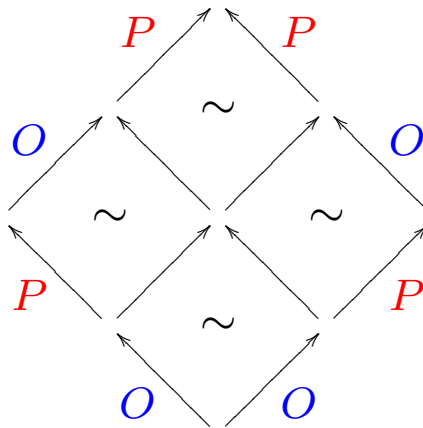
Strategies become **closure operators** on complete lattices as in Abramsky-M. concurrent games.

Part IV

A concurrent account of Böhm trees

Extraction in asynchronous games

The transition space of OP-transitions



A new transition space.

Extraction in the OP-transition space

$$s_1 \cdot O \cdot P \cdot s_2 \cdot m \cdot n \xrightarrow[\sim]{OP} s_1 \cdot s_2 \cdot m \cdot n$$

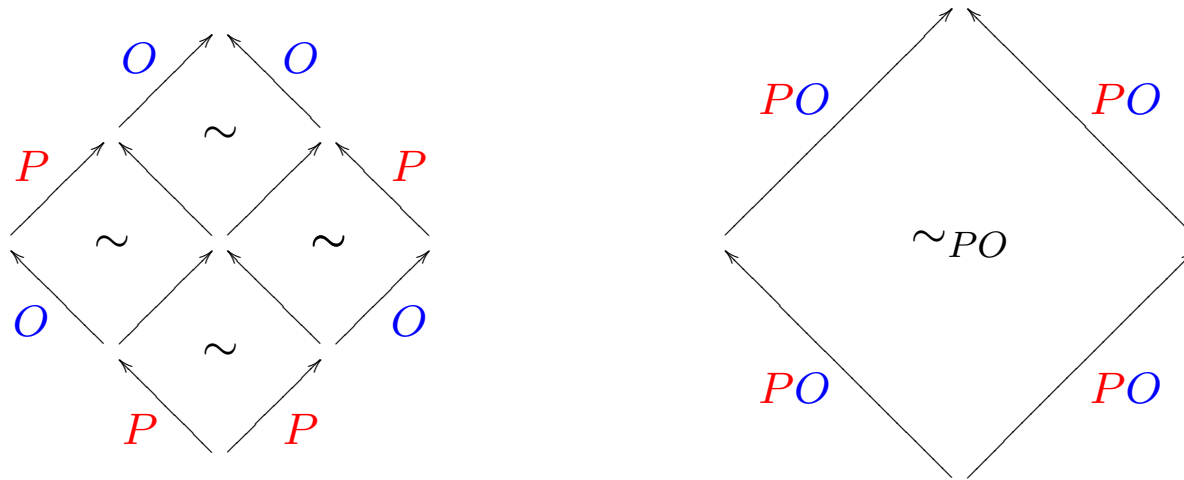
when the two moves O and P do not justify any move in the path $s_2 \cdot m \cdot n$.

Property: the Player view $\lceil s \rceil$ of a play s is the result of the extraction:

$$s \xrightarrow[\sim]{OP} \dots \xrightarrow[\sim]{OP} \lceil s \rceil.$$

Corollary: an OP-event is a Player view.

The transition space of PO-transitions



Dually, extraction $\overset{PO}{\rightsquigarrow}$ computes the Opponent view.

The reconstruction of the Böhm tree

A key observation [Vincent Danos, Hugo Herbelin, Laurent Regnier 1996.]

A Player view is the same thing as a branch of a Böhm tree.

A node in the syntax:

$$\lambda x_1 \cdots \lambda x_m. y P_1 \cdots P_n$$

represents a cluster of Opponent moves:

$$\lambda x_1 \cdots \lambda x_m. y P_1 \cdots P_n \mapsto y P_1 \cdots P_n$$

and a cluster of Player moves:

$$y P_1 \cdots P_n \mapsto y, P_1, \cdots, P_n.$$

Immediate consequence

Every legal play s with final position x defines two Böhm trees:

A Program $\langle s |$ \longleftrightarrow An Environment $|s \rangle$

The position x is obtained by letting $\langle s |$ and $|s \rangle$ interact together.

One needs a **locative** and **non uniform** λ -calculus for that.

Part V

A game model of linear logic

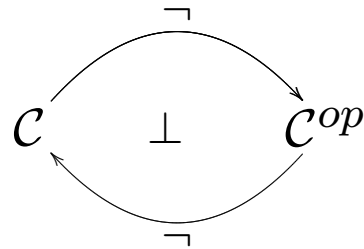
A commutative continuation monad

An adjunction

Every object \perp in a symmetric monoidal closed category \mathcal{C} defines a functor:

$$\begin{array}{ccc} \neg & : & \mathcal{C} & \longrightarrow & \mathcal{C}^{op}. \\ & & A & \longmapsto & A \multimap \perp \end{array}$$

This induces an adjunction



The continuation monad

The resulting continuation monad

$$A \mapsto \neg\neg A$$

is strong:

$$A \otimes \neg\neg B \xrightarrow{lst} \neg\neg(A \otimes B) \qquad \neg\neg A \otimes B \xrightarrow{rst} \neg\neg(A \otimes B)$$

The continuation monad

Theorem: [Masahito Hasegawa]

The kleisli category is $*$ -autonomous when the continuation monad is commutative.

A categorification of the phase space semantics.

Conclusion

Here, a conceptual and concurrent reconstruction of Böhm trees.

Currently:

Non alternating games [with Samuel Mimram]

Operads in categories of games [with Nicolas Tabareau].

A categorical analysis of innocence [with Russ Harmer and Martin Hyland]

Comparison with the work by Claudia Faggian, Pierre-Louis Curien, Martin Hyland, Andrea Schalk, Dan Ghica, Andrej Murawski, Jim Laird.