Sémantique des jeux asynchrones et réécriture 2-dimensionnelle

Soutenance de thèse de doctorat

Samuel Mimram

Laboratoire PPS (CNRS – Université Paris Diderot)

1er 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++;
    }
  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*sizeof(void*);
  /* 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!

The Curry-Howard correspondence

- Typing programs can avoid some errors

```
1  :  int
```

The Curry-Howard correspondence

- Typing programs can avoid some errors

$$\texttt{fun } x \longrightarrow x\texttt{+1} \quad : \quad \texttt{int} \Rightarrow \texttt{int}$$

The Curry-Howard correspondence

- Typing programs can avoid some errors

$$\texttt{fun}\ x \longrightarrow \texttt{not}\ x \quad : \quad \texttt{bool} \Rightarrow \texttt{bool}$$

The Curry-Howard correspondence

- Typing programs can avoid some errors

$$\texttt{fun}\ x \longrightarrow \texttt{not}\ x \quad : \quad \texttt{bool} \Rightarrow \texttt{bool}$$

- The Curry-Howard correspondence:

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

- Studying programs is the same as studying proofs.

Denotational semantics

A **model** interprets

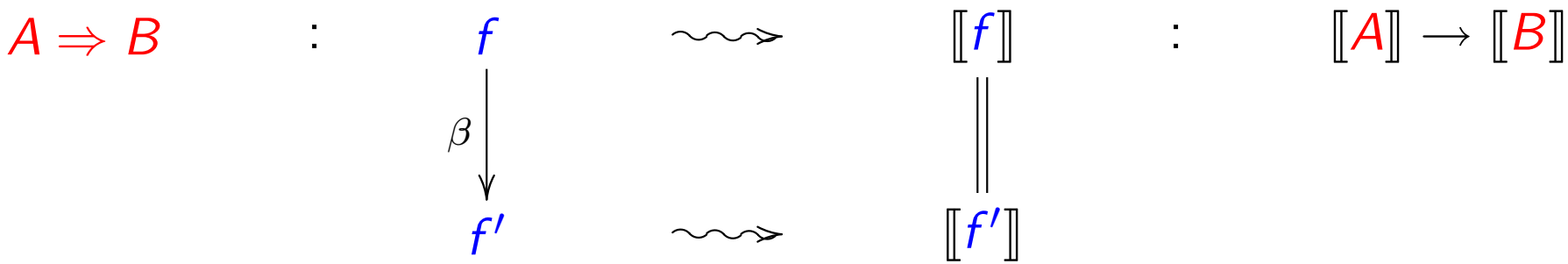
- a type A as a *computation space* $[\![A]\!]$
- a program $f : A \Rightarrow B$ as a *transformation* $[\![f]\!] : [\![A]\!] \longrightarrow [\![B]\!]$

$$A \Rightarrow B \quad : \quad f \quad \rightsquigarrow \quad [\![f]\!] \quad : \quad [\![A]\!] \longrightarrow [\![B]\!]$$

Denotational semantics

A **denotational model** interprets

- a type A as a *computation space* $[\![A]\!]$
- a program $f : A \Rightarrow B$ as a *transformation* $[\![f]\!] : [\![A]\!] \longrightarrow [\![B]\!]$



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*.

$$(\texttt{fun}\ x \longrightarrow \texttt{not}\ x)\texttt{false} \rightsquigarrow \texttt{true}$$
$$(\texttt{fun}\ x \longrightarrow \texttt{not}\ x)\texttt{true} \rightsquigarrow \texttt{false}$$

$\Rightarrow$ 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

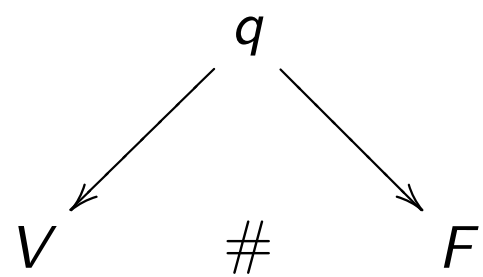
- A type A is interpreted as a **game**.
- A program $f : A$ is interpreted as a **strategy** playing on the game associated to A.

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*.

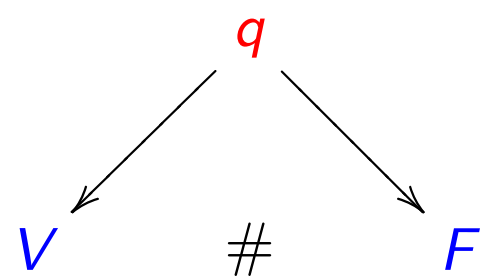


Definition

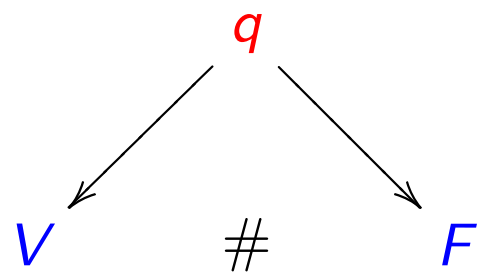
A **game** is

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

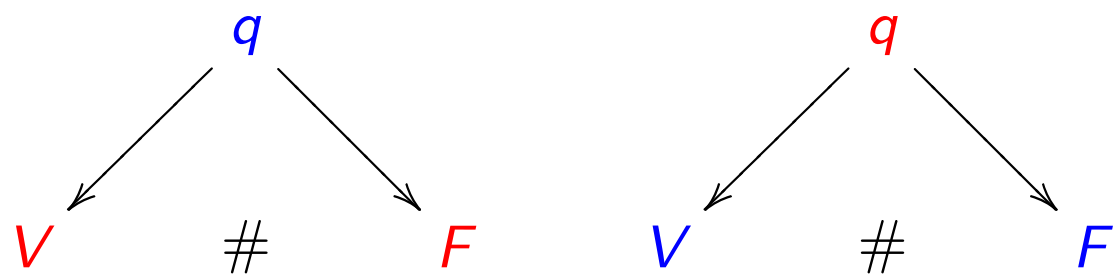
$\texttt{bool} \quad =$



The game `bool` :



The game $\texttt{bool} \Rightarrow \texttt{bool}$:



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`:

$$\texttt{bool} \quad \Rightarrow \quad \texttt{bool}$$

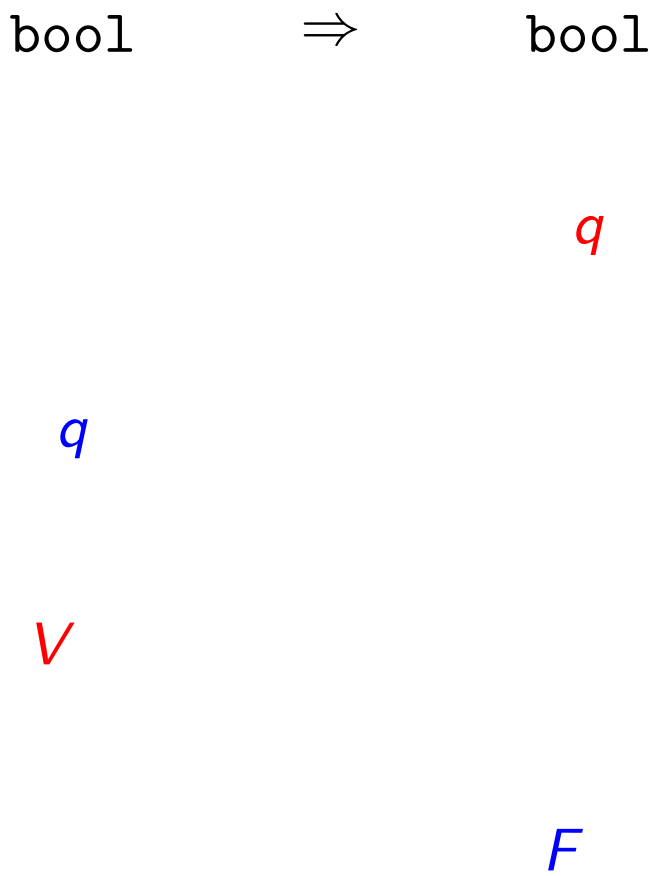
The strategy `not`:

$$\texttt{bool} \quad \Rightarrow \quad \texttt{bool}$$

q

q

The strategy `not`:



The strategy `not`:

`bool` $\Rightarrow$ `bool`

q

q

F

V

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):

$$\frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}(\parr) \qquad \frac{\vdash \Gamma_1, A \qquad \vdash \Gamma_2, B}{\vdash \Gamma_1, \Gamma_2, A \otimes B}(\otimes)$$

$$\frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \,\&\, B}(\&) \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}(\oplus)$$

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),

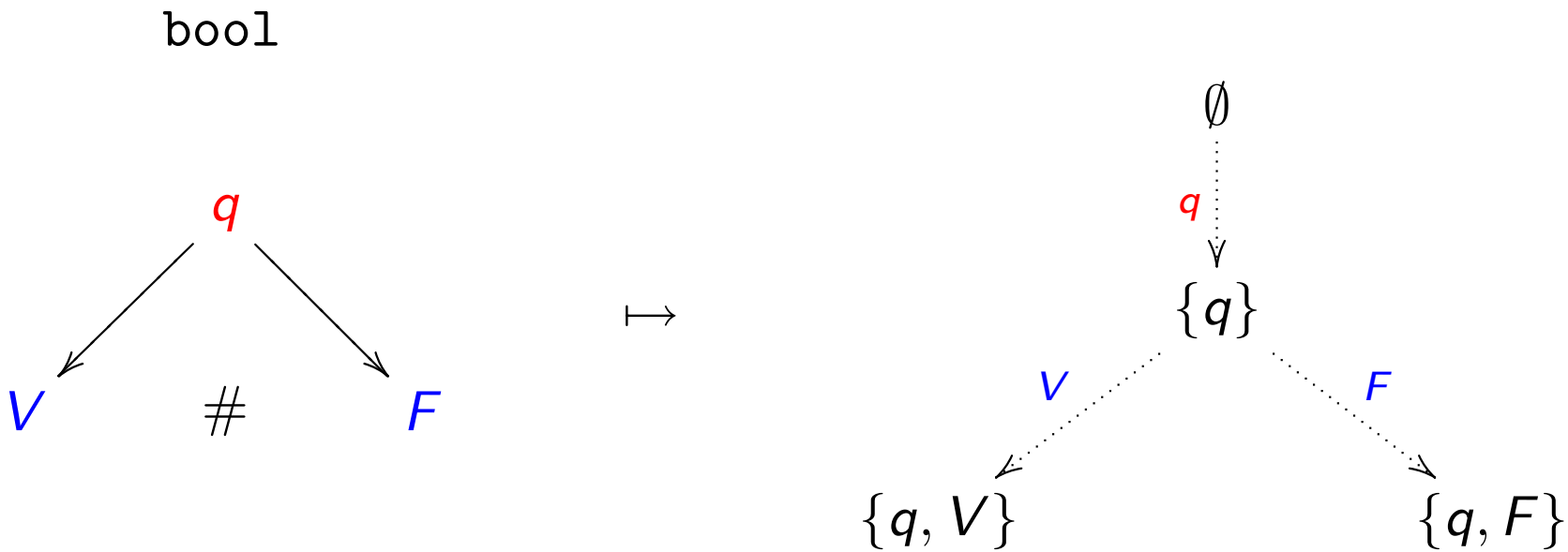
in which we characterize

- a fully complete model of MLL (without units),
- HO innocent strategies in LL,

by using ideas coming from

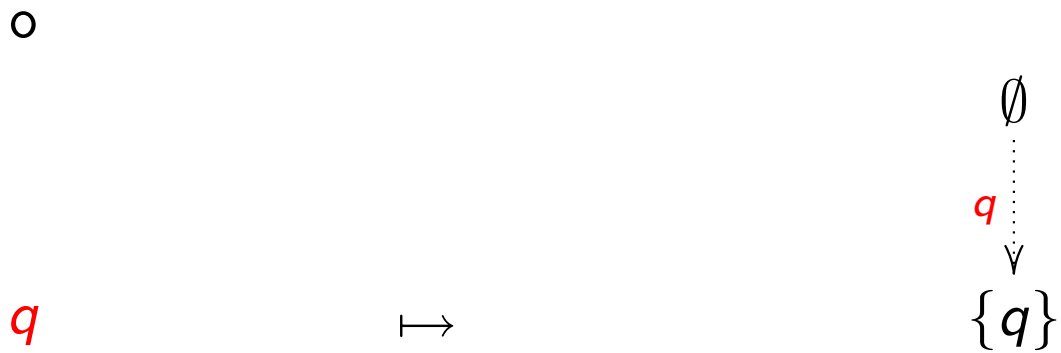
- game semantics,
- concurrency theory,
- linear logic,
- category theory.

The asynchronous graph of a game



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

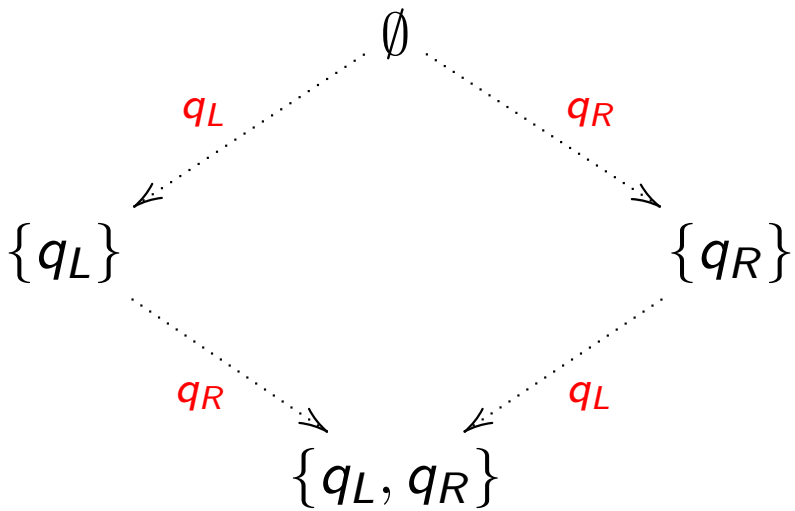
The asynchronous graph of a game



The asynchronous graph of a game

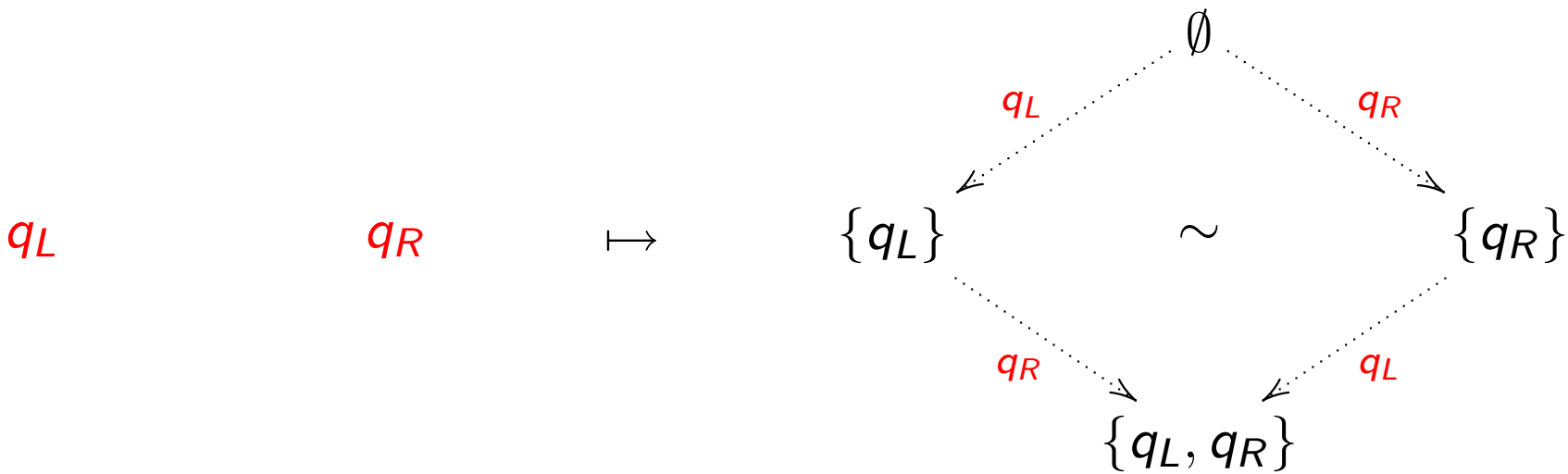
$$\circ \otimes \circ$$

$q_L \qquad q_R \qquad \mapsto$



The asynchronous graph of a game

$\circ \otimes \circ$



Implementations of conjunction

Left implementation of conjunction:

$$\texttt{bool} \quad \otimes \quad \texttt{bool} \quad \multimap \quad \texttt{bool}$$

q

q_L

V_L

q_R

F_R

F

Implementations of conjunction

Right implementation of conjunction:

$$\texttt{bool} \quad \otimes \quad \texttt{bool} \quad \multimap \quad \texttt{bool}$$

q

q_R

F_R

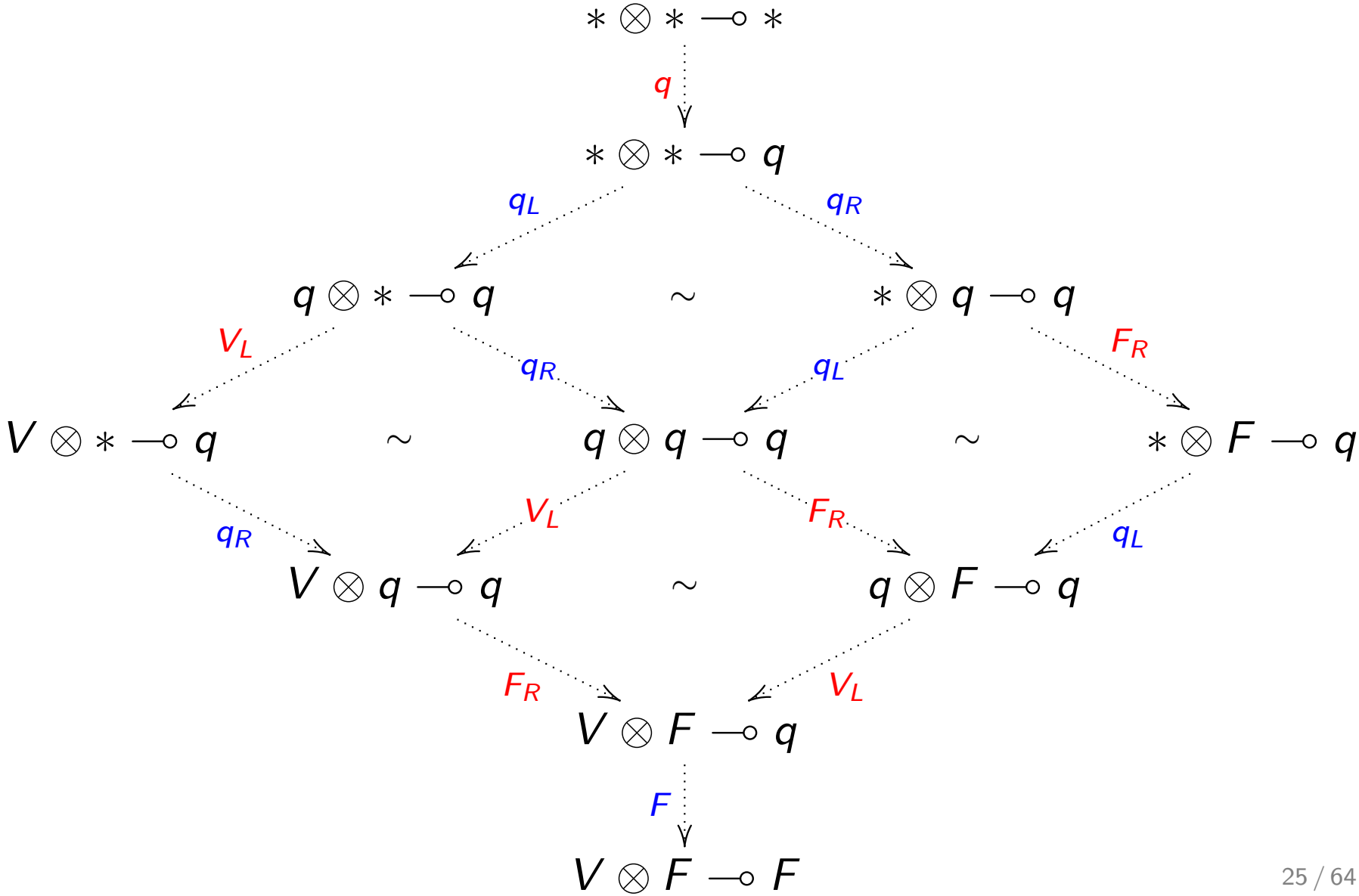
q_L

V_L

F

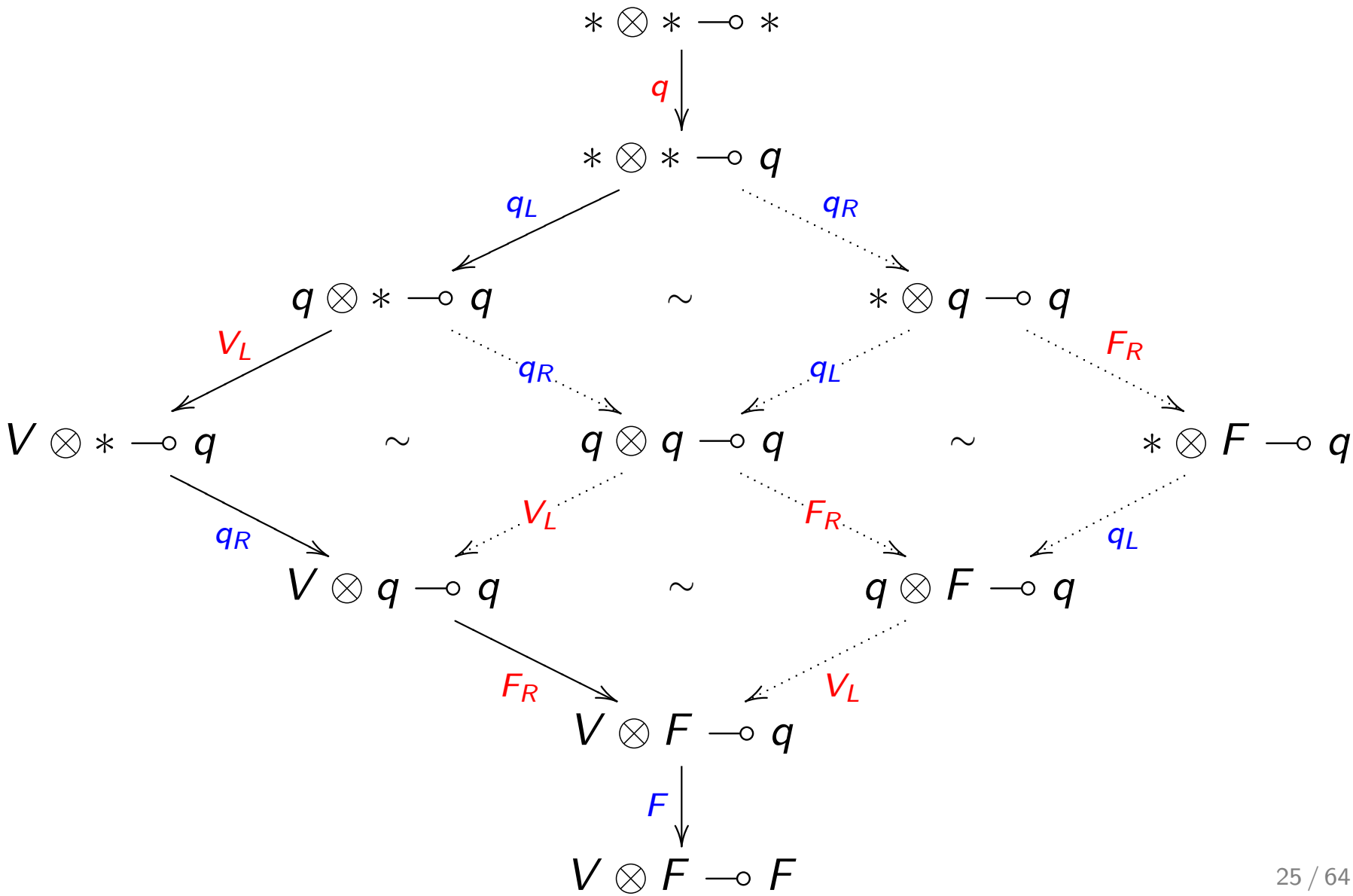
Strategies of conjunction

The game $\mathtt{bool} \otimes \mathtt{bool} \multimap \mathtt{bool}$ contains eight subgraphs:



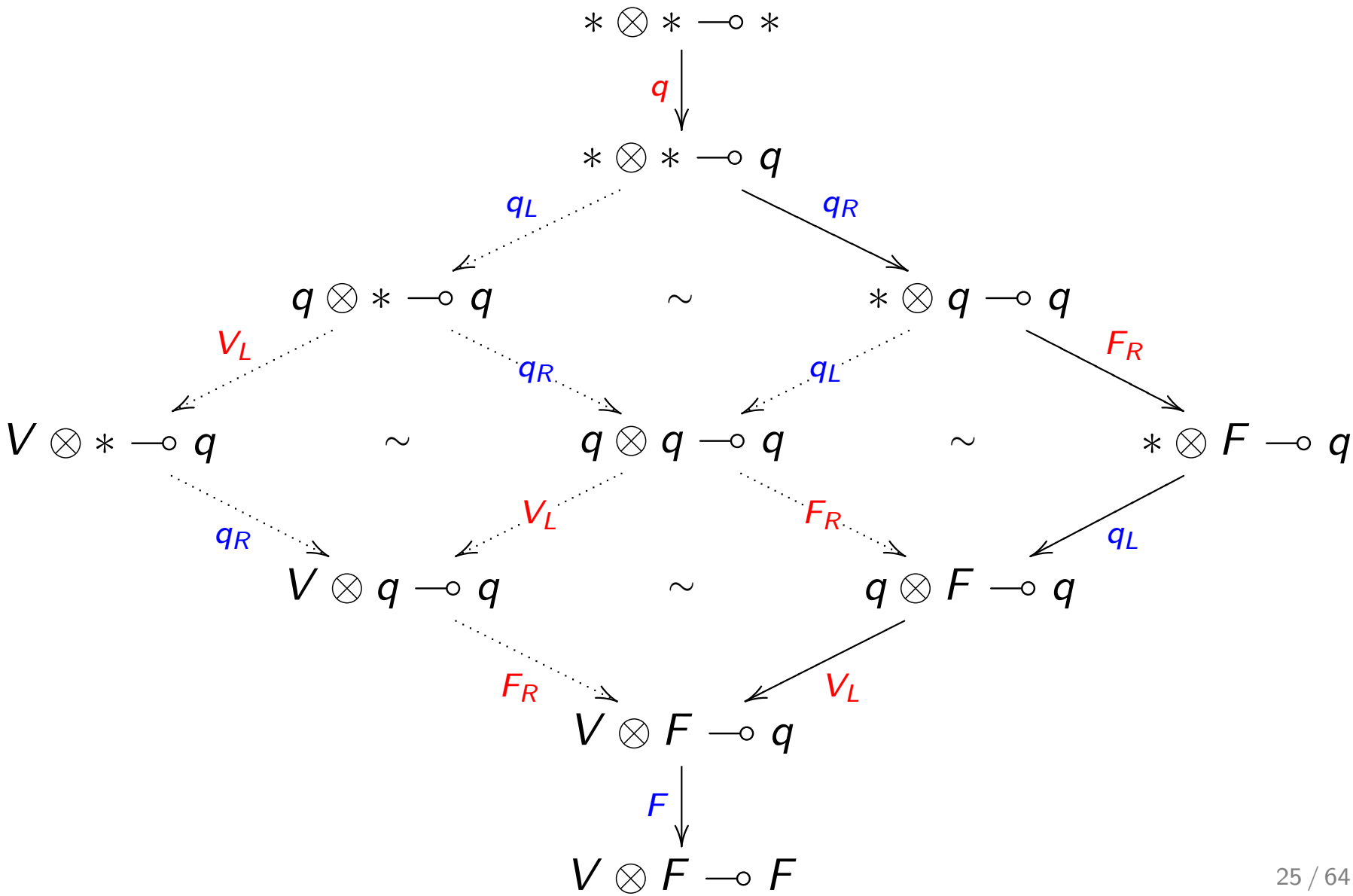
Strategies of conjunction

Left implementation of conjunction:



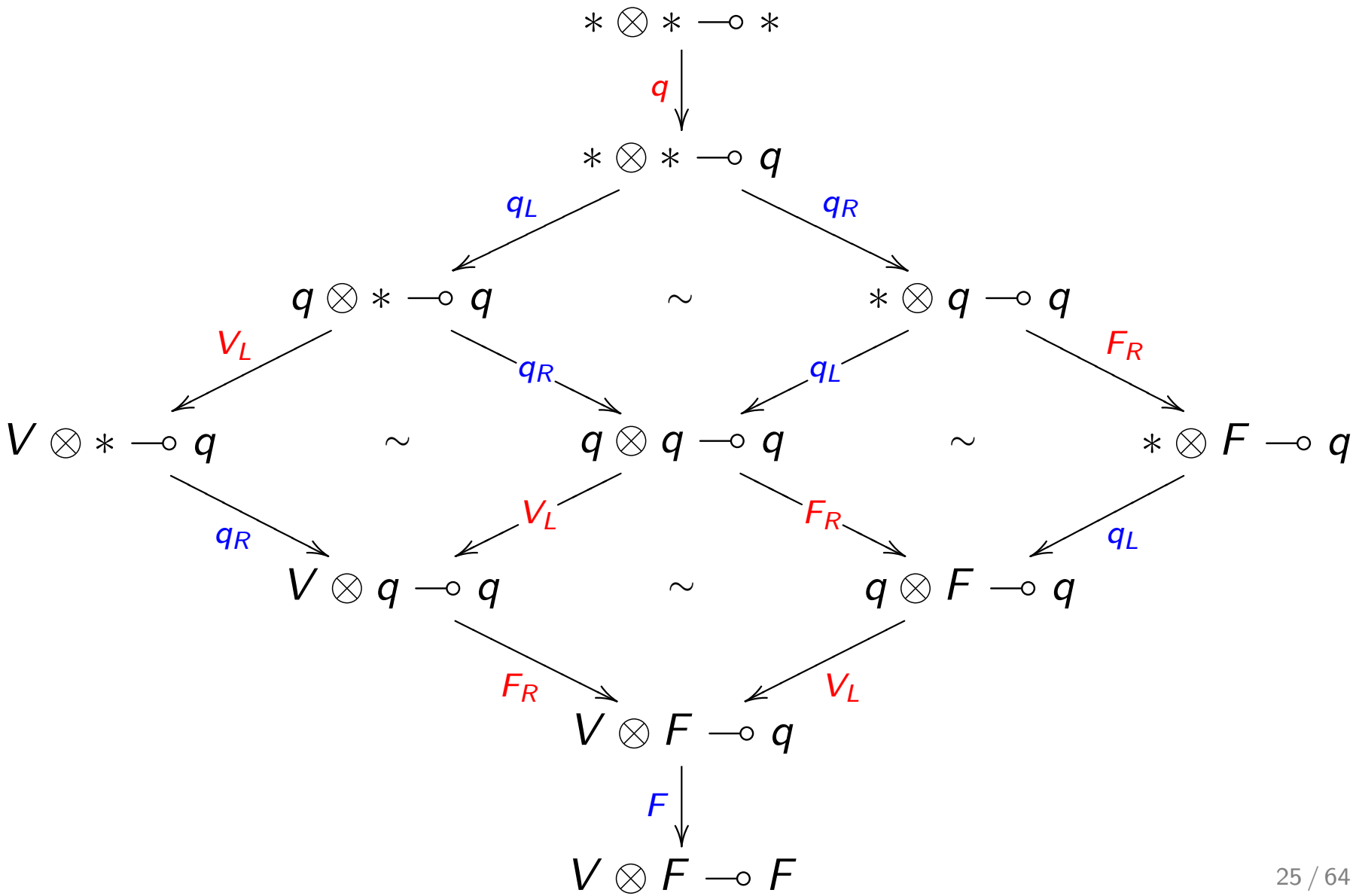
Strategies of conjunction

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.

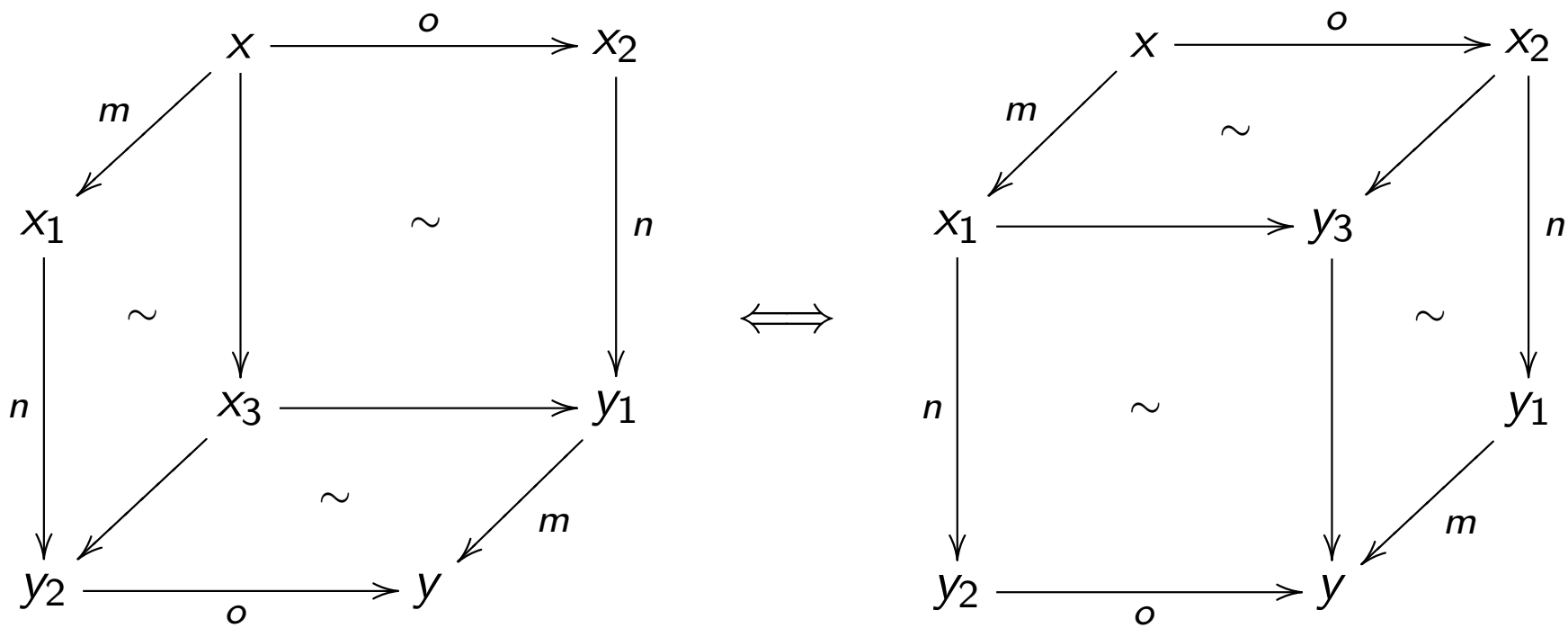
An MLL proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{A^*, A} \quad \dfrac{\vdots}{B^*, B}}{A^* \otimes B^*, A, B}(\otimes)}{A^* \otimes B^*, A \parr B}(\parr) \quad \dfrac{\vdots}{C^*, C}}{A^* \otimes B^*, C^*, (A \parr B) \otimes C}(\otimes)}{(A^* \otimes B^*) \parr C^*, (A \parr B) \otimes C}(\parr)}{((A^* \otimes B^*) \parr C^*) \parr ((A \parr B) \otimes C)}(\parr)$$

- 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.

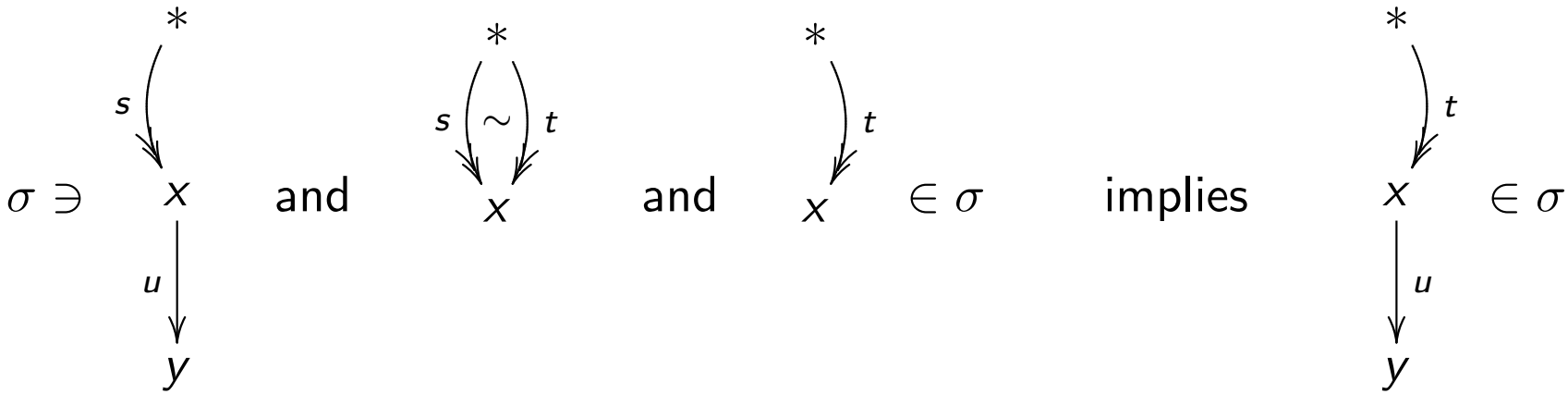
Asynchronous games

Definition

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

We are going to consider strategies σ which

- are **positional** (or *history-free*):



  i.e. characterized by the subgraph of the game they explore.

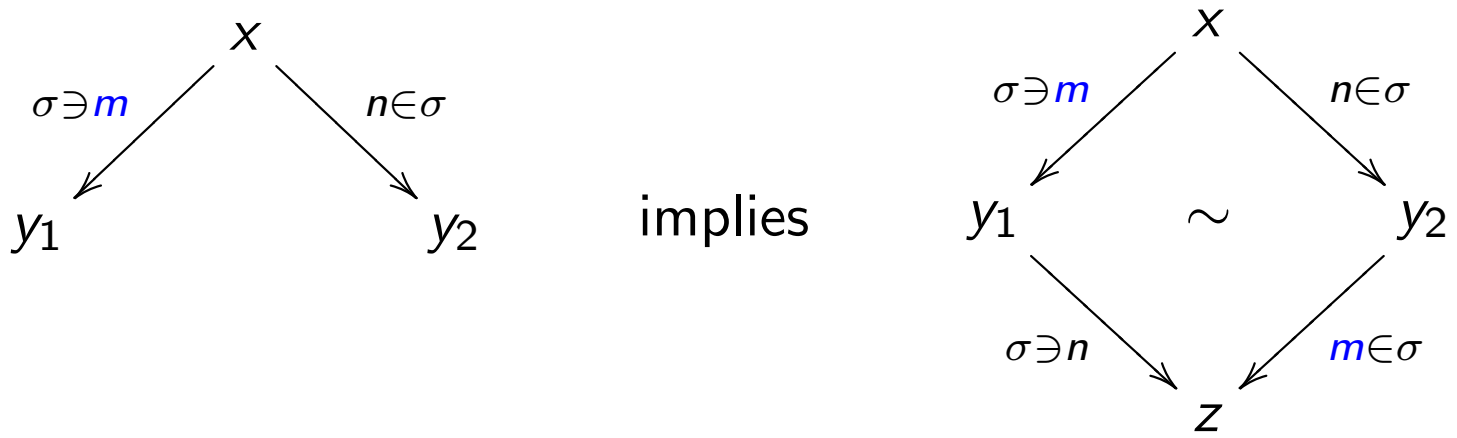
- satisfy the **Cube property**.

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 σ 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!

$$A \otimes B \quad = \quad A \parr B$$

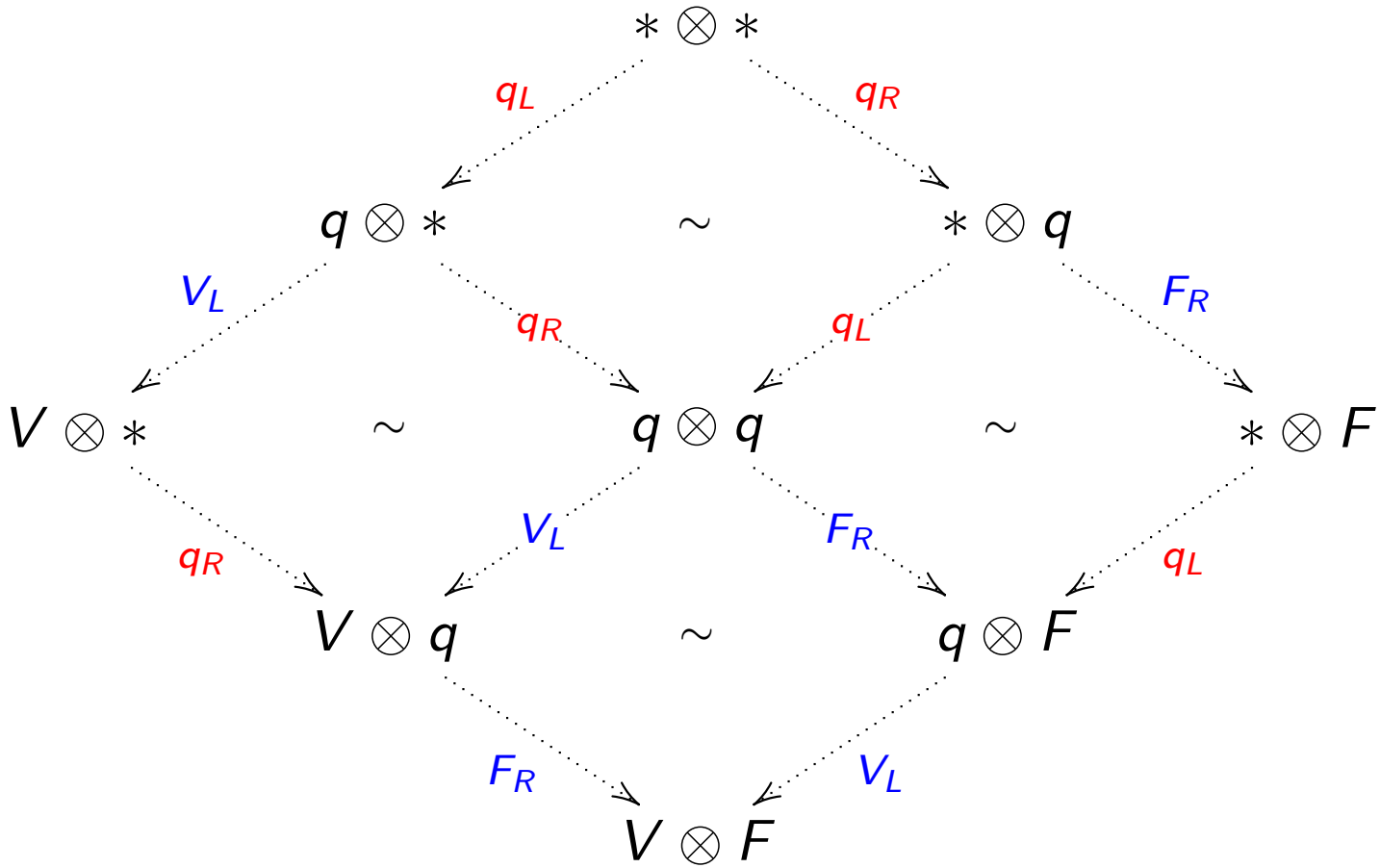
Halting positions

In the spirit of the relational model, a strategy σ should be characterized by its set σ° of halting positions.

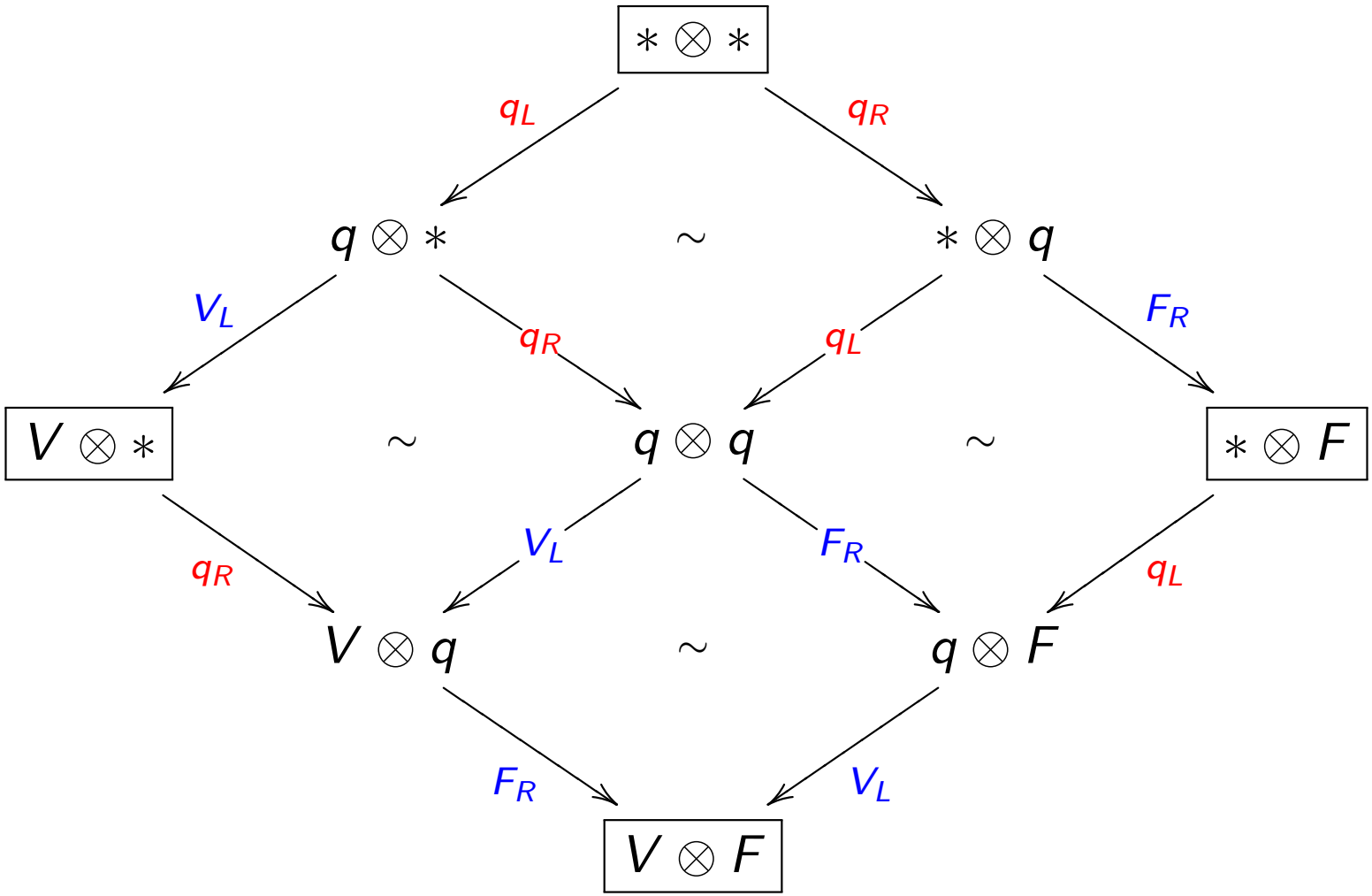
Definition

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

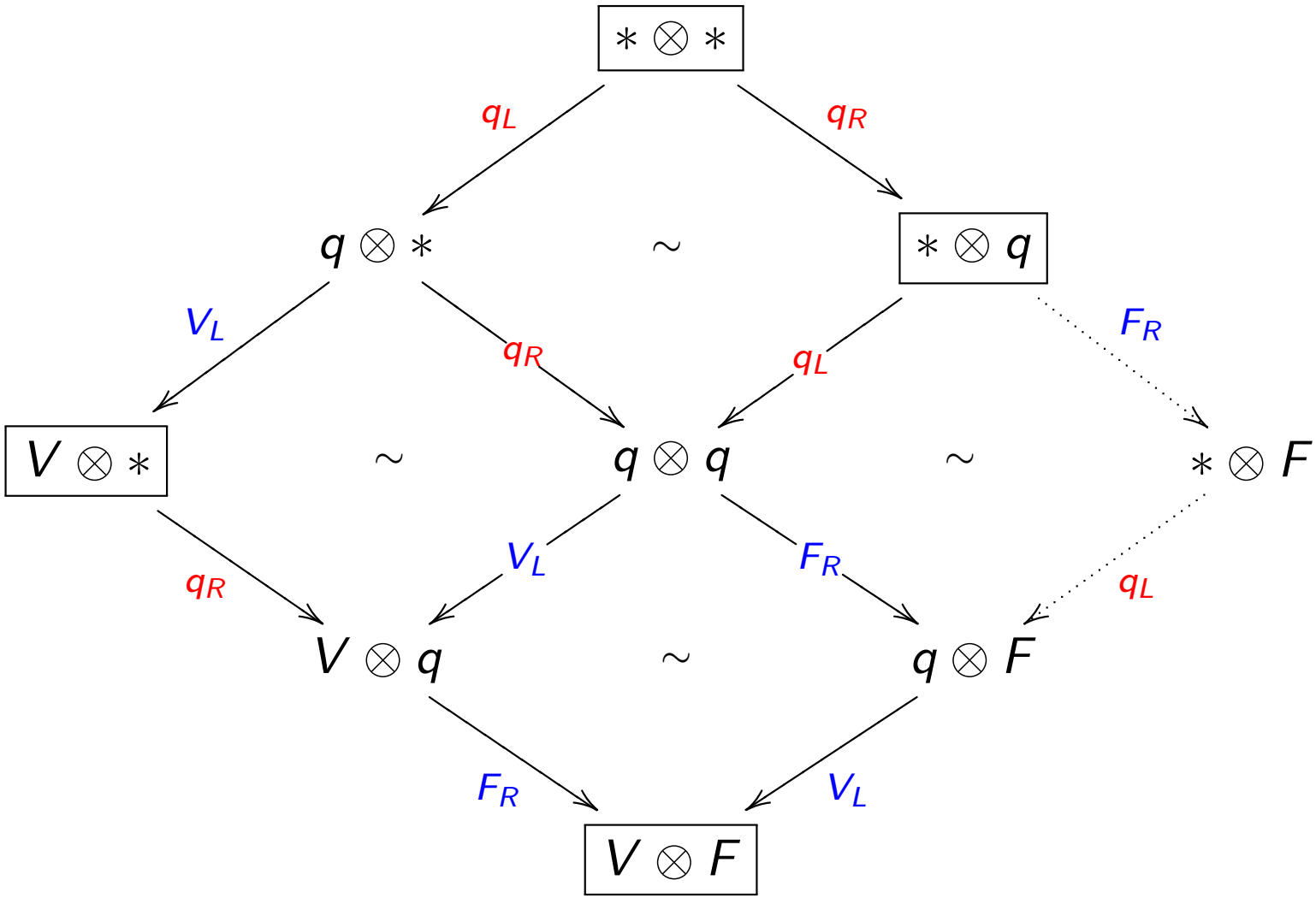
The game $\texttt{bool} \otimes \texttt{bool}$ contains the subgraph:



The pair $\texttt{true} \otimes \texttt{false}$:



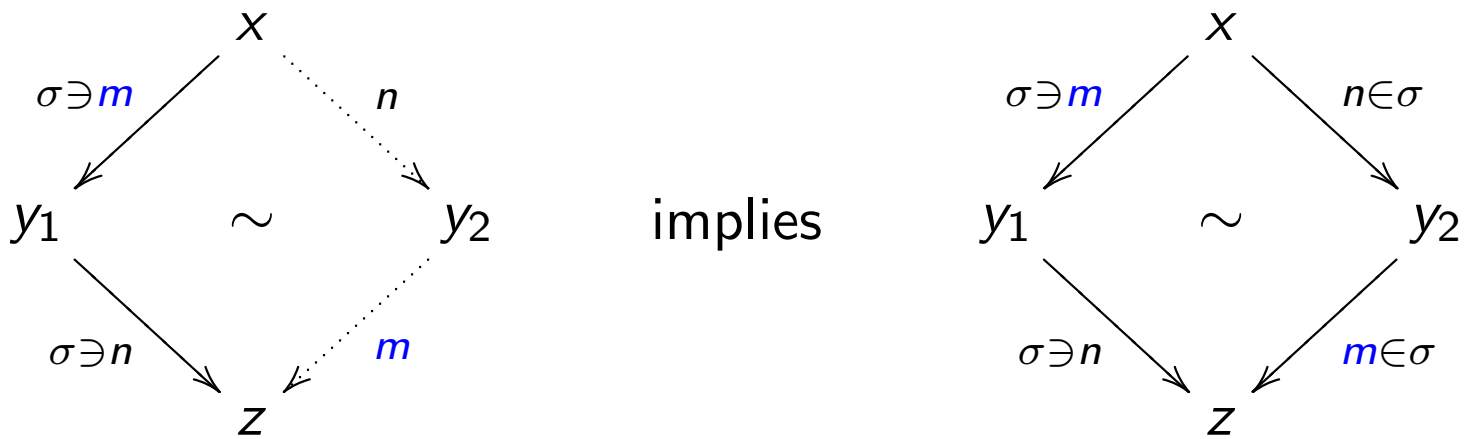
The left biased pair $\texttt{true} \oslash \texttt{false}$:



Courteous strategies

Definition

An ingenuous strategy σ is **courteous** when it satisfies



where m is a Player move.

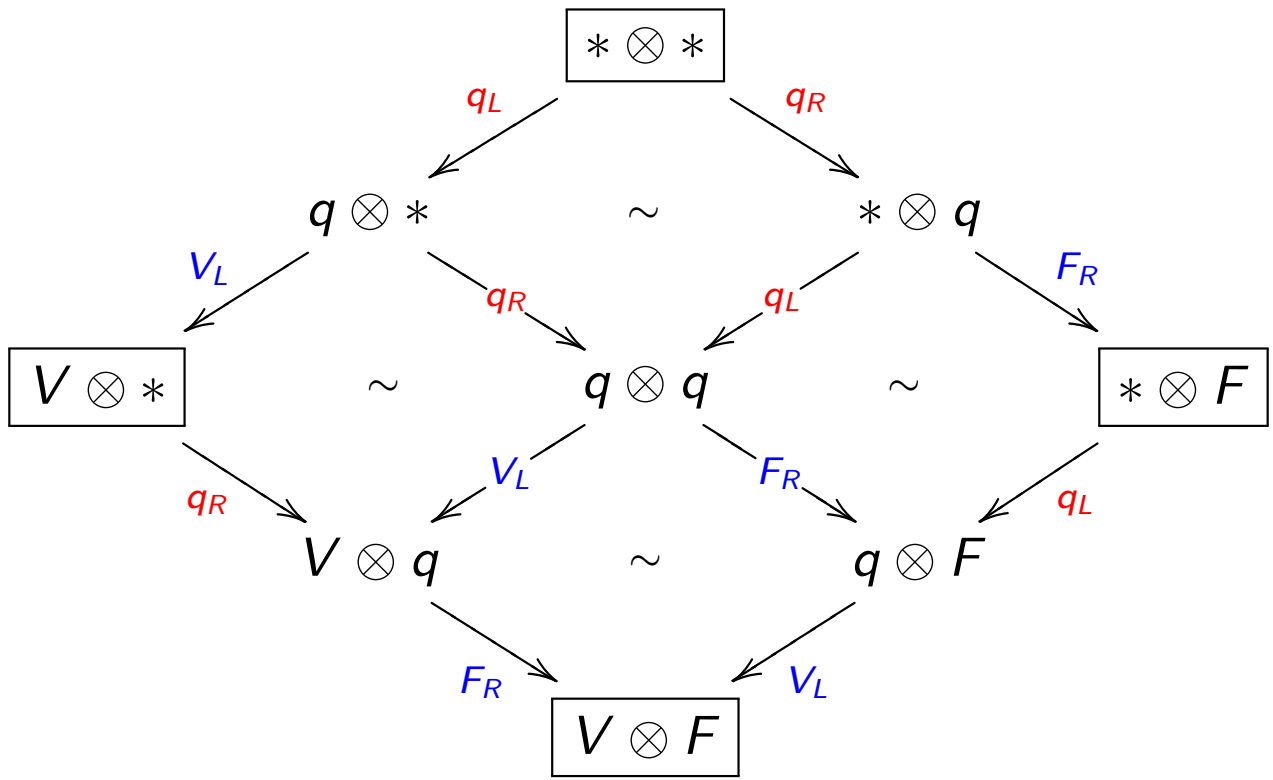
Theorem

A courteous ingenuous strategy σ is characterized by its set σ° of halting positions.

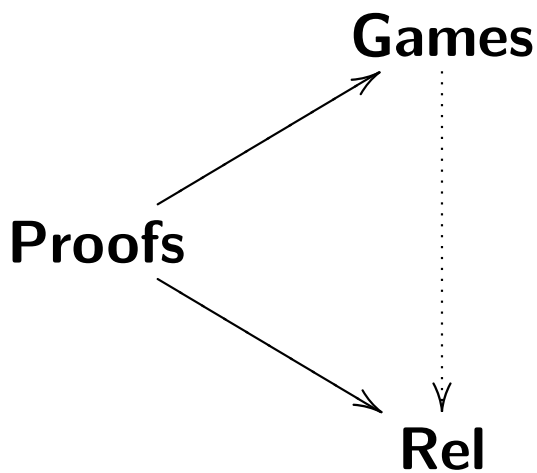
Concurrent strategies

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 $\texttt{true} \oslash \texttt{false}$ leads to a deadlock.

Scheduled strategies: avoiding deadlocks

Composing the *right* implementation of the conjunction with the *left* biased pair $\texttt{true} \oslash \texttt{false}$ leads to a deadlock.

Scheduled strategies: avoiding deadlocks

Composing the *right* implementation of the conjunction with the *left* biased pair $\texttt{true} \oslash \texttt{false}$ leads to a deadlock.

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

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

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:

$$A \quad ::= \quad \forall x.A \quad | \quad \exists x.A \quad | \quad P(x) \quad | \quad \dots$$

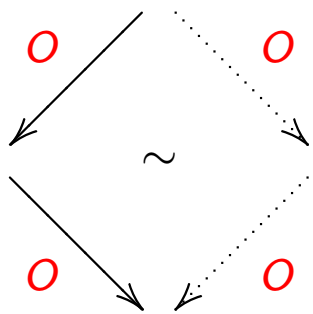
- Rules:

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

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

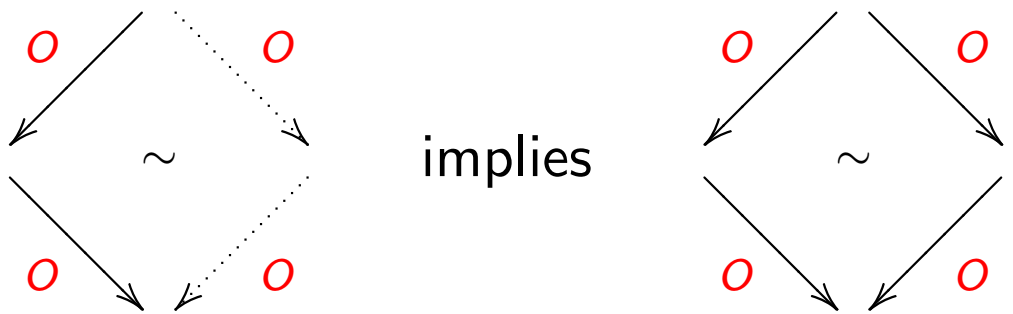
Causality in proofs

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



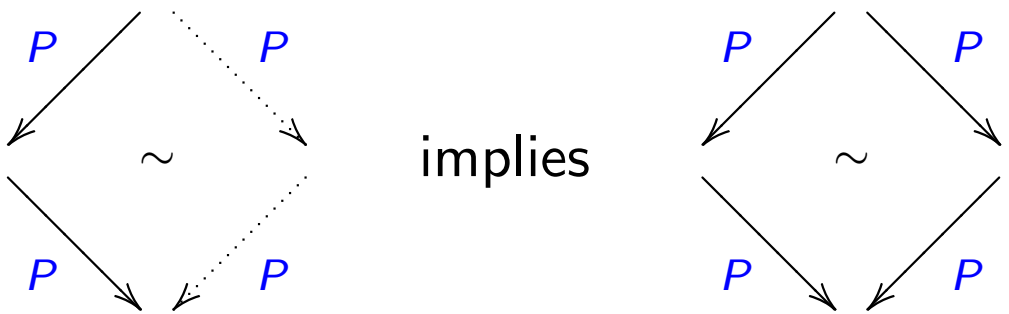
Causality in proofs

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



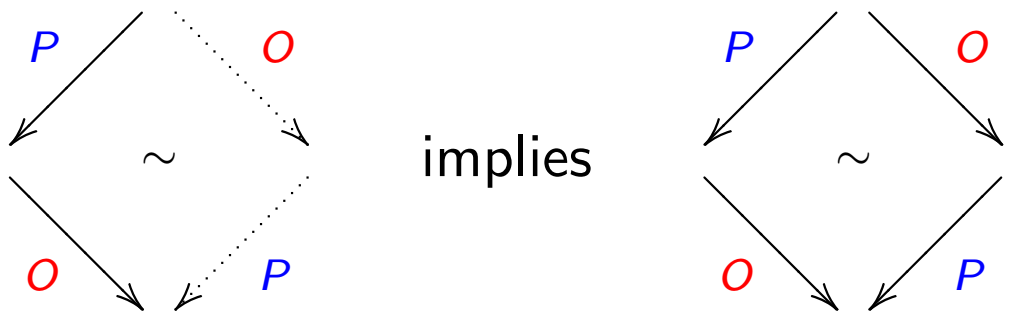
Causality in proofs

$$\dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash A[t/x], B[t'/y], \Delta}}{\Gamma \vdash A[t/x], \exists y.B, \Delta}(\exists)}{\Gamma \vdash \exists x.A, \exists y.B, \Delta}(\exists) \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\pi}{\Gamma \vdash A[t/x], B[t'/y], \Delta}}{\Gamma \vdash \exists x.A, B[t'/y], \Delta}(\exists)}{\Gamma \vdash \exists x.A, \exists y.B, \Delta}(\exists)$$



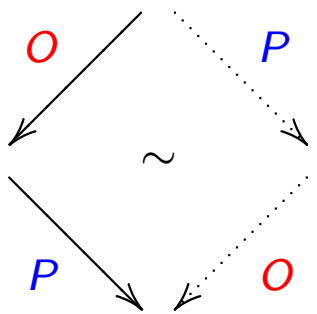
Causality in proofs

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



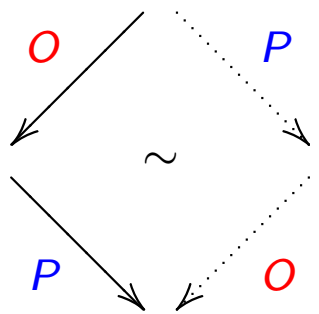
Causality in proofs

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



Causality in proofs

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



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

Causality in proofs

Essential causal dependencies induced by proofs are

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

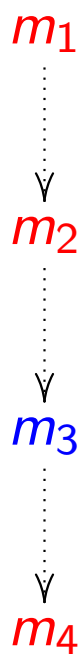
where the witness t given for y admits x as free variable.

Games

A formula

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

will be interpreted by the game (= polarized poset)

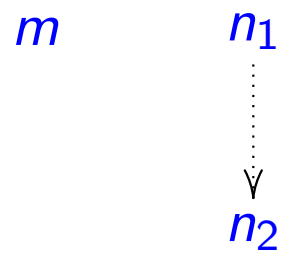


Games

A sequent

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

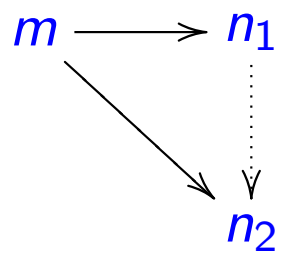
will be interpreted by the game (= polarized poset)



A proof

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{P(x) \vdash P(x) \wedge P(x)}}{P(x) \vdash \exists z.P(x) \wedge P(z)}(\exists)}{P(x) \vdash \exists y.\exists z.P(y) \wedge P(z)}(\exists)}{\exists x.P(x) \vdash \exists y.\exists z.P(y) \wedge P(z)}(\exists)$$

will be interpreted by the strategy



Causal strategies

game A	$=$	partial order on the moves
strategy σ	$=$	relation on moves

Causal strategies

$$\begin{aligned} \textbf{game } A &= \text{partial order on the moves} \\ \textbf{strategy } \sigma &= \text{relation on moves} \end{aligned}$$

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

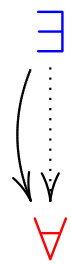
Causal strategies

game A $=$ partial order on the moves
strategy σ $=$ relation on moves

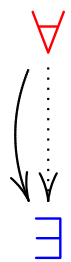
A strategy $\sigma : A$ is **causal** when

1. if $m \xrightarrow{\sigma} n$ then m Opponent and n Player
2. the relation $\leq_A \cup \sigma$ is acyclic

Forbids:



Admits:



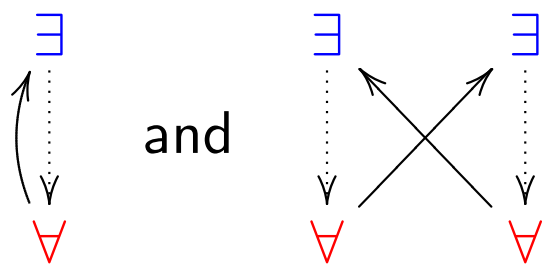
Causal strategies

$$\begin{aligned} \textbf{game } A &= \text{partial order on the moves} \\ \textbf{strategy } \sigma &= \text{relation on moves} \end{aligned}$$

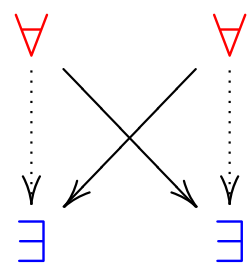
A strategy $\sigma : A$ is **causal** when

1. if $m \xrightarrow{\sigma} n$ then m Opponent and n Player
2. the relation $\leq_A \cup \sigma$ is acyclic

Forbids:



Allows:



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

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

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

Illustration: the simplicial category

The **simplicial category** Δ 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** Δ has

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

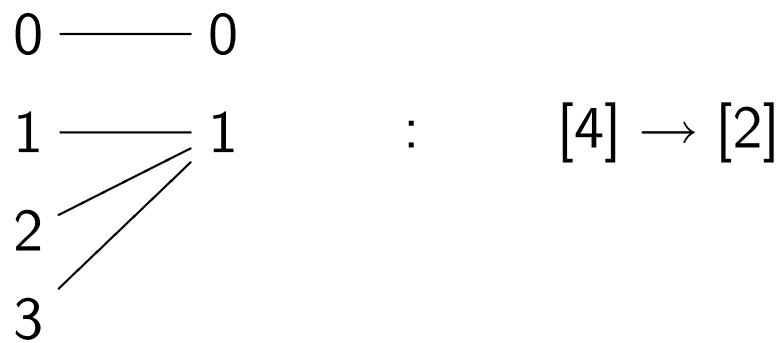


Illustration: the simplicial category

The **simplicial category** Δ 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 ($\circ$)

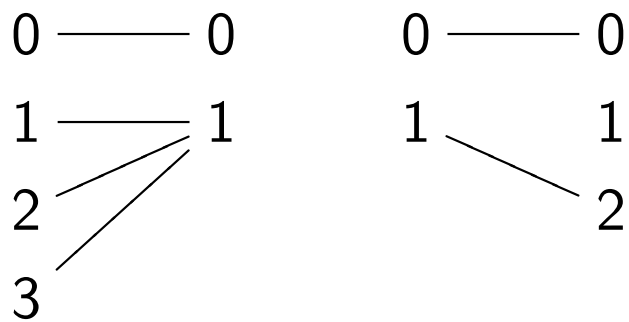


Illustration: the simplicial category

The **simplicial category** Δ 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 ($\circ$)

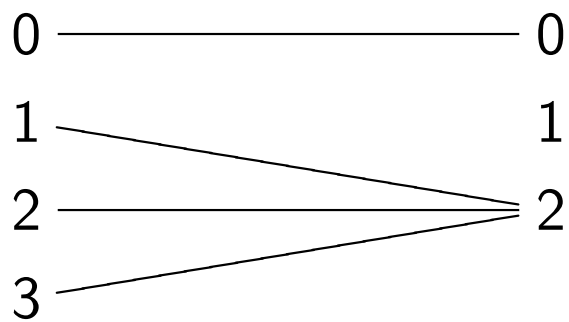


Illustration: the simplicial category

The **simplicial category** Δ 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 ($\otimes$)

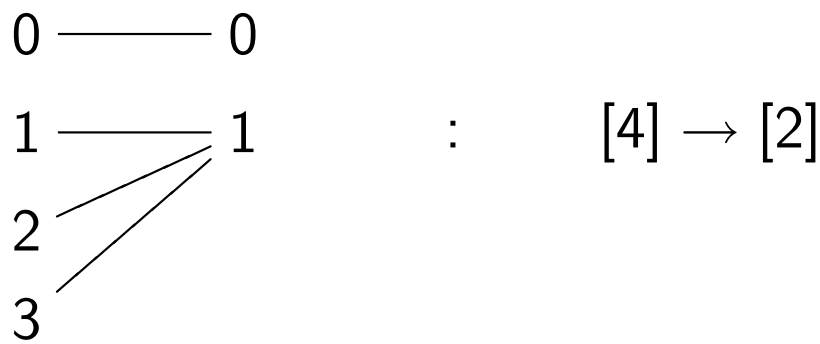


Illustration: the simplicial category

The **simplicial category** Δ 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 ($\otimes$)

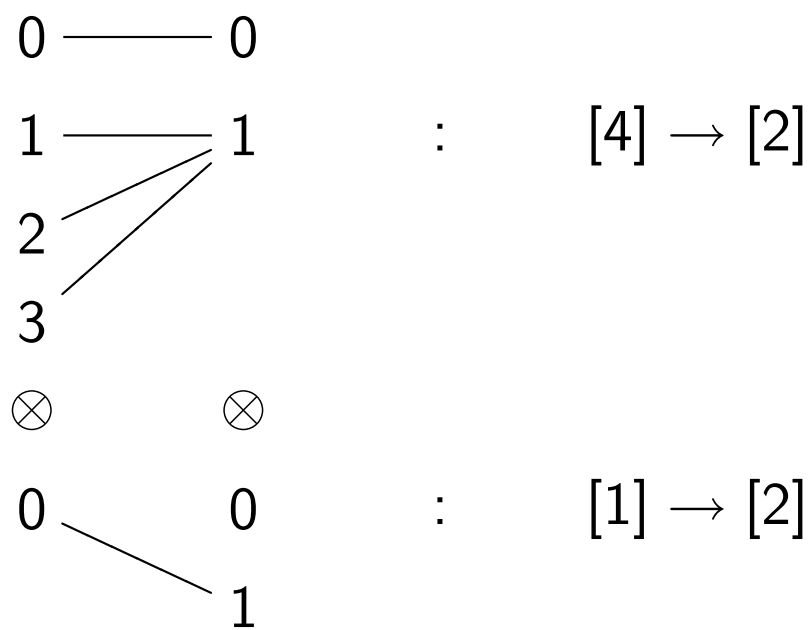
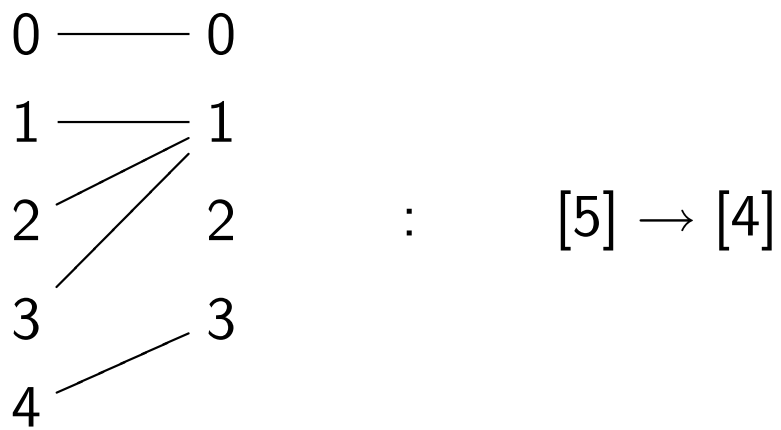


Illustration: the simplicial category

The **simplicial category** Δ has

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

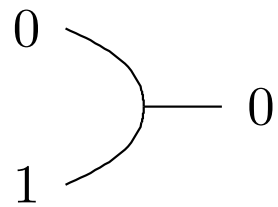
This category is monoidal: vertical composition ($\otimes$)

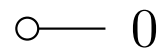


A theory of monoids

The category Δ contains two morphisms:

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

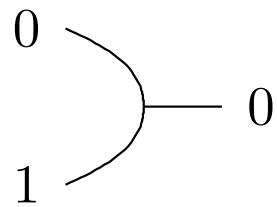


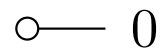


A theory of monoids

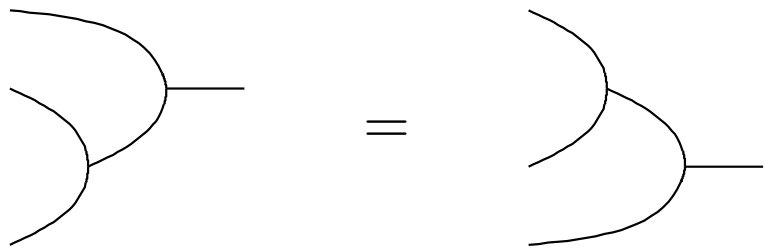
The category Δ contains two morphisms:

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





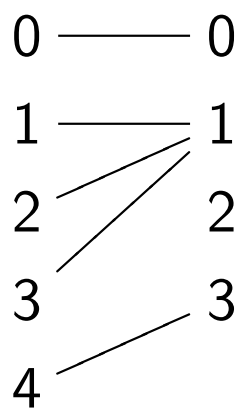
which satisfy



and

Property

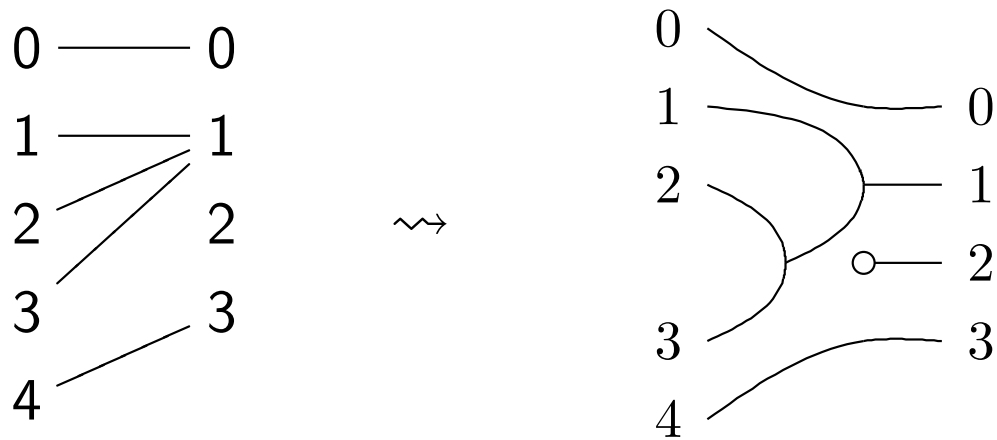
The morphisms μ and η generate Δ.



A theory of monoids

Property

The morphisms μ and η generate Δ.

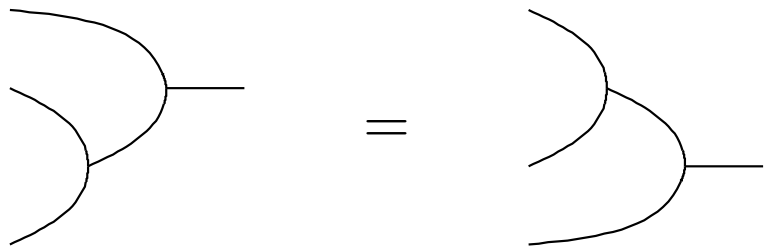


A presentation of the category Δ

The category Δ is isomorphic to the free monoidal category on the two generators

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

quotiented by the relations



and

A theory of monoids

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

$=$

monoid in $\mathcal{C}$

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

A theory of games

strict monoidal functor **Games** $\longrightarrow \mathcal{C}$

$=$

????? in $\mathcal{C}$

The corresponding theory is a polarized variant of relations.

The category **Games**

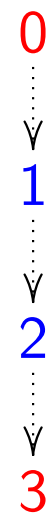
The category **Games** is the category whose

- objects are integers

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

  equipped with a polarization function

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



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

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

$\forall$

$\exists$

$\exists$

$\forall$

The category **Games**

The category **Games** is the category whose

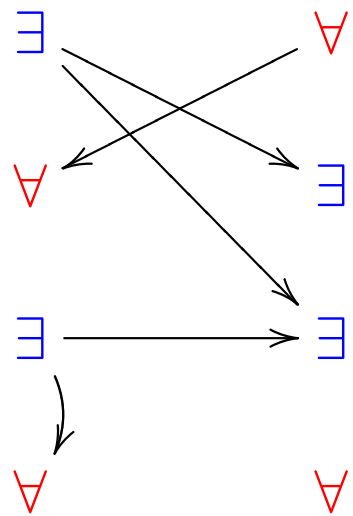
- objects are integers

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

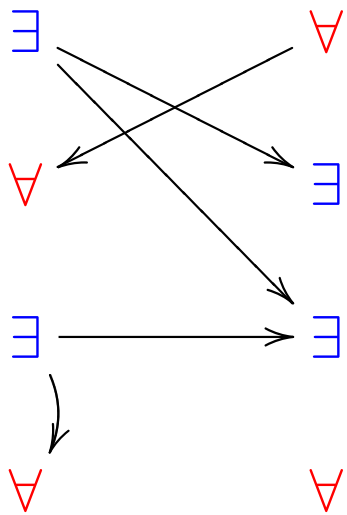
  equipped with a polarization function

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

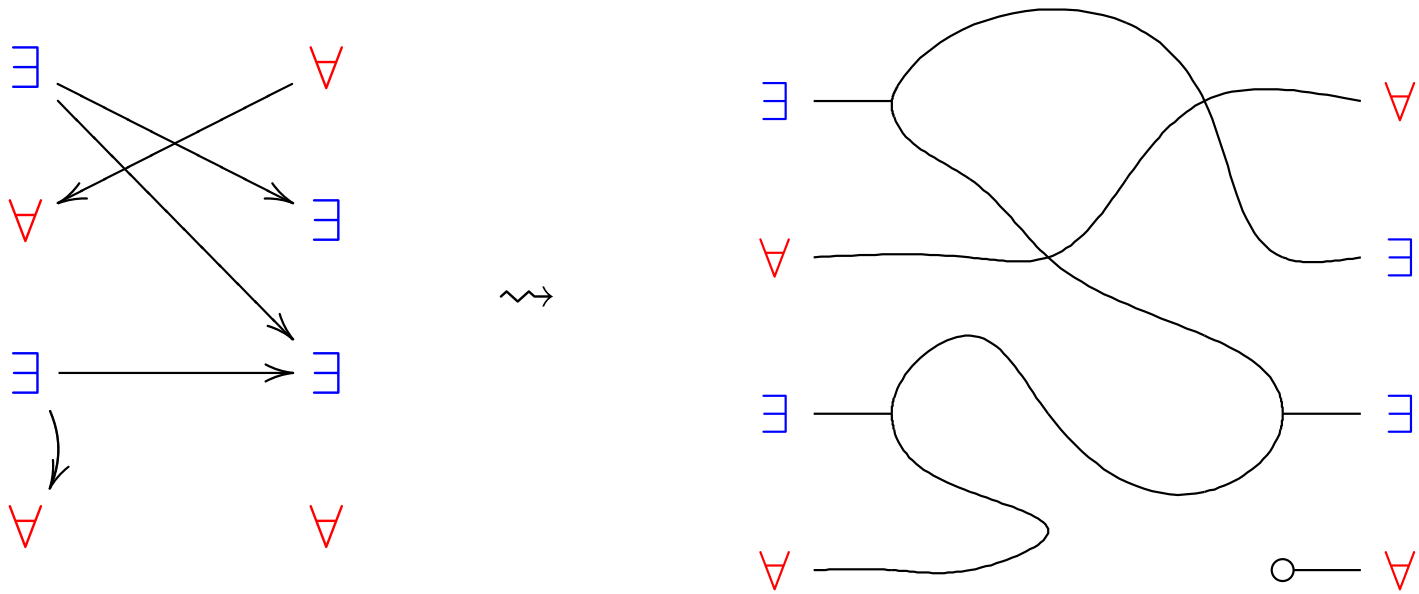
- morphisms are causal strategies.



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**

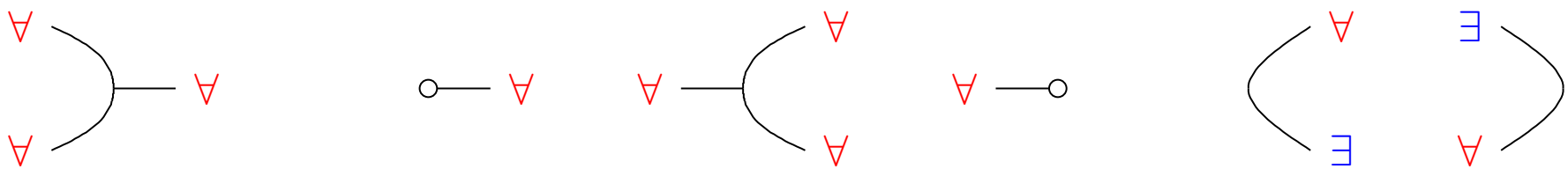
The category **Games** is presented by the polygraph with

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

Presentation of the category **Games**

The category **Games** is presented by the polygraph with

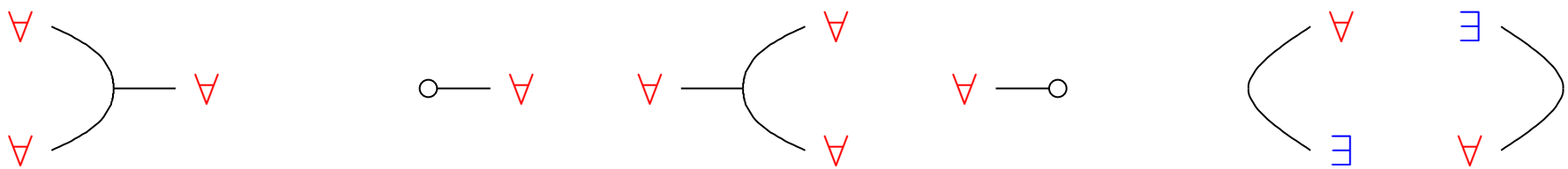
- one 0-cell,
- two 1-cells $\forall$ and $\exists$,
- six 2-cells



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

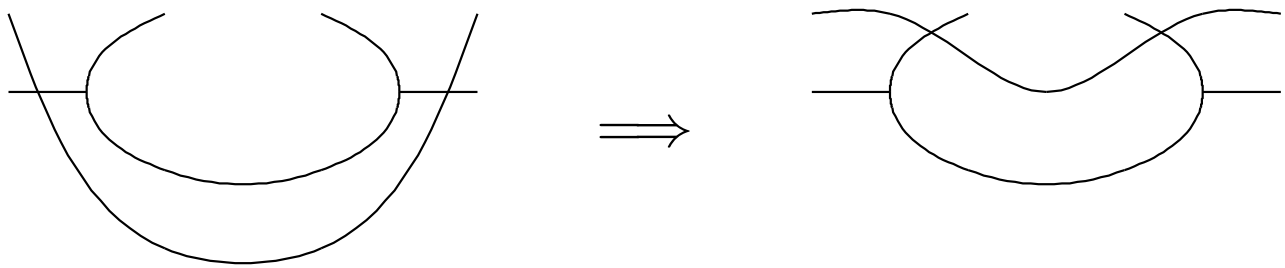
- The proof is done by showing that every diagram is in relation with a diagram in **canonical form** and that these canonical forms are in bijection with the morphisms of this category.

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**.

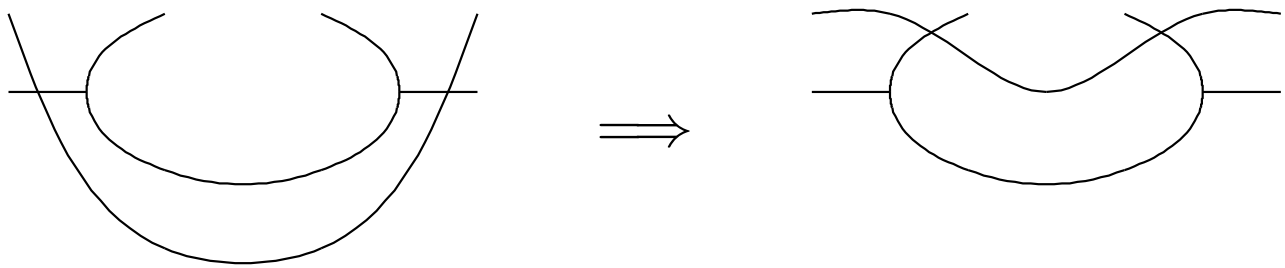
2-dimensional rewriting

- 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 **Mat**($\mathbb{N}$) into a **confluent rewriting system** ($\S 5.5$)



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 **Mat**($\mathbb{N}$) into a **confluent rewriting system** ($\S 5.5$)



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

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!