A Logical Investigation of Interaction Systems

Une investigation logique des systèmes d'interaction

Pierre Hyvernat

Insitut mathématique de Luminy
Université Aix-Marseille 2

PhD Defense, December 12th 2005

Supervisors:
Thierry Coquand (Chalmers Universitet, Göteborg, Sweden)
Thomas Ehrhard (Institut mathématique de Luminy, Marseille, France)

What is this about?

Programs and proofs are complex things!

What is this about?

Programs and proofs are complex things!

We want to give a simpler "denotation" and proofs or programs.

(We can thus forget about syntactical details like the choice of programming language...)

What is this about?

Programs and proofs are complex things!

We want to give a simpler "denotation" and proofs or programs.

(We can thus forget about syntactical details like the choice of programming language...)

We use a notion of abstract games...

What is this about?

Programs and proofs are complex things!

We want to give a simpler "denotation" and proofs or programs.

(We can thus forget about syntactical details like the choice of programming language...)

We use a notion of abstract games...

Games also give a computational interpretation to "topology"...

Where are we?

Games

"Interaction Systems"

We are interested in games between players with full information.

(Like chess or Go and unlike soccer or Poker)

Games

"Interaction Systems"

We are interested in games between players with full information.

(Like chess or Go and unlike soccer or Poker)

- the first player is called the Angel;

Games

"Interaction Systems"

We are interested in games between players with full information.

(Like chess or Go and unlike soccer or Poker)

- the first player is called the Angel;
- the second is called the Demon;

Games

"Interaction Systems"

We are interested in games between players with full information.

(Like chess or Go and unlike soccer or Poker)

- the first player is called the Angel;
- the second is called the Demon;
- moves alternate: first the Angel, then the Demon etc.

"Simulations"

Some games are equivalent:

"Simulations"

Some games are equivalent:

all moves in the first games can be translated
into moves in the second game; and vice and versa.

"Simulations"

Some games are equivalent:

all moves in the first games can be translated
into moves in the second game; and vice and versa.

More generally, we say that a game G_1 is easier than a game G_2 if:

"Simulations"

Some games are equivalent:

all moves in the first games can be translated
into moves in the second game; and vice and versa.

More generally, we say that a game G_1 is easier than a game G_2 if:

- Angel moves in G_1 can be translated into Angels moves in G_2;

"Simulations"

Some games are equivalent:

all moves in the first games can be translated
into moves in the second game; and vice and versa.

More generally, we say that a game G_1 is easier than a game G_2 if:

- Angel moves in G_1 can be translated into Angels moves in G_2;
- Demon moves in G_2 can be translated into moves in G_1.

"Simulations"

Some games are equivalent:

all moves in the first games can be translated
into moves in the second game; and vice and versa.

More generally, we say that a game G_1 is easier than a game G_2 if:

- Angel moves in G_1 can be translated into Angels moves in G_2;
- Demon moves in G_2 can be translated into moves in G_1.

Thus G_1 is easier for the Angel but more difficult for the Demon.

"Simulations"

Some games are equivalent:

all moves in the first games can be translated
into moves in the second game; and vice and versa.

More generally, we say that a game G_1 is easier than a game G_2 if:

- Angel moves in G_1 can be translated into Angels moves in G_2;
- Demon moves in G_2 can be translated into moves in G_1.

Thus G_1 is easier for the Angel but more difficult for the Demon.

We write $G_1 \leq G_2$ and say "G_2 simulates G_1".

Examples of games

- The game of Chess;

Examples of games

- The game of Chess;
- game of "*Devinettes*":
  - the Angel asks the questions,
  - the Demon answers by YES or NO;

Examples of games

- The game of Chess;
- game of "*Devinettes*":
  - the Angel asks the questions,
  - the Demon answers by YES or NO;
- potential executions of a program:
  - the Angel is the user,
  - the Demon is the computer;

Examples of games

- The game of Chess;
- game of "*Devinettes*":
  - the Angel asks the questions,
  - the Demon answers by YES or NO;
- potential executions of a program:
  - the Angel is the user,
  - the Demon is the computer;
- a specification for a sequential / interactive program.

This was the starting intuition... (*cf.* Peter Hancock)

Safety Properties

The denotation of a program/proof will be a safety property...

Safety Properties

The denotation of a program/proof will be a safety property...

... *i.e.* a set of "winning positions" for the Angel

Safety Properties

The denotation of a program/proof will be a safety property...

... *i.e.* a set of "winning positions" for the Angel:

from each winning position,

Safety Properties

The denotation of a program/proof will be a safety property...

... *i.e.* a set of "winning positions" for the Angel:

from each winning position,
the Angel can find a smart move

Safety Properties

The denotation of a program/proof will be a safety property...

... *i.e.* a set of "winning positions" for the Angel:

from each winning position,
the Angel can find a smart move
to always remain in a winning position

Safety Properties

The denotation of a program/proof will be a safety property...

... *i.e.* a set of "winning positions" for the Angel:

from each winning position,
the Angel can find a smart move
to always remain in a winning position
no matter what the Demon plays...

Safety Properties

The denotation of a program/proof will be a safety property...

... *i.e.* a set of "winning positions" for the Angel:

from each winning position,
the Angel can find a smart move
to always remain in a winning position
no matter what the Demon plays...

(In particular, the Angel always has a move to play!)

Where are we?

Objects: Interaction Systems

Definition

An interaction system w is given by the following:

Objects: Interaction Systems

Definition

An interaction system w is given by the following:

- a set S of states;

Objects: Interaction Systems

Definition

An interaction system w is given by the following:

- a set S of states;
- for each state $s \in S$, a set $A(s)$ of actions;

Objects: Interaction Systems

Definition

An interaction system w is given by the following:

- a set S of states;
- for each state $s \in S$, a set $A(s)$ of actions;
- for each action $a \in A(s)$, a set $D(s, a)$ of reactions;

Objects: Interaction Systems

Definition

An interaction system w is given by the following:

- a set S of states;
- for each state $s \in S$, a set $A(s)$ of actions;
- for each action $a \in A(s)$, a set $D(s, a)$ of reactions;
- for each reaction $d \in D(s, a)$, a new state $n(s,a,d) \in S$.

Objects: Interaction Systems

Definition

An interaction system w is given by the following:

- a set S of states;
- for each state $s \in S$, a set $A(s)$ of actions;
- for each action $a \in A(s)$, a set $D(s,a)$ of reactions;
- for each reaction $d \in D(s,a)$, a new state $n(s,a,d) \in S$.

(Equivalently, an interaction system is a coalgebra for the monad $\mathcal{F}^2$ of "doubly iterated families" over the category **Set**.)

Morphisms: Simulations

Definition

If w_1 and w_2 are interaction systems,
a relation $R \subseteq S_1 \times S_2$ is a simulation from w_1 to w_2 iff

Morphisms: Simulations

Definition

If w_1 and w_2 are interaction systems,
a relation $R \subseteq S_1 \times S_2$ is a simulation from w_1 to w_2 iff

$$(s_1, s_2) \in R \Rightarrow \begin{array}{l} \forall a_1 \in A_1(s_1) \\ \exists a_2 \in A_2(s_2) \end{array}$$

$$\big(n_1(s_1, a_1 \quad), n_2(s_2, a_2 \quad)\big) \in R$$

Morphisms: Simulations

Definition

If w_1 and w_2 are interaction systems,
a relation $R \subseteq S_1 \times S_2$ is a simulation from w_1 to w_2 iff

$$
\begin{aligned}
(s_1, s_2) \in R \Rightarrow \quad & \forall a_1 \in A_1(s_1) \\
& \exists a_2 \in A_2(s_2) \\
& \forall d_2 \in D_2(s_2, a_2) \\
& \exists d_1 \in D_1(s_1, a_1) \\
& \qquad \big(n_1(s_1, a_1, d_1), n_2(s_2, a_2, d_2)\big) \in R
\end{aligned}
$$

Morphisms: Simulations

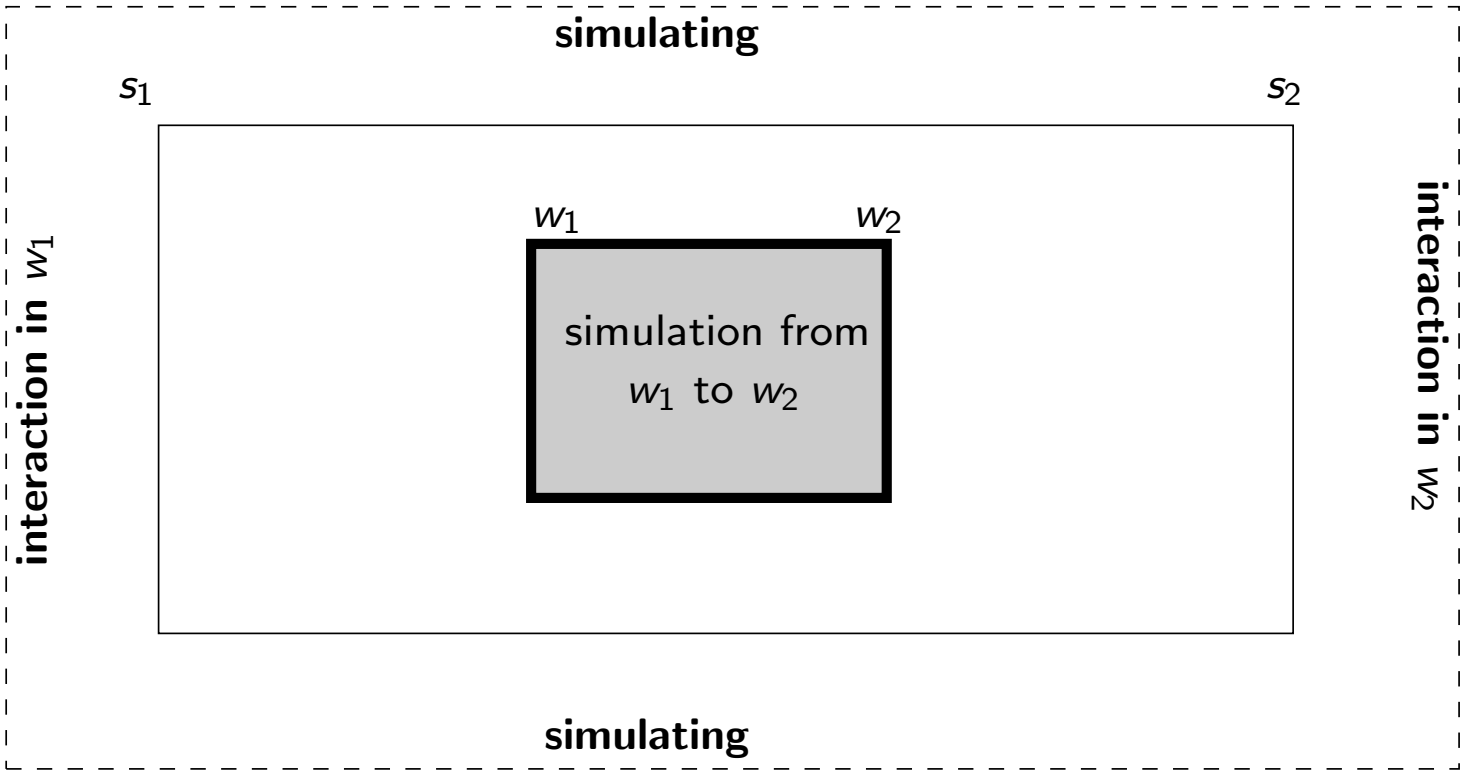
Definition

If w_1 and w_2 are interaction systems,
a relation $R \subseteq S_1 \times S_2$ is a simulation from w_1 to w_2 iff

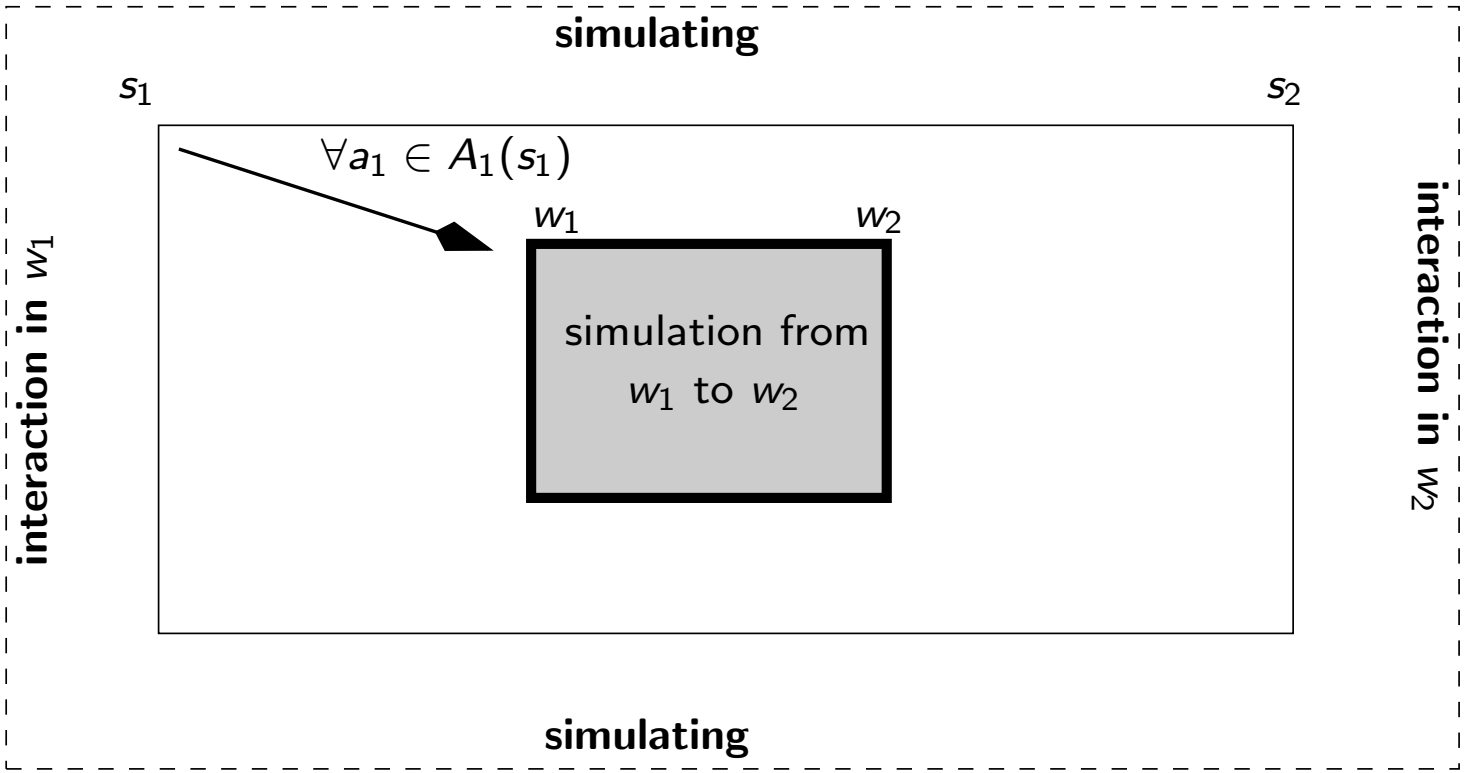
$$
\begin{aligned}
(s_1, s_2) \in R \Rightarrow \quad & \forall a_1 \in A_1(s_1) \\
& \exists a_2 \in A_2(s_2) \\
& \forall d_2 \in D_2(s_2, a_2) \\
& \exists d_1 \in D_1(s_1, a_1) \\
& \qquad \big(n_1(s_1, a_1, d_1), n_2(s_2, a_2, d_2)\big) \in R
\end{aligned}
$$

(This is **not** a morphism of coalgebras...)

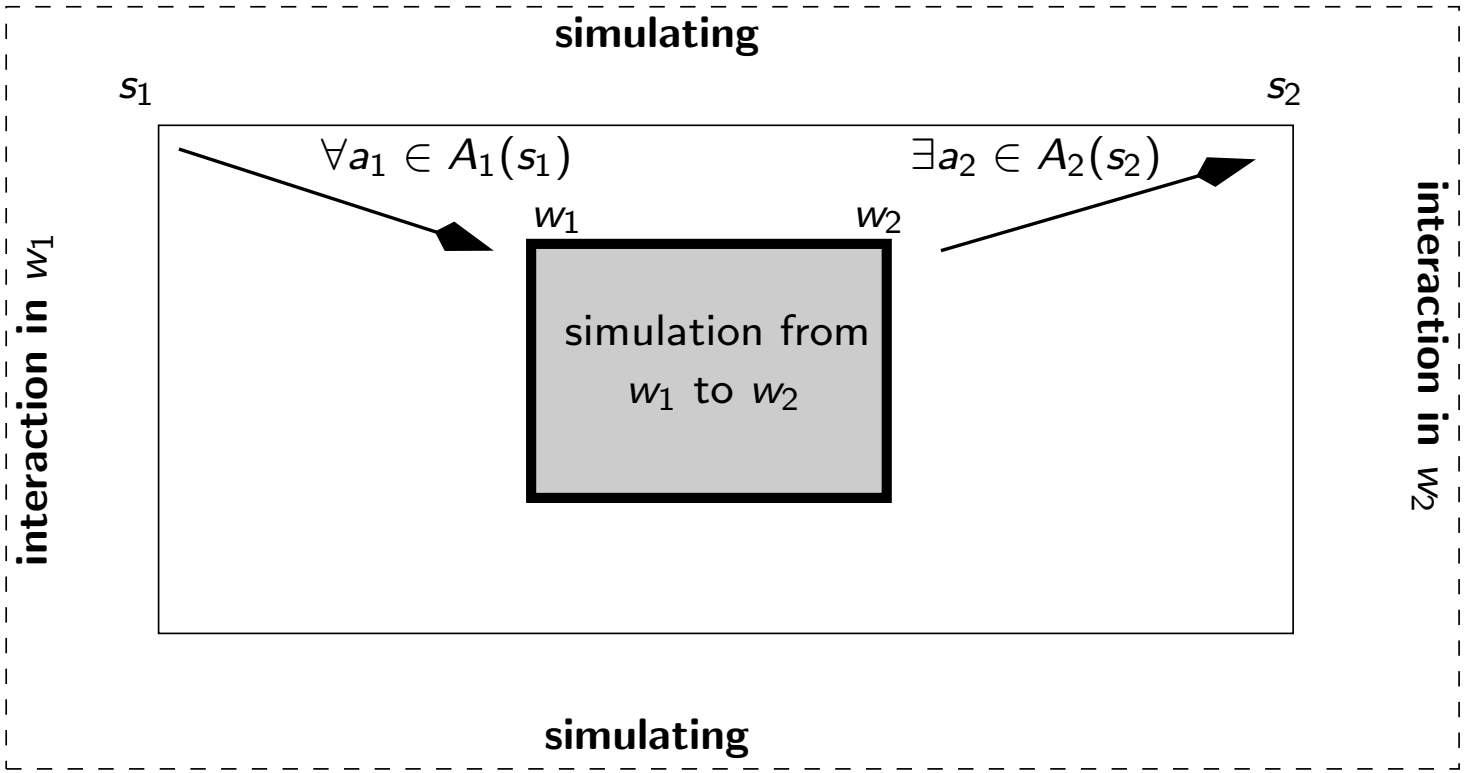
Simulation, Visually



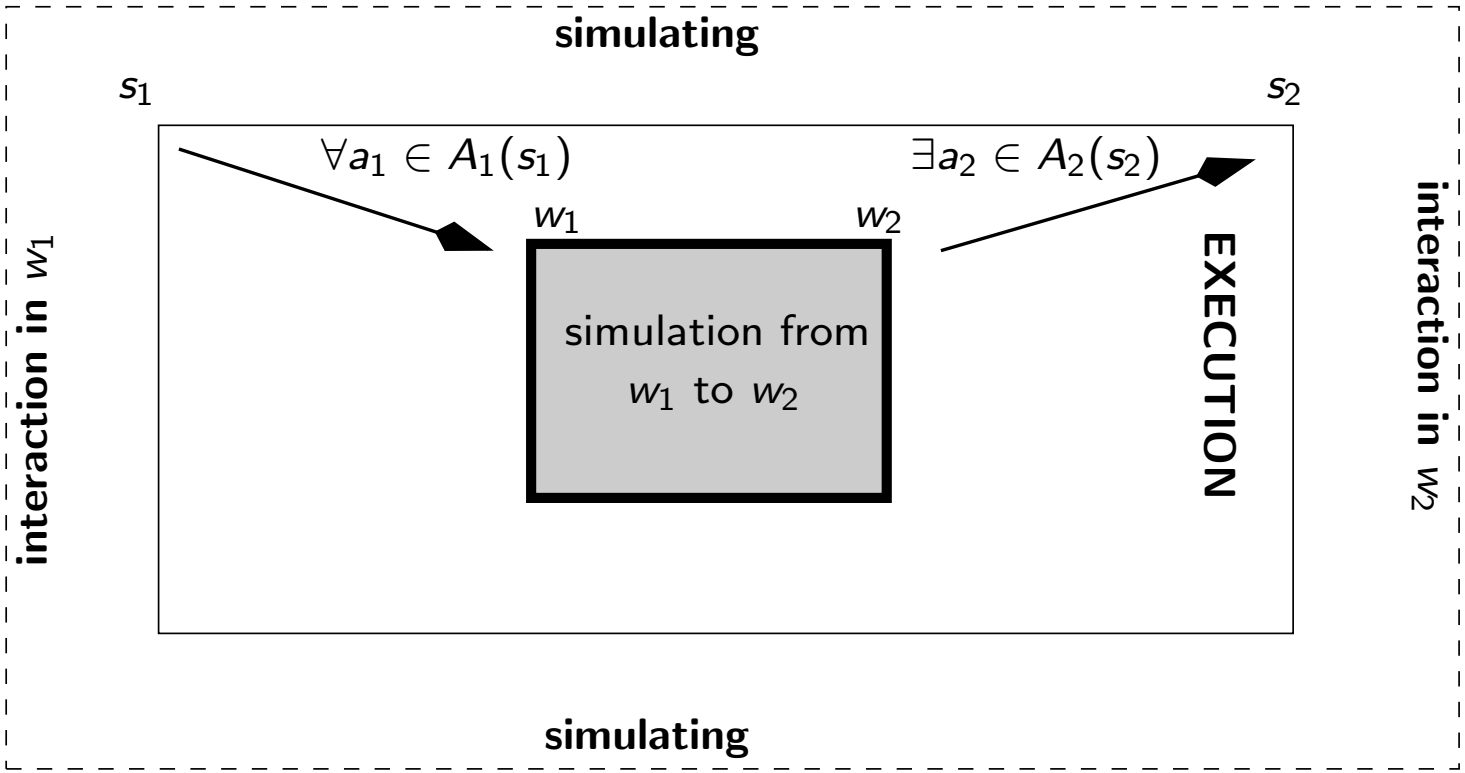
Simulation, Visually



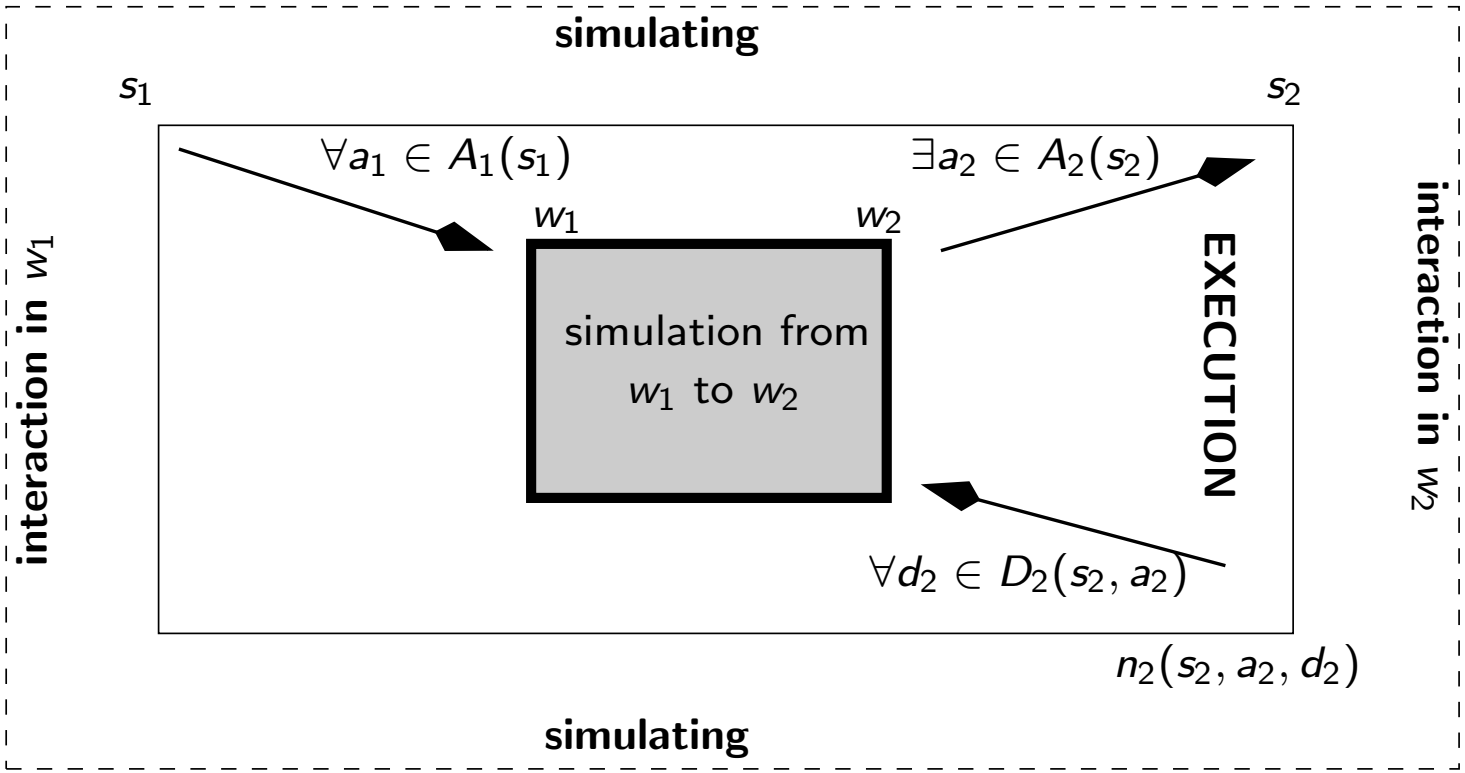
Simulation, Visually



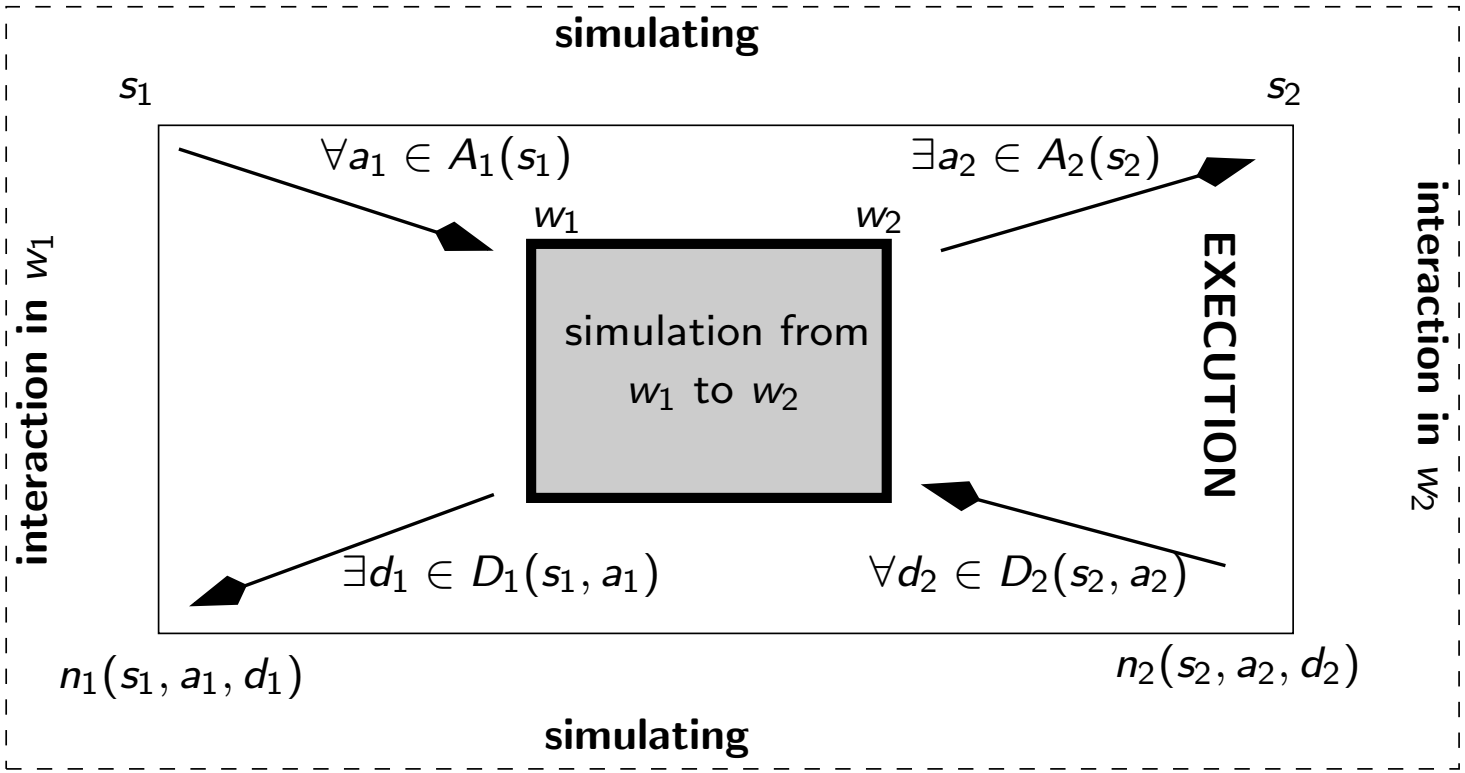
Simulation, Visually



Simulation, Visually



Simulation, Visually

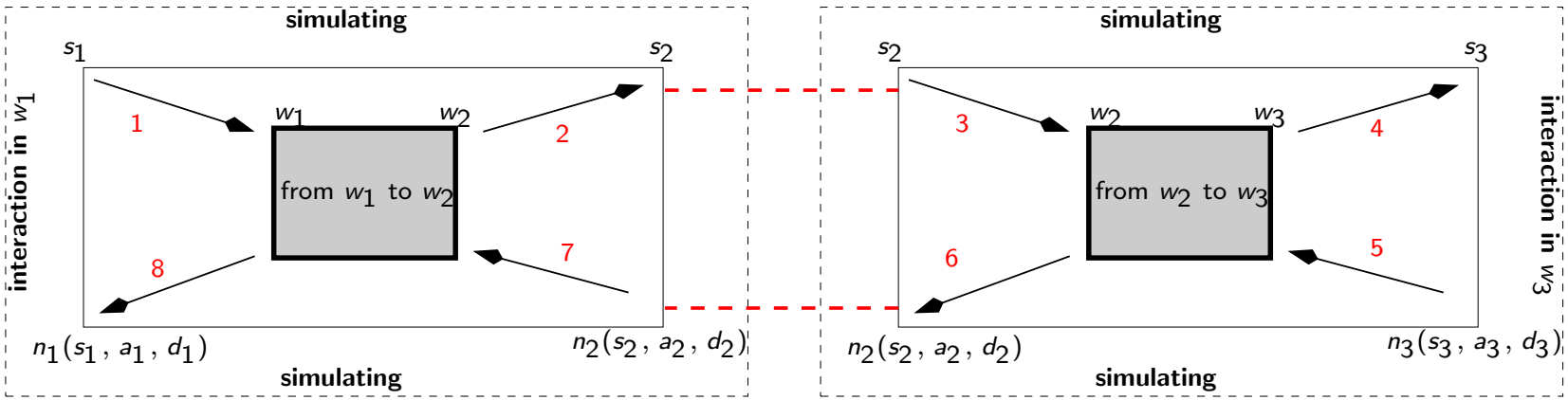


Composition

To compose two simulations from w_1 to w_2 and from w_2 to w_3...
...use the relational composition:

Composition

To compose two simulations from w_1 to w_2 and from w_2 to w_3...
...use the relational composition:



(flow of interaction)

Reflexive and Transitive Closure

Definition

There is a functorial operation $w \mapsto w^*$

Reflexive and Transitive Closure

Definition

There is a functorial operation $w \mapsto w^*$ s.t.

- an Angel action in w^* is a strategy to play several times in w;

Reflexive and Transitive Closure

Definition

There is a functorial operation $w \mapsto w^*$ s.t.

- an Angel action in w^* is a strategy to play several times in w;
- a Demon reaction is a sequence of responses.

Reflexive and Transitive Closure

Definition

There is a functorial operation $w \mapsto w^*$ s.t.

- an Angel action in w^* is a strategy to play several times in w;
- a Demon reaction is a sequence of responses.

This operation satisfies w^* is "least" s.t. $w^* \simeq \mathsf{skip} \cup w; w^*$,

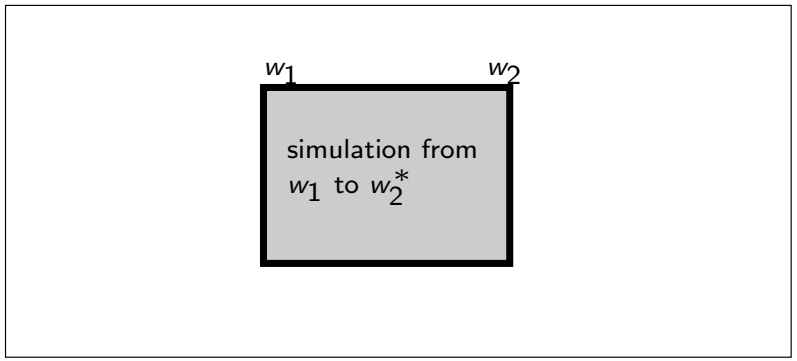
where $w_1; w_2$ is the game

"one move in w_1 and then one move in w_2."

Programming Interpretation

A simulation from w_1 to w_2^* is a relation R s.t. if $(s_1, s_2) \in R$:

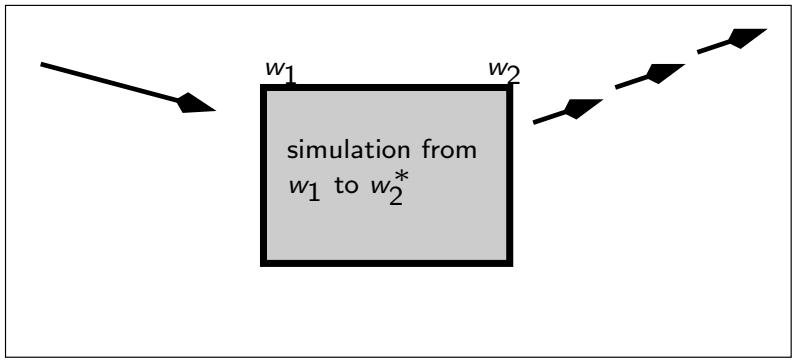
- for every action from s_1, there is a "list" of actions from s_2;
- s.t. for any "list" of reactions, there is a reaction;
- s.t. the simulation can be sustained from the new states.



Programming Interpretation

A simulation from w_1 to w_2^* is a relation R s.t. if $(s_1, s_2) \in R$:

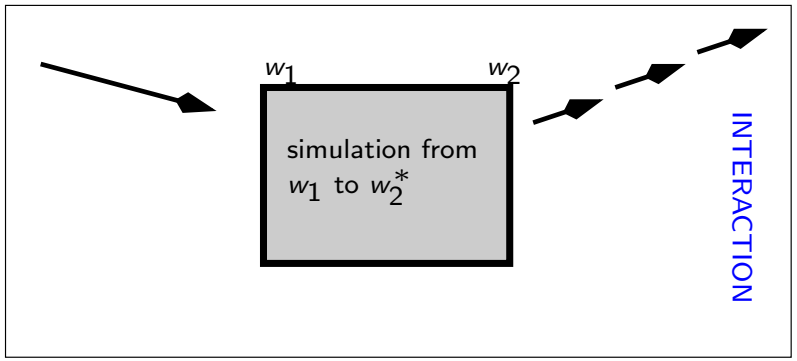
- for every action from s_1, there is a "list" of actions from s_2;
- s.t. for any "list" of reactions, there is a reaction;
- s.t. the simulation can be sustained from the new states.



Programming Interpretation

A simulation from w_1 to w_2^* is a relation R s.t. if $(s_1, s_2) \in R$:

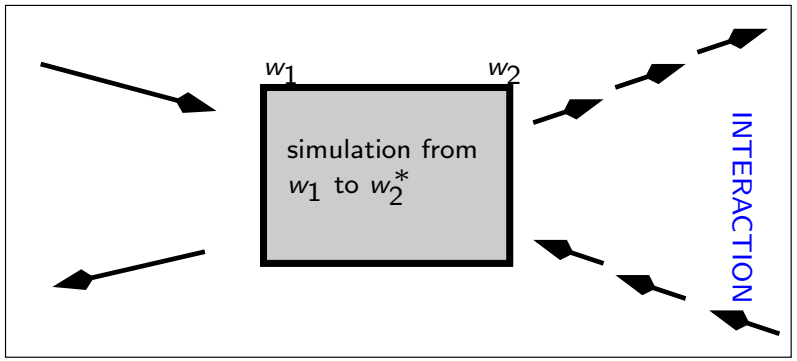
- for every action from s_1, there is a "list" of actions from s_2;
- s.t. for any "list" of reactions, there is a reaction;
- s.t. the simulation can be sustained from the new states.



Programming Interpretation

A simulation from w_1 to w_2^* is a relation R s.t. if $(s_1, s_2) \in R$:

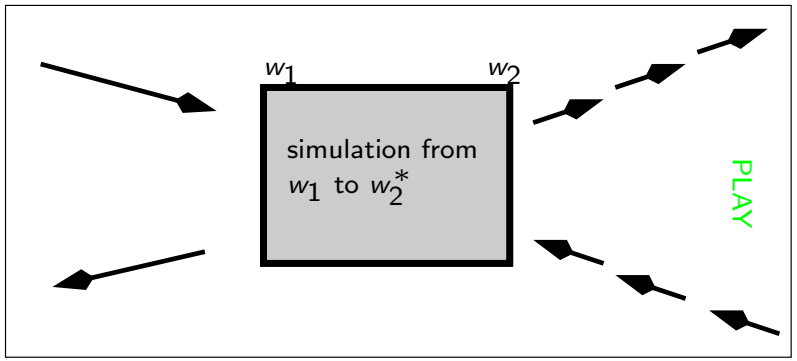
- for every action from s_1, there is a "list" of actions from s_2;
- s.t. for any "list" of reactions, there is a reaction;
- s.t. the simulation can be sustained from the new states.



Programming Interpretation

A simulation from w_1 to w_2^* is a relation R s.t. if $(s_1, s_2) \in R$:

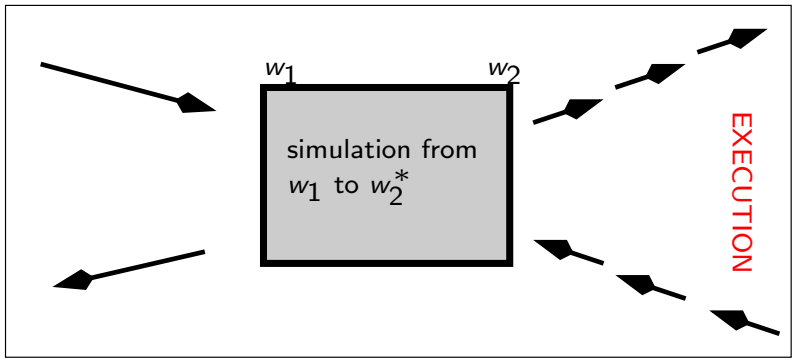
- for every move from s_1, there is a strategy from s_2;
- s.t. for any counter-strategy, there is a counter-move;
- s.t. the simulation can be sustained from the new states.



Programming Interpretation

A simulation from w_1 to w_2^* is a relation R s.t. if $(s_1, s_2) \in R$:

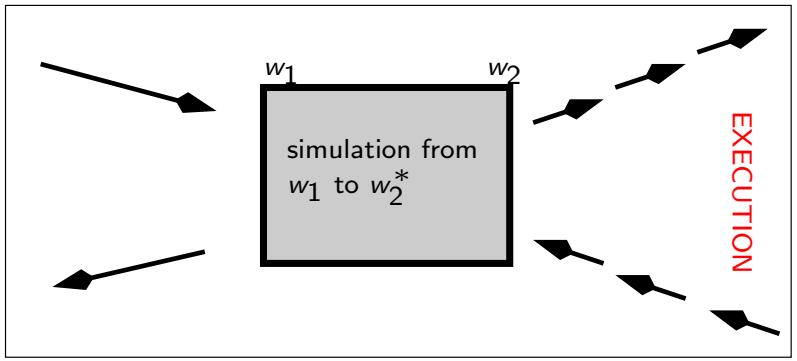
- for every command from s_1, there is a program from s_2;
- s.t. for any sequence of responses, there is a response;
- s.t. the simulation can be sustained from the new states.



Programming Interpretation

A simulation from w_1 to w_2^* is a relation R s.t. if $(s_1, s_2) \in R$:

- for every command from s_1, there is a program from s_2;
- s.t. for any sequence of responses, there is a response;
- s.t. the simulation can be sustained from the new states.



This is just a program implementing w_1 in terms of w_2!

Topology Interpretation

- S is a basis for a topological space;

Topology Interpretation

- S is a basis for a topological space;
- $A(s)$ corresponds to the atomic covering of the basic open s;

Topology Interpretation

- S is a basis for a topological space;
- $A(s)$ corresponds to the atomic covering of the basic open s;
- $D(s, a)$ indexes the basic opens from the covering a;

Topology Interpretation

- S is a basis for a topological space;
- $A(s)$ corresponds to the atomic covering of the basic open s;
- $D(s, a)$ indexes the basic opens from the covering a;
- $n(s, a, d)$ is the basic open corresponding to index d.

(This bears similarities with Grothendieck topologies.)

Topology Interpretation

- S is a basis for a topological space;
- $A(s)$ corresponds to the atomic covering of the basic open s;
- $D(s, a)$ indexes the basic opens from the covering a;
- $n(s, a, d)$ is the basic open corresponding to index d.

(This bears similarities with Grothendieck topologies.)

Theorem

There is a full and faithful functor from $\mathbf{Ref}^{op}$ *to* **BTop**.

Topology Interpretation

- S is a basis for a topological space;
- $A(s)$ corresponds to the atomic covering of the basic open s;
- $D(s, a)$ indexes the basic opens from the covering a;
- $n(s, a, d)$ is the basic open corresponding to index d.

(This bears similarities with Grothendieck topologies.)

Theorem

There is a full and faithful functor from $\mathbf{Ref}^{op}$ *to* $\mathbf{BTop}$.

i.e. a simulation from w_1 to w_2^* ...

Topology Interpretation

- S is a basis for a topological space;
- $A(s)$ corresponds to the atomic covering of the basic open s;
- $D(s, a)$ indexes the basic opens from the covering a;
- $n(s, a, d)$ is the basic open corresponding to index d.

(This bears similarities with Grothendieck topologies.)

Theorem

There is a full and faithful functor from $\mathbf{Ref}^{op}$ *to* **BTop**.

i.e. a simulation from w_1 to w_2^* ...
... is exactly a continuous function from w_2 to w_1.

Simplifying the Presentation

Advantages of interaction systems:

- very concrete (*cf.* link with programming);

Simplifying the Presentation

Advantages of interaction systems:

- very concrete (*cf.* link with programming);
- simple computational content;

Simplifying the Presentation

Advantages of interaction systems:

- very concrete (*cf.* link with programming);
- simple computational content;
- adequate to model "predicative" topology.

Simplifying the Presentation

Advantages of interaction systems:

- very concrete (*cf.* link with programming);
- simple computational content;
- adequate to model "predicative" topology.

Drawbacks of interaction systems:

- very concrete;

Simplifying the Presentation

Advantages of interaction systems:

- very concrete (*cf.* link with programming);
- simple computational content;
- adequate to model "predicative" topology.

Drawbacks of interaction systems:

- very concrete;
- too simple (?!) computational content;

Simplifying the Presentation

Advantages of interaction systems:

- very concrete (*cf.* link with programming);
- simple computational content;
- adequate to model "predicative" topology.

Drawbacks of interaction systems:

- very concrete;
- too simple (?!) computational content;
- some simple operations look complicated.

Predicate Transformers

In a classical setting, we simplify the structure:

Predicate Transformers

In a classical setting, we simplify the structure:
to any interaction system w we associate

$$\begin{array}{rcccl} w^\circ & : & \mathcal{P}(S) & \to & \mathcal{P}(S) \\ & & U & \mapsto & \left\{ s \mid (\exists a)(\forall d)\, n(s,a,d) \in U \right\} \end{array}$$

i.e. $s \in w^\circ(U)$ iff "the Angel can reach U from s in exactly one interaction".

Predicate Transformers

In a classical setting, we simplify the structure:
to any interaction system w we associate

$$\begin{array}{lcccl} w^\circ & : & \mathcal{P}(S) & \to & \mathcal{P}(S) \\ & & U & \mapsto & \{s \mid (\exists a)(\forall d)\, n(s,a,d) \in U\} \end{array}$$

Theorem

We have that R is a simulation from w_1 to w_2 iff

$$R \cdot w_1^\circ \quad \subseteq \quad w_2^\circ \cdot R$$

Predicate Transformers

In a classical setting, we simplify the structure:
to any interaction system w we associate

$$\begin{array}{rcccl} w^\circ & : & \mathcal{P}(S) & \to & \mathcal{P}(S) \\ & & U & \mapsto & \{s \mid (\exists a)(\forall d)\, n(s,a,d) \in U\} \end{array}$$

Theorem

We have that R is a simulation from w_1 to w_2 iff

$$R \cdot w_1^\circ \quad \subseteq \quad w_2^\circ \cdot R$$

This defines an equivalence of categories **PT** $\simeq$ **Sim**!

Monoidal Structure

For $P_1 : \mathcal{P}(S_1) \to \mathcal{P}(S_1)$ and $P_2 : \mathcal{P}(S_2) \to \mathcal{P}(S_2)$:

Monoidal Structure

For $P_1 : \mathcal{P}(S_1) \to \mathcal{P}(S_1)$ and $P_2 : \mathcal{P}(S_2) \to \mathcal{P}(S_2)$:

$P_1^\perp : \mathcal{P}(S_1) \to \mathcal{P}(S_1)$

(In particular involutivity of $\_^\perp$ is trivial.)

Monoidal Structure

For $P_1 : \mathcal{P}(S_1) \to \mathcal{P}(S_1)$ and $P_2 : \mathcal{P}(S_2) \to \mathcal{P}(S_2)$:

$$P_1^\perp : \mathcal{P}(S_1) \to \mathcal{P}(S_1)$$
$$x \mapsto \overline{P_1(\overline{x})}$$

(In particular involutivity of $\_^\perp$ is trivial.)

Monoidal Structure

For $P_1 : \mathcal{P}(S_1) \to \mathcal{P}(S_1)$ and $P_2 : \mathcal{P}(S_2) \to \mathcal{P}(S_2)$:

$P_1^\perp : \mathcal{P}(S_1) \to \mathcal{P}(S_1)$
$x \mapsto \overline{P_1(\overline{x})}$

$P_1 \otimes P_2 : \mathcal{P}(S_1 \times S_2) \to \mathcal{P}(S_1 \times S_2)$

(In particular involutivity of $\_^\perp$ is trivial.)

Monoidal Structure

For $P_1 : \mathcal{P}(S_1) \to \mathcal{P}(S_1)$ and $P_2 : \mathcal{P}(S_2) \to \mathcal{P}(S_2)$:

$$P_1^\perp : \mathcal{P}(S_1) \to \mathcal{P}(S_1) \qquad x \mapsto \overline{P_1(\overline{x})}$$

$$P_1 \otimes P_2 : \mathcal{P}(S_1 \times S_2) \to \mathcal{P}(S_1 \times S_2) \qquad r \mapsto \bigcup_{x_1 \times x_2 \subseteq r} P_1(x_1) \times P_2(x_2)$$

(In particular involutivity of $\_^\perp$ is trivial.)

Monoidal Structure

For $P_1 : \mathcal{P}(S_1) \to \mathcal{P}(S_1)$ and $P_2 : \mathcal{P}(S_2) \to \mathcal{P}(S_2)$:

$$P_1^\perp : \mathcal{P}(S_1) \to \mathcal{P}(S_1) \qquad x \mapsto \overline{P_1(\overline{x})}$$

$$P_1 \otimes P_2 : \mathcal{P}(S_1 \times S_2) \to \mathcal{P}(S_1 \times S_2) \qquad r \mapsto \bigcup_{x_1 \times x_2 \subseteq r} P_1(x_1) \times P_2(x_2)$$

This gives a self-dual symmetric monoidal category.

(In particular involutivity of $\_^\perp$ is trivial.)

Those correspond to concrete operations on interaction systems...

Monoidal closure

We can extend this to a self-dual monoidal closed category.

Monoidal closure

We can extend this to a self-dual monoidal closed category.

The adjoint to $\otimes$ is given by

$$P_1 \multimap P_2 : \mathcal{P}(S_1 \times S_2) \to \mathcal{P}(S_1 \times S_2)$$

Monoidal closure

We can extend this to a self-dual monoidal closed category.

The adjoint to $\otimes$ is given by

$$P_1 \multimap P_2 : \mathcal{P}(S_1 \times S_2) \to \mathcal{P}(S_1 \times S_2)$$

with

$$(s_1, s_2) \in \big(P_1 \multimap P_2\big)(r)$$
$$\text{iff}$$
$$(\forall x_1 \subseteq S_1)\, s_1 \in P_1(x_1) \Rightarrow s_2 \in P_2\big(r(x_1)\big)$$

Linear Logic

With an appropriate construction

$$!P : \mathcal{P}\big(\mathcal{M}_f(S)\big) \to \mathcal{P}\big(\mathcal{M}_f(S)\big)$$

Linear Logic

With an appropriate construction

$$!P : \mathcal{P}\big(\mathcal{M}_f(S)\big) \to \mathcal{P}\big(\mathcal{M}_f(S)\big)$$

we can interpret all of linear logic or typed λ-calculus.

(This corresponds to the construction of the free $\otimes$-comonoid...)

Linear Logic

With an appropriate construction

$$!P : \mathcal{P}\big(\mathcal{M}_f(S)\big) \to \mathcal{P}\big(\mathcal{M}_f(S)\big)$$

we can interpret all of linear logic or typed λ-calculus.

(This corresponds to the construction of the free $\otimes$-comonoid...)

A proof/term becomes a safety property,

i.e. a subset $x \subseteq S$ s.t. $x \subseteq P(x)$.

Differential λ-calculus

Differential λ-calculus has an intrinsic notion of

- non-determinism (addition);
- linear substitution (differentiation).

Differential λ-calculus

Differential λ-calculus has an intrinsic notion of

- non-determinism (addition);
- linear substitution (differentiation).

Traditional models do not model those new features!

Differential λ-calculus

Differential λ-calculus has an intrinsic notion of

- non-determinism (addition);
- linear substitution (differentiation).

Traditional models do not model those new features!

Safety properties are closed under arbitrary union,

Differential λ-calculus

Differential λ-calculus has an intrinsic notion of

- non-determinism (addition);
- linear substitution (differentiation).

Traditional models do not model those new features!

Safety properties are closed under arbitrary union,
we can thus interpret non-determinism

Differential λ-calculus

Differential λ-calculus has an intrinsic notion of

- non-determinism (addition);
- linear substitution (differentiation).

Traditional models do not model those new features!

Safety properties are closed under arbitrary union,
we can thus interpret non-determinism
and even differentiation.

Differential λ-calculus

Differential λ-calculus has an intrinsic notion of

- non-determinism (addition);
- linear substitution (differentiation).

Traditional models do not model those new features!

Safety properties are closed under arbitrary union,
we can thus interpret non-determinism
and even differentiation.

We get a simple, non-trivial model for the differential λ-calculus!

Where are we?

Achievements

- a new category of games and simulations;
- an intuitive/informal model for "real-life" programming;
- giving a computational interpretation of "basic topologies";
- concrete example of interaction system to give a (complete) topological semantics to "linear geometric theories";
- this category is a denotational model for full linear logic;
- and the differential (typed) λ-calculus;
- which can extended to second order.

Future Work

- link the topology part and the linear logic part;

Future Work

- link the topology part and the linear logic part;
- study the model of differential λ-calculus in more details;

Future Work

- link the topology part and the linear logic part;
- study the model of differential λ-calculus in more details;
- do we have denotational completeness?

Future Work

- link the topology part and the linear logic part;
- study the model of differential λ-calculus in more details;
- do we have denotational completeness?
- Study in particular untyped differential λ-calculus;

Future Work

- link the topology part and the linear logic part;
- study the model of differential λ-calculus in more details;
- do we have denotational completeness?
- Study in particular untyped differential λ-calculus;
- do we get a model of Lionel Vaux's differential $\lambda\mu$-calculus?

Future Work

- link the topology part and the linear logic part;
- study the model of differential λ-calculus in more details;
- do we have denotational completeness?
- Study in particular untyped differential λ-calculus;
- do we get a model of Lionel Vaux's differential $\lambda\mu$-calculus?
- generalize in the spirit of "containers";

Future Work

- link the topology part and the linear logic part;
- study the model of differential λ-calculus in more details;
- do we have denotational completeness?
- Study in particular untyped differential λ-calculus;
- do we get a model of Lionel Vaux's differential $\lambda\mu$-calculus?
- generalize in the spirit of "containers";
- study concrete example of interfaces.

...

Et voilà !