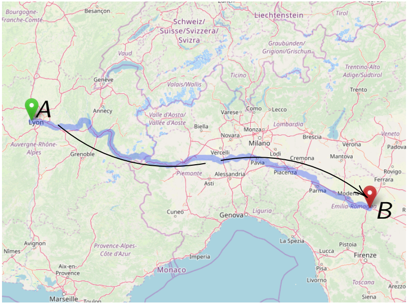
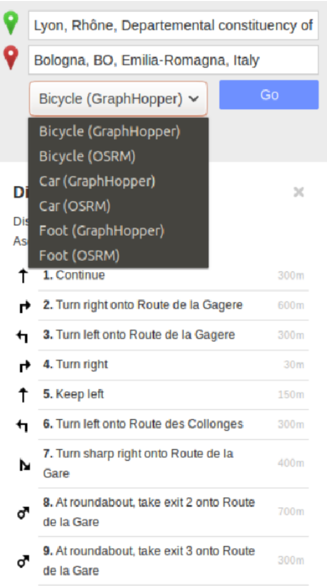
Enriched Concurrent Games:
Witnesses for Proofs and Resource Analysis

Aurore Alcolei

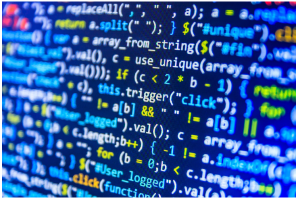
PhD defense – October, 17 2019

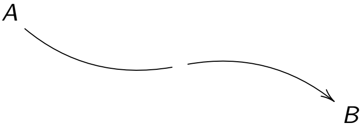
Finding our way in semantics

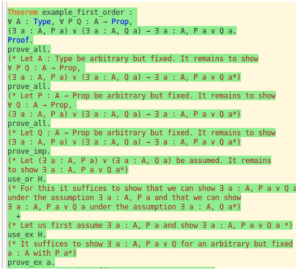




Finding our way in semantics

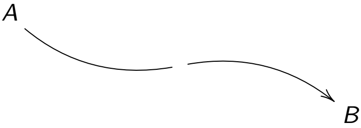


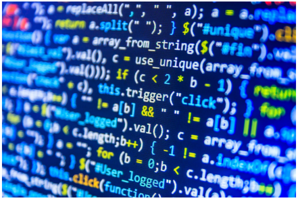


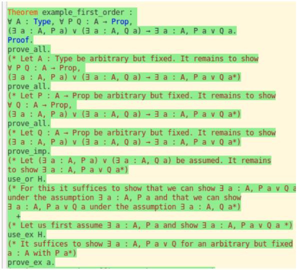


```
Theorem example_first_order :
∀ A : Type, ∀ P Q : A → Prop,
(∃ a : A, P a) v (∃ a : A, Q a) → ∃ a : A, P a v Q a.
Proof.
prove_all.
(* Let A : Type be arbitrary but fixed. It remains to show
∀ P Q : A → Prop,
(∃ a : A, P a) v (∃ a : A, Q a) → ∃ a : A, P a v Q a*)
prove_all.
(* Let P : A → Prop be arbitrary but fixed. It remains to show
∀ Q : A → Prop,
(∃ a : A, P a) v (∃ a : A, Q a) → ∃ a : A, P a v Q a*)
prove_all.
(* Let Q : A → Prop be arbitrary but fixed. It remains to show
(∃ a : A, P a) v (∃ a : A, Q a) → ∃ a : A, P a v Q a*)
prove_imp.
(* Let (∃ a : A, P a) v (∃ a : A, Q a) be assumed. It remains
to show ∃ a : A, P a v Q a*)
use_or H.
(* For this it suffices to show that we can show ∃ a : A, P a v Q a
under the assumption ∃ a : A, P a and that we can show
∃ a : A, P a v Q a under the assumption ∃ a : A, Q a*)
  +
(* Let us first assume ∃ a : A, P a and show ∃ a : A, P a v Q a *)
use_ex H.
(* It suffices to show ∃ a : A, P a v Q for an arbitrary but fixed
a : A with P a*)
prove_ex a.
```

Finding our way in semantics







```
Theorem example_first_order :
∀ A : Type, ∀ P Q : A → Prop,
(∃ a : A, P a) ∨ (∃ a : A, Q a) → ∃ a : A, P a ∨ Q a.
Proof.
prove_all.
(* Let A : Type be arbitrary but fixed. It remains to show
∀ P Q : A → Prop,
(∃ a : A, P a) ∨ (∃ a : A, Q a) → ∃ a : A, P a ∨ Q a*)
prove_all.
(* Let P : A → Prop be arbitrary but fixed. It remains to show
∀ Q : A → Prop,
(∃ a : A, P a) ∨ (∃ a : A, Q a) → ∃ a : A, P a ∨ Q a*)
prove_all.
(* Let Q : A → Prop be arbitrary but fixed. It remains to show
(∃ a : A, P a) ∨ (∃ a : A, Q a) → ∃ a : A, P a ∨ Q a*)
prove_imp.
(* Let (∃ a : A, P a) ∨ (∃ a : A, Q a) be assumed. It remains
to show ∃ a : A, P a ∨ Q a*)
use_or H.
(* For this it suffices to show that we can show ∃ a : A, P a ∨ Q a
under the assumption ∃ a : A, P a and that we can show
∃ a : A, P a ∨ Q a under the assumption ∃ a : A, Q a*)
  +
(* Let us first assume ∃ a : A, P a and show ∃ a : A, P a ∨ Q a *)
use_ex H.
(* It suffices to show ∃ a : A, P a ∨ Q for an arbitrary but fixed
a : A with P a*)
prove_ex a.
```

Denotational semantics

Operational semantics

Semantics of functional programs

$$\mathrm{P} \;=\; \mathrm{fun}\ x \mapsto 4 * x$$

What does P compute?

Denotational semantics.

$$[\![\mathrm{P}]\!] = \begin{array}{ccc} \mathbb{N} & \to & \mathbb{N} \\ n & \mapsto & 4n \end{array}$$

Operational semantics.

$$\begin{array}{rcl} \mathrm{P}\ 2 & = & (\mathrm{fun}\ x \mapsto 4 * x)\ 2 \\ & \to & (4 * 2) \\ & \to & 8 \end{array}$$

$$\mathrm{P}\ 2 \Downarrow 8$$

Semantics of functional programs

$$\mathsf{P} \;=\; \mathsf{fun}\ x \mapsto 4 * x$$

What does P compute?

Denotational semantics.

$$[\![\mathsf{P}]\!] = \begin{array}{ccc} \mathbb{N} & \to & \mathbb{N} \\ n & \mapsto & 4n \end{array}$$

Operational semantics.

$$\begin{array}{rcl} \mathsf{P}\ 2 & = & (\mathsf{fun}\ x \mapsto 4 * x)\ 2 \\ & \to & (4 * 2) \\ & \to & 8 \end{array}$$

$$\mathsf{P}\ 2 \Downarrow 8$$

Semantics of functional programs

$$\begin{aligned}\text{twice} &= \text{fun } f \mapsto \text{fun } x \mapsto f\ (f\ x)\\ \text{double} &= \text{fun } y \mapsto y + y\\ \text{P} &= \text{twice double}\end{aligned}$$

What does P compute?

Denotational semantics.

$$[\![\text{P}]\!] = \begin{array}{ccc}\mathbb{N} & \to & \mathbb{N}\\ n & \mapsto & 4n\end{array}$$

Operational semantics.

$$\begin{array}{lll}\text{P } 2 & \to & (\text{fun } f \mapsto \text{fun } x \mapsto f\ (f\ x))\ \text{double } 2\\ & \to & (\text{fun } x \mapsto \text{double (double } x))\ 2\\ & \to & \text{double (double 2)}\\ & = & (\text{fun } y \mapsto y + y)\ (\text{double 2})\\ & \to & (\text{double 2}) + (\text{double 2})\\ & = & (\text{fun } y \mapsto y + y\ 2) + (\text{double 2})\\ & \to & (2+2) + (\text{double 2})\\ & = & (2+2) + ((\text{fun } y \mapsto y + y)\ 2)\\ & \to & (2+2) + (2+2)\\ & \to & 8\end{array}$$

Semantics of functional programs

$$\begin{aligned} \text{twice} &= \text{fun } f \mapsto \text{fun } x \mapsto f\ (f\ x) \\ \text{double} &= \text{fun } y \mapsto y + y \\ \text{P} &= \text{twice double} \end{aligned}$$

What does P compute?

Denotational semantics.

$$[\![\text{P}]\!] = \begin{array}{ccc} \mathbb{N} & \to & \mathbb{N} \\ n & \mapsto & 4n \end{array}$$

- compositionality:

$$[\![\text{P}]\!] = [\![\text{twice}]\!] \circ [\![\text{double}]\!]$$

- invariant of computation:

$$P \Downarrow v \text{ iff } [\![P]\!] = [\![v]\!]$$

Operational semantics.

$$\begin{aligned} \text{P } 2 &\to (\text{fun } f \mapsto \text{fun } x \mapsto f\ (f\ x))\ \text{double } 2 \\ &\to (\text{fun } x \mapsto \text{double (double } x))\ 2 \\ &\to \text{double (double } 2) \\ &= (\text{fun } y \mapsto y + y)\ (\text{double } 2) \\ &\to (\text{double } 2) + (\text{double } 2) \\ &= (\text{fun } y \mapsto y + y\ 2) + (\text{double } 2) \\ &\to (2+2) + (\text{double } 2) \\ &= (2+2) + ((\text{fun } y \mapsto y + y)\ 2) \\ &\to (2+2) + (2+2) \\ &\to 8 \end{aligned}$$

Curry-Howard isomorphism

Type system

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash P(x) : B}{\Gamma \vdash (\text{fun } x \mapsto P(x)) : A \to B} \qquad \frac{\Gamma \vdash P : A \to B \quad \Gamma \vdash M : A}{\Gamma \vdash P\ M : B}$$

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma, x : A \vdash P(x) : B}}{\Gamma \vdash (\text{fun } x \mapsto P(x)) : A \to B} \qquad \dfrac{\pi_2}{\Gamma \vdash M : A}}{\Gamma \vdash (\text{fun } x \mapsto P(x))\ M : B}$$

Curry-Howard isomorphism

Deduction system

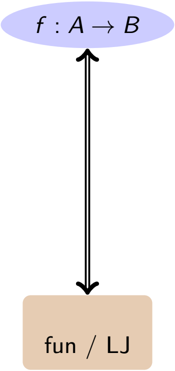
$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma, x : A \vdash P(x) : B}{\Gamma \vdash (\text{fun } x \mapsto P(x)) : A \to B} \qquad \frac{\Gamma \vdash P : A \to B \quad \Gamma \vdash M : A}{\Gamma \vdash P\ M : B}$$

$$\cfrac{\cfrac{\cfrac{\pi_1}{\Gamma, x : A \vdash P(x) : B}}{\Gamma \vdash (\text{fun } x \mapsto P(x)) : A \to B} \quad \cfrac{\pi_2}{\Gamma \vdash M : A}}{\Gamma \vdash (\text{fun } x \mapsto P(x))\ M : B}$$

$\rightsquigarrow$ Computational meaning of proofs: proofs as programs/functions.

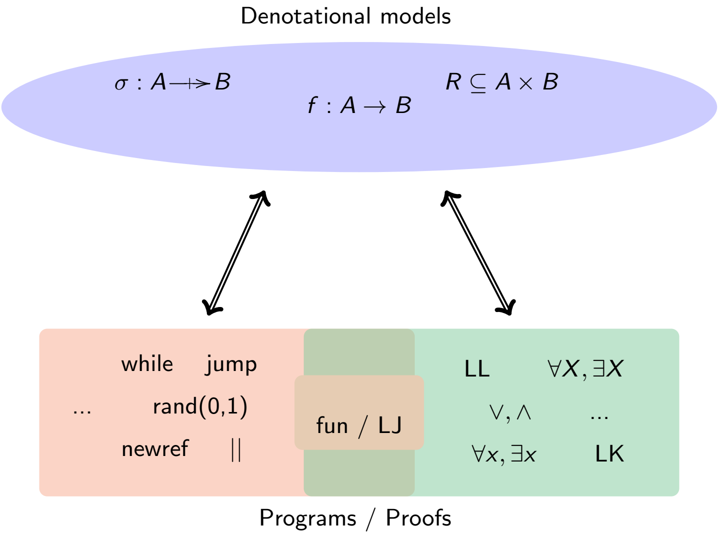
Semantics of programs and proofs

Denotational models



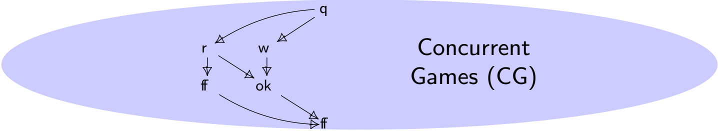
Programs / Proofs

Semantics of programs and proofs



Semantics of programs and proofs

Denotational models



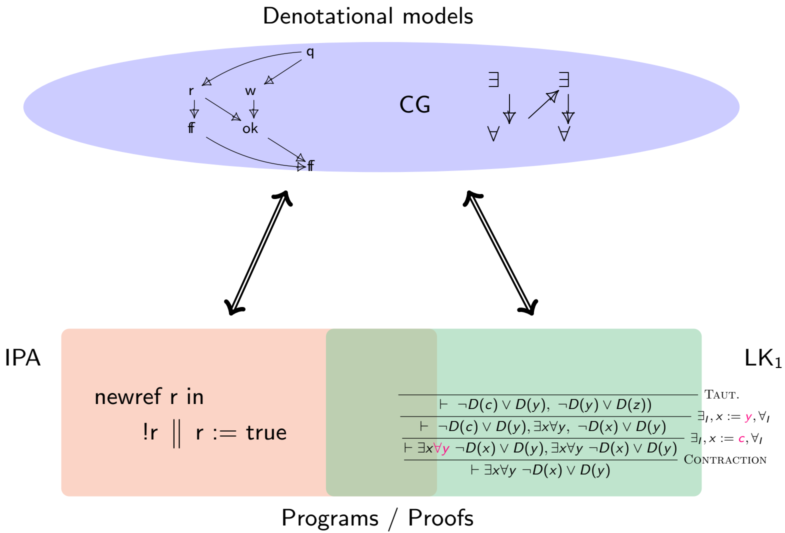
IPA

newref r in
    !r ∥ r := true

Programs / Proofs

[Mel05] Melliès. Asynchronous games
[RW11] Rideau and Winskel. Concurrent strategies

Semantics of programs and proofs

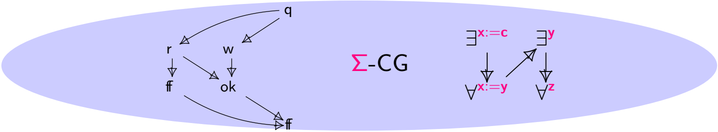


[Mel05] Melliès. Asynchronous games

[RW11] Rideau and Winskel. Concurrent strategies

Semantics of programs and proofs

Denotational models



IPA

newref r in

!r ∥ r := true

LK$_1$

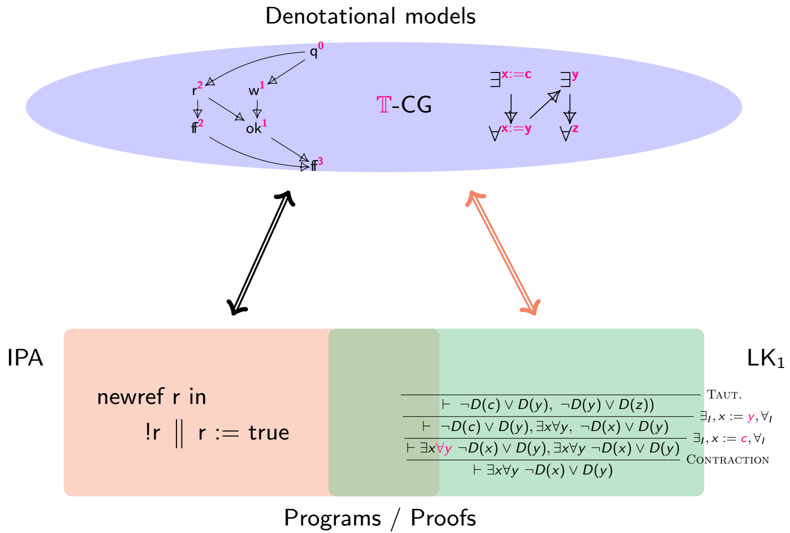
$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash \neg D(c) \vee D(y),\ \neg D(y) \vee D(z))}\ \textsc{Taut.}}{\vdash \neg D(c) \vee D(y), \exists x \forall y,\ \neg D(x) \vee D(y)}\ \exists_I, x := y, \forall_I}{\vdash \exists x \forall y\ \neg D(x) \vee D(y), \exists x \forall y\ \neg D(x) \vee D(y)}\ \exists_I, x := c, \forall_I}{\vdash \exists x \forall y\ \neg D(x) \vee D(y)}\ \textsc{Contraction}$$

Programs / Proofs

[Mel05] Melliès. Asynchronous games

[RW11] Rideau and Winskel. Concurrent strategies

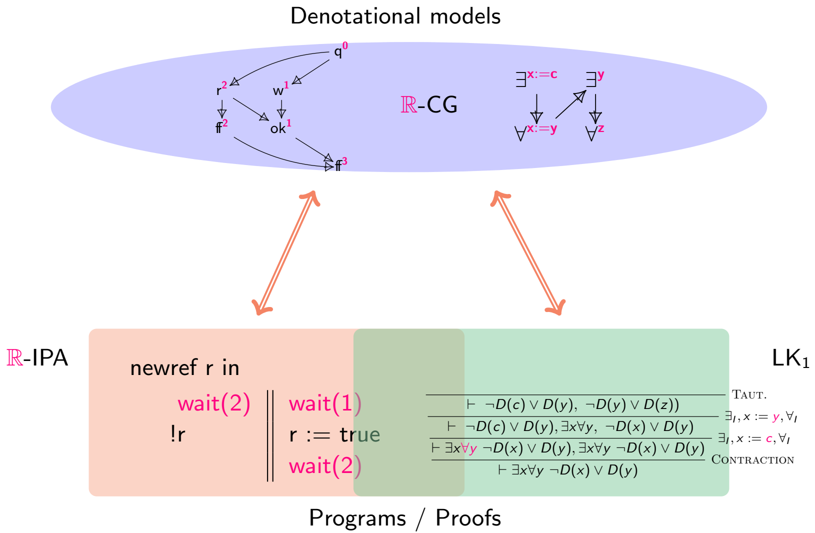
Semantics of programs and proofs



[Mel05] Melliès. Asynchronous games

[RW11] Rideau and Winskel. Concurrent strategies

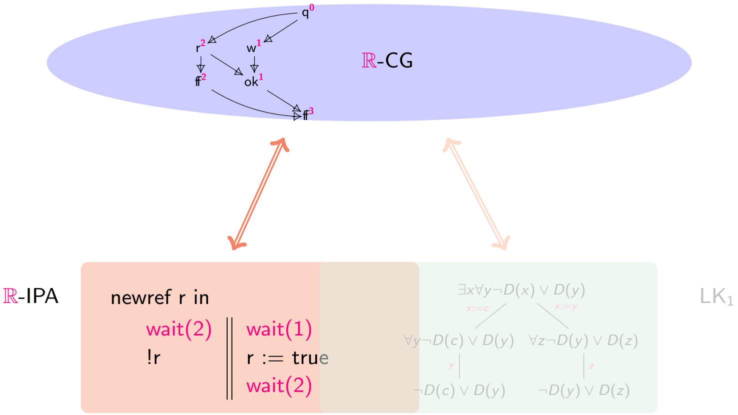
Semantics of programs and proofs



[Mel05] Melliès. Asynchronous games

[RW11] Rideau and Winskel. Concurrent strategies

Annotated concurrent games for time analysis



Time analysis

P =

$$\begin{array}{l}\mathsf{newref\ r\ in} \\ \quad !\mathsf{r} \parallel \mathsf{r} := \mathsf{true}\end{array}$$

Computational adequacy.

$$P \Downarrow v \qquad \text{iff} \qquad [\![v]\!] = [\![P]\!], \qquad\qquad v \in \{\mathsf{true}, \mathsf{false}\}$$

Time analysis

P =

newref r in
    !r ‖ r := true

Computational adequacy.

$$P \Downarrow v \qquad \text{iff} \qquad [\![v]\!] \in [\![P]\!], \qquad\qquad v \in \{\text{true}, \text{false}\}$$

Time analysis

P =

$$\begin{array}{l}\text{newref r in}\\ \quad !\text{r} \parallel \text{r} := \text{true}\end{array}$$

Computational adequacy.

$$P \Downarrow^{t} v \qquad \text{iff} \qquad [\![v]\!]^{t} \in [\![P]\!], \qquad\qquad v \in \{\text{true}, \text{false}\}$$

Q. What is the minimal amount of time necessary to run P?

$\rightarrow$ to get true?

$\rightarrow$ to get false?

Time analysis

P =

$$\text{newref r in} \quad \begin{array}{l|l} \text{wait(2)} & \text{wait(1)} \\ \text{!r} & \text{r := true} \\ & \text{wait(2)} \end{array}$$

Computational adequacy.

$$P \Downarrow^{t} v \qquad \text{iff} \qquad [\![v]\!]^{t} \in [\![P]\!], \qquad\qquad v \in \{\text{true}, \text{false}\}$$

Q. What is the minimal amount of time necessary to run P?

$\rightarrow$ to get true?

$\rightarrow$ to get false?

The $\mathbb{R}$-IPA language

- Types:

$$\begin{array}{lcl} \mathcal{B} & := & \textbf{com} \mid \textbf{bool} \mid \textbf{mem}_R \mid \textbf{mem}_W \\ A, B & := & \mathcal{B} \mid A \multimap B \end{array}$$

- Syntax:

$$\begin{array}{lcll} M, N & := & \mid x \mid \lambda x.t \mid MN & \\ & & \mid \text{true} \mid \text{false} \mid \text{ifte } b\ M\ N & \\ & & \mid \text{skip} \mid M; N \mid M \mid\mid N \mid \bot & \\ & & \mid \text{wait}(\alpha) & \text{with } \alpha \in \mathbb{R} \\ & & \mid \text{newref } r \text{ in } M \mid !M \mid M := \text{true} & \end{array}$$

Interleaving based semantics

P =

```
newref r in
    wait(2)  ||  wait(1)
    !r       ||  r := true
             ||  wait(2)
```

0

Interleaving based semantics

P =

```
newref r in
   wait(2)  ||  wait(1)
   !r       ||  r := true          2
            ||  wait(2)
```

wait(2),

Interleaving based semantics

P =

```
newref r in
   wait(2)  ||  wait(1)
   !r       ||  r := true        2
            ||  wait(2)
```

wait(2), !r,

Interleaving based semantics

P =

```
newref r in
    wait(2)  ||  wait(1)
    !r       ||  r := true          3
             ||  wait(2)
```

wait(2), !r, wait(1),

Interleaving based semantics

P =

```
newref r in
    wait(2)  ||  wait(1)
    !r       ||  r := true        3
             ||  wait(2)
```

wait(2), !r, wait(1), r:=true,

Interleaving based semantics

P =

```
newref r in
    wait(2)  ||  wait(1)
    !r       ||  r := true
             ||  wait(2)
```

5

wait(2), !r, wait(1), r:=true, wait(2) $\quad P \Downarrow^{5}$ false

Interleaving based semantics

P =

newref r in

wait(2)	wait(1)
!r	r := true
	wait(2)

1

wait(2), !r, wait(1), r:=true, wait(2) $P \Downarrow^5$ false

wait(1),

Interleaving based semantics

P =

newref r in

wait(2)	wait(1)
!r	r := true
	wait(2)

3

wait(2), !r, wait(1), r:=true, wait(2) $P \Downarrow^5$ false
wait(1), wait(2),

Interleaving based semantics

P =

```
newref r in
  wait(2)  ||  wait(1)
  !r       ||  r := true        3
           ||  wait(2)
```

wait(2), !r, wait(1), r:=true, wait(2) $P \Downarrow^{5}$ false
wait(1), wait(2), r:=true,

Interleaving based semantics

P =

```
newref r in
    wait(2)  ||  wait(1)
    !r       ||  r := true          3
             ||  wait(2)
```

wait(2), !r, wait(1), r:=true, wait(2) $P \Downarrow^5$ false
wait(1), wait(2), r:=true, !r,

Interleaving based semantics

P =

newref r in

wait(2)	wait(1)
!r	r := true
	wait(2)

5

wait(2), !r, wait(1), r:=true, wait(2) $P \Downarrow^5$ false

wait(1), wait(2), r:=true, !r, wait(2) $P \Downarrow^5$ true

. . .

Slot games[1]

P =

```
newref r in
   wait(2) || wait(1)
   !r      || r := true
           || wait(2)
```

5

wait(2), !r, wait(1), r:=true, wait(2) $P \Downarrow^{5}$ false
wait(1), wait(2), r:=true, !r, wait(2) $P \Downarrow^{5}$ true
...

$[\![P]\!] = \{\text{run } ② \; ① \; ② \; \text{ff}, \; \text{run } ① \; ② \; ② \; \text{tt}, \; \ldots\}$

Computational adequacy: $P \Downarrow^{t} v$ iff $\exists s \in [\![M]\!]$ st $|s| = t$

[1] [Ghica05] Ghica. Slot games: a quantitative model of computation

True concurrency?

Q: What about multicore systems?

P =

```
newref r in
    wait(2)  ||  wait(1)
    !r       ||  r := true
             ||  wait(2)
```

0

True concurrency?

Q: What about multicore systems?

P =

```
newref r in
    wait(2)  ||  wait(1)
    !r       ||  r := true
             ||  wait(2)
```

2

True concurrency?

Q: What about multicore systems?

P =

newref r in		
wait(2)	wait(1)	
!r	r := true	2
	wait(2)	

True concurrency?

Q: What about multicore systems?

P =

newref r in

wait(2)	wait(1)
!r	r := true
	wait(2)

2

True concurrency?

Q: What about multicore systems?

P =

newref r in

wait(2)	wait(1)
!r	r := true
	wait(2)

4

$P \Downarrow^{4}$ false !

True concurrency?

Q: What about multicore systems?

P =

```
newref r in
    wait(2)  ||  wait(1)
    !r       ||  r := true            1
             ||  wait(2)
```

True concurrency?

Q: What about multicore systems?

P =

```
newref r in
    wait(2)  ||  wait(1)
    !r       ||  r := true        1
             ||  wait(2)
```

True concurrency?

Q: What about multicore systems?

P =

```
newref r in
    wait(2)  ||  wait(1)
    !r       ||  r := true        3
             ||  wait(2)
```

True concurrency?

Q: What about multicore systems?

P =

```
newref r in
   wait(2)  ||  wait(1)
   !r       ||  r := true        3
            ||  wait(2)
```

$P \Downarrow^{3}$ true !

True concurrency?

Q: What about multicore systems?

$$\frac{\langle M, s, t\rangle \rightarrow \langle M', s', t'\rangle}{\langle M || N, s, t\rangle \rightarrow \langle M' || N, s', t'\rangle} \qquad \cdots$$

$$\frac{\langle M, s, t\rangle \rightarrow^* \langle M', s', t'\rangle \quad \langle N, s, t\rangle \rightarrow^* \langle N', s'', t''\rangle}{\langle M \mid\mid N, s, t\rangle \rightarrow^* \langle M' \mid\mid N', s' \vee s'', \max(t', t'')\rangle} \; s', s'' \text{non interfering}$$

True concurrency?

Q: What about multicore systems?

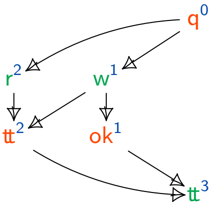
$$\frac{\langle M, s, t\rangle \to \langle M', s', t'\rangle}{\langle M || N, s, t\rangle \to \langle M' || N, s', t'\rangle} \qquad \cdots$$

$$\frac{\langle M, s, t\rangle \to^* \langle M', s', t'\rangle \quad \langle N, s, t\rangle \to^* \langle N', s'', t''\rangle}{\langle M \mid\mid N, s, t\rangle \to^* \langle M' \mid\mid N', s' \vee s'', \mathsf{max}(t', t'')\rangle} \; s', s'' \text{ non interfering}$$

Annotated concurrent games

```
newref r in
    wait(2)  || wait(1)
    !r       || r := true
             || wait(2)
```

$P \Downarrow^{3}$ true

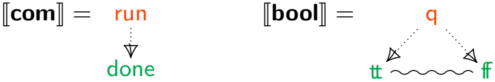


[CC16][2] + time annotations

[2] [CC16] Castellan and Clairambault. Causality vs. interleavings in concurrent game semantics

Concurrent game semantics of IPA [CC16]

- Types as games



Definition

A game is an event structure with polarity: $(|A|, \leq_A, \#_A, \mathrm{pol}_A)$

$\mathscr{C}(A)$ is the set of **configurations**: down-closed compatible subsets of A.

Concurrent game semantics of IPA [CC16]

- Types as games

$$\mathbf{mem} = \quad [\![\mathbf{mem}_R]\!] \quad \otimes \quad [\![\mathbf{mem}_W]\!]$$

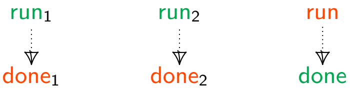
Constructions on games.

- If A is a game, $A^\perp$ has the same structure with polarity inverted.
- If A, B are games, $A \otimes B$ has events $|A| + |B|$, and components inherited.

Concurrent game semantics of IPA [CC16]

- Programs as strategies

$$[\![\, || \,]\!] : \ (\mathbf{com} \quad \otimes \quad \mathbf{com})^{\perp} \quad \otimes \quad \mathbf{com}$$



Definition

A play $(q, \leq_q) : A$ is a partial order s.t.:

* (rule respecting) $\mathscr{C}(q) \subseteq \mathscr{C}(A)$
* (courteous) $a \rightarrowtriangle b$

Concurrent game semantics of IPA [CC16]

- Programs as strategies

$$[\![\ \|\ ]\!] : \quad \mathbf{com} \quad \otimes \quad \mathbf{com} \quad \multimap \quad \mathbf{com}$$

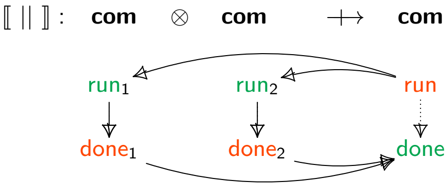
Definition

A play $(q, \leq_q) : A$ is a partial order s.t.:

- (rule respecting) $\mathscr{C}(q) \subseteq \mathscr{C}(A)$
- (courteous) $a \rightarrow b$

Concurrent game semantics of IPA [CC16]

- Programs as strategies



Definition

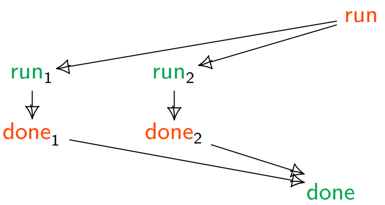
A play $(q, \leq_q) : A$ is a partial order s.t.:

∗ (rule respecting) $\mathscr{C}(q) \subseteq \mathscr{C}(A)$ ∗ (courteous) $a \rightarrowtriangle b$

Concurrent game semantics of IPA [CC16]

- Programs as strategies

$$[\![\, || \,]\!] : \quad \mathbf{com} \quad \otimes \quad \mathbf{com} \quad \stackrel{+}{\to} \quad \mathbf{com}$$



Definition

A play $(q, \leq_q) : A$ is a partial order s.t.:

- (rule respecting) $\mathscr{C}(q) \subseteq \mathscr{C}(A)$
- (courteous) $a \rightarrow b$

Concurrent game semantics of IPA [CC16]

- Programs as strategies

$[\![\text{cell}]\!] \quad : \quad \textbf{mem}$

Definition

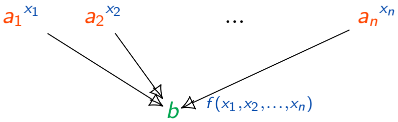
A play $(q, \leq_q) : A$ is a partial order s.t.:

$*$ (rule respecting) $\mathscr{C}(q) \subseteq \mathscr{C}(A)$ $\quad$ $*$ (courteous) $a \rightarrowtriangle b$

A strategy is a down-closed set of plays (with extra conditions).

Annotated concurrent game semantics of ℝ-IPA

- Programs as ℝ-strategies



Definition

A play $(q, \leq_q) : A$ is a partial order s.t.:

- (rule respecting) $\mathscr{C}(q) \subset \mathscr{C}(A)$
- (courteous) $a \rightarrow b$

A ℝ-annotation for q is a mapping $\lambda : (b \in |q|^P) \longrightarrow (\mathbb{R}^{[b]_O} \to \mathbb{R})$.

A ℝ-strategy is a down-closed set of ℝ-annotated plays (with extra conditions).

Annotated concurrent game semantics of $\mathbb{R}$-IPA

- Programs as $\mathbb{R}$-strategies

$[\![\text{skip}]\!]$: **com**

run^{x}

$\downarrow$

done^{x}

Definition

A play $(q, \leq_q) : A$ is a partial order s.t.:

$*$ (rule respecting) $\mathscr{C}(q) \subset \mathscr{C}(A)$ $*$ (courteous) $a \rightarrow b$

A $\mathbb{R}$-annotation for q is a mapping $\lambda : (b \in |q|^{P}) \longrightarrow (\mathbb{R}^{[b]_O} \to \mathbb{R})$.

A $\mathbb{R}$-strategy is a down-closed set of $\mathbb{R}$-annotated plays (with extra conditions).

Annotated concurrent game semantics of $\mathbb{R}$-IPA

- Programs as $\mathbb{R}$-strategies

$$[\![\text{wait}(\alpha)]\!] : \quad \textbf{com}$$

$$\text{run}^{x}$$
$$\downarrow$$
$$\text{done}^{x+\alpha}$$

Definition

A play $(q, \leq_q) : A$ is a partial order s.t.:

$*$ (rule respecting) $\mathscr{C}(q) \subset \mathscr{C}(A)$ $\quad$ $*$ (courteous) $a \rightarrow b$

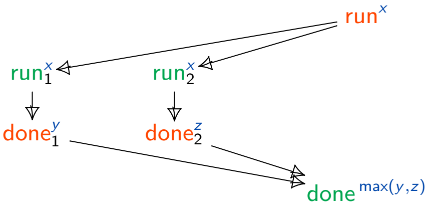
A $\mathbb{R}$-annotation for q is a mapping $\lambda : (b \in |q|^{P}) \longrightarrow (\mathbb{R}^{[b]_O} \to \mathbb{R})$.

A $\mathbb{R}$-strategy is a down-closed set of $\mathbb{R}$-annotated plays (with extra conditions).

Annotated concurrent game semantics of $\mathbb{R}$-IPA

- Programs as $\mathbb{R}$-strategies

$$[\![\ \|\ ]\!] : \mathbf{com} \otimes \mathbf{com} \multimap \mathbf{com}$$



Definition

A play $(q, \leq_q) : A$ is a partial order s.t.:

$*$ (rule respecting) $\mathscr{C}(q) \subseteq \mathscr{C}(A)$ $*$ (courteous) $a \rightarrowtriangle b$

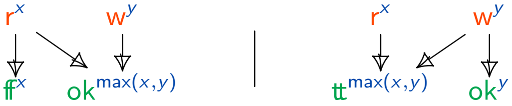
A $\mathbb{R}$-annotation for q is a mapping $\lambda : (b \in |q|^P) \longrightarrow (\mathbb{R}^{[b]_O} \to \mathbb{R})$.

A $\mathbb{R}$-strategy is a down-closed set of $\mathbb{R}$-annotated plays (with extra conditions).

Annotated concurrent game semantics of $\mathbb{R}$-IPA

- Programs as $\mathbb{R}$-strategies

$$[\![\text{cell}]\!] \quad : \quad \textbf{mem}$$



Definition

A play $(q, \leq_q) : A$ is a partial order s.t.:

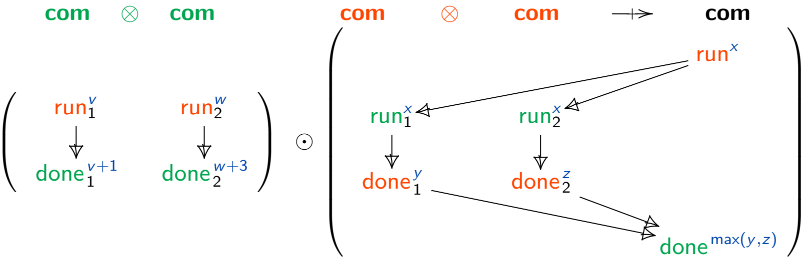
$*$ (rule respecting) $\mathscr{C}(q) \subset \mathscr{C}(A)$ $*$ (courteous) $a \rightarrowtriangle b$

A $\mathbb{R}$-annotation for q is a mapping $\lambda : (b \in |q|^P) \longrightarrow (\mathbb{R}^{[b]_O} \to \mathbb{R})$.

A $\mathbb{R}$-strategy is a down-closed set of $\mathbb{R}$-annotated plays (with extra conditions).

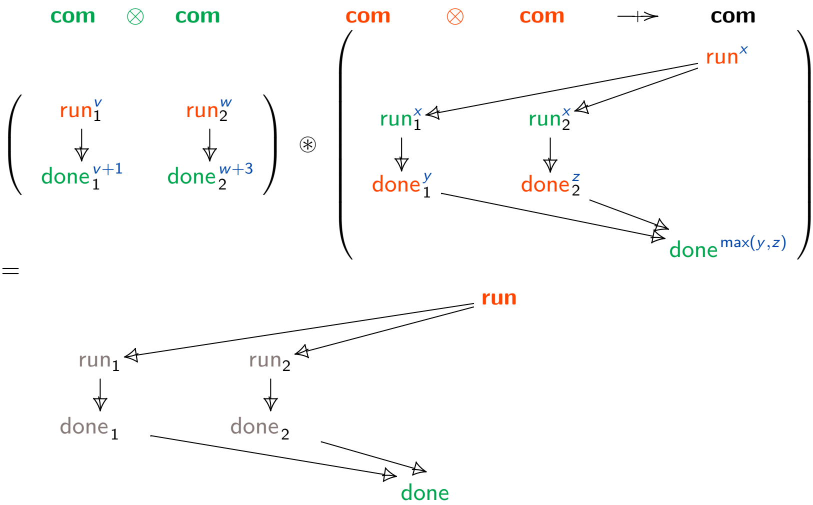
Composition example

$$[\![\mathrm{wait}(1) \parallel \mathrm{wait}(3)]\!] \quad = \quad [\![\ \parallel\ ]\!] \odot ([\![\mathrm{wait}(1)]\!] \otimes [\![\mathrm{wait}(3)]\!])$$



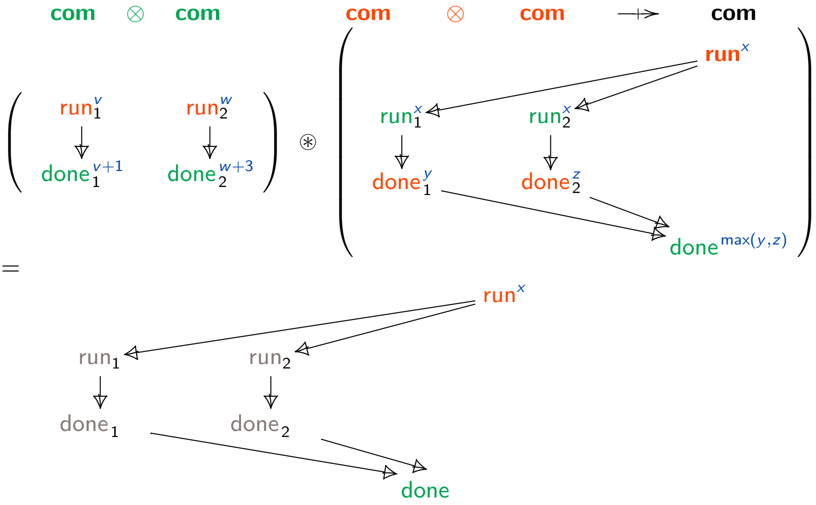
Composition example

$$[\![\text{wait}(1) \parallel \text{wait}(3)]\!] \quad = \quad [\![\ \parallel\ ]\!] \odot ([\![\text{wait}(1)]\!] \otimes [\![\text{wait}(3)]\!])$$



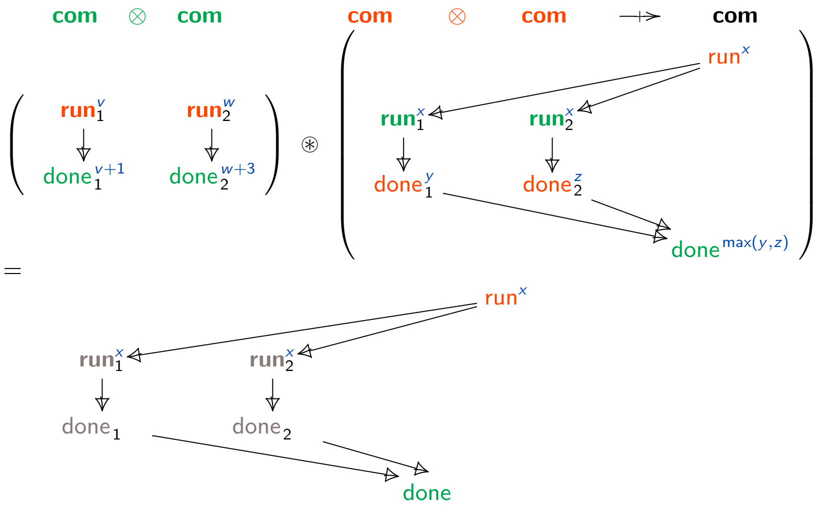
Composition example

$$[\![\text{wait}(1) \parallel \text{wait}(3)]\!] \quad = \quad [\![\ \parallel\ ]\!] \odot ([\![\text{wait}(1)]\!] \otimes [\![\text{wait}(3)]\!])$$



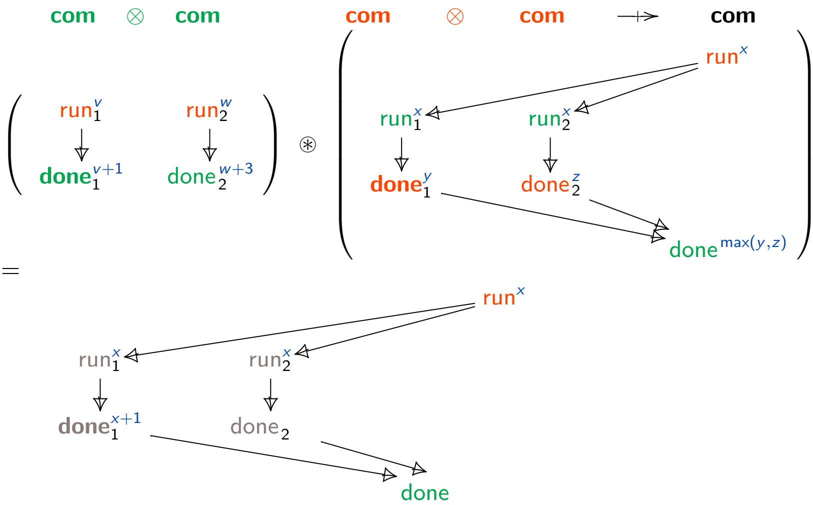
Composition example

$$[\![\mathsf{wait}(1) \mid\mid \mathsf{wait}(3)]\!] \quad = \quad [\![\ \mid\mid\ ]\!] \odot ([\![\mathsf{wait}(1)]\!] \otimes [\![\mathsf{wait}(3)]\!])$$



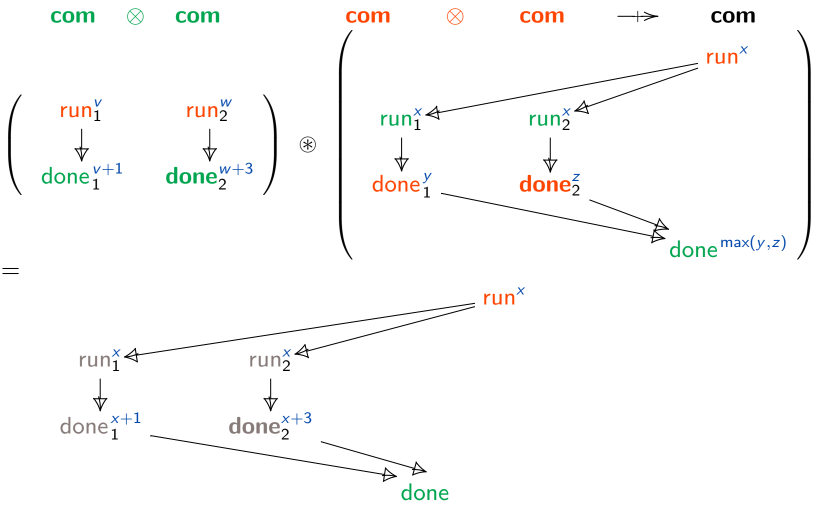
Composition example

$$[\![\mathrm{wait}(1) \parallel \mathrm{wait}(3)]\!] \quad = \quad [\![\ \parallel\ ]\!] \odot ([\![\mathrm{wait}(1)]\!] \otimes [\![\mathrm{wait}(3)]\!])$$



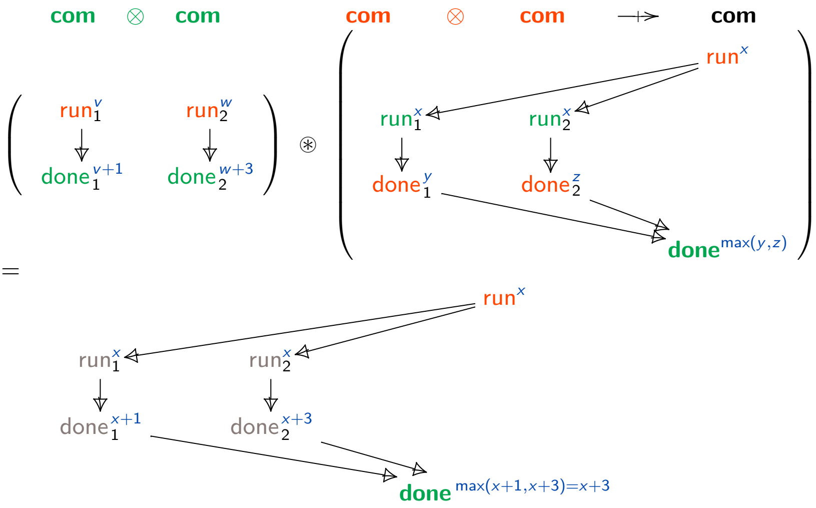
Composition example

$$[\![\text{wait}(1) \mid\mid \text{wait}(3)]\!] \quad = \quad [\![\ \mid\mid\ ]\!] \odot ([\![\text{wait}(1)]\!] \otimes [\![\text{wait}(3)]\!])$$



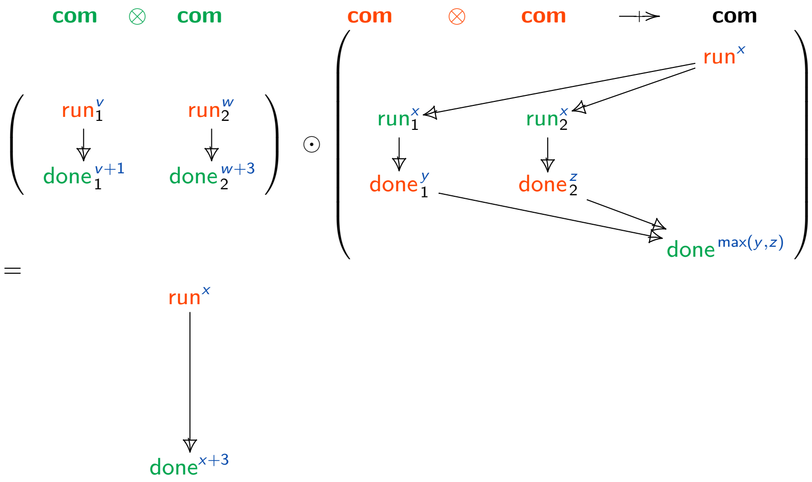
Composition example

$$[\![\mathsf{wait}(1) \mathbin{||} \mathsf{wait}(3)]\!] \quad = \quad [\![\ || \ ]\!] \odot ([\![\mathsf{wait}(1)]\!] \otimes [\![\mathsf{wait}(3)]\!])$$



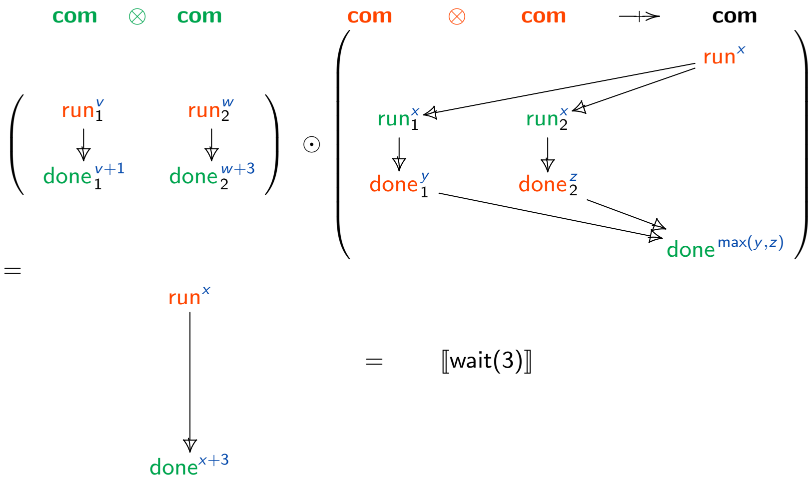
Composition example

$$[\![\text{wait}(1) \parallel \text{wait}(3)]\!] \quad = \quad [\![\ \parallel\ ]\!] \odot ([\![\text{wait}(1)]\!] \otimes [\![\text{wait}(3)]\!])$$



Composition example

$$[\![\mathrm{wait}(1) \mid\mid \mathrm{wait}(3)]\!] \quad = \quad [\![\ \mid\mid\ ]\!] \odot ([\![\mathrm{wait}(1)]\!] \otimes [\![\mathrm{wait}(3)]\!])$$



$$= \quad [\![\mathrm{wait}(3)]\!]$$

Example

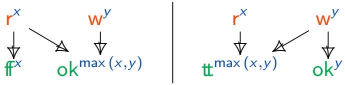
P =

newref r in

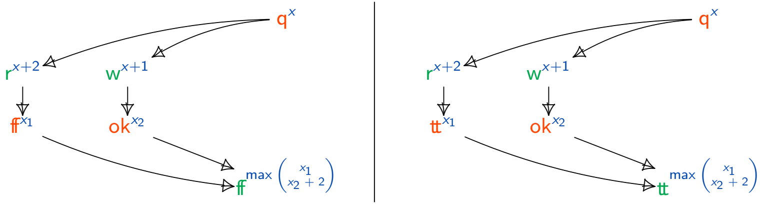
P' {

wait(2)	wait(1)
!r	r := true
	wait(2)

⟦cell⟧ : **mem**



⟦P'⟧ : **mem** ⇸ **bool**



Example

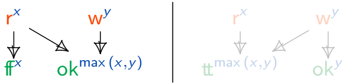
P =

newref r in

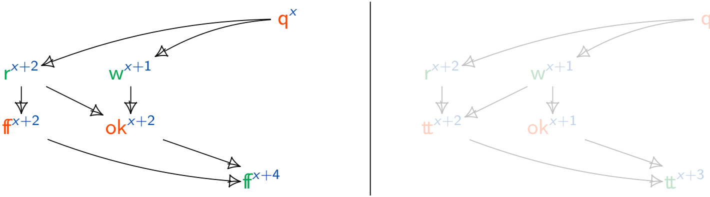
P' {

wait(2)	‖	wait(1)
!r	‖	r := true
	‖	wait(2)

⟦cell⟧ : **mem**



⟦P' ⊛ cell⟧ : **mem** ⟶ **bool**



Example

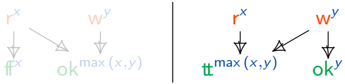
P =

newref r in

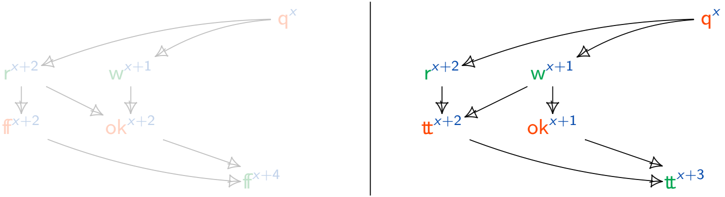
P' {

wait(2)	wait(1)
!r	r := true
	wait(2)

⟦cell⟧ : **mem**



⟦P' ⊛ cell⟧ : **mem** ⟶ **bool**



Interpretation of $\mathbb{R}$-IPA

Theorem

Games and $\mathbb{R}$-strategies with $\odot, \otimes, \perp$ form a compact closed category.

In fact, well-threaded negative games and $\mathbb{R}$-strategies form a symmetric monoidal closed category (smcc) with products.

$$[\![\Gamma \vdash M : A]\!] : [\![\Gamma]\!] \twoheadrightarrow [\![A]\!]$$

$$\begin{array}{rcll}
M, N & := & \mid x \mid \lambda x.t \mid MN & \\
 & & \mid \text{true} \mid \text{false} \mid \text{ifte } b\ M\ N & \\
 & & \mid \text{skip} \mid M; N \mid M \mid\mid N \mid \perp & \checkmark \\
 & & \mid \text{wait}(\alpha) & \checkmark \\
 & & \mid \text{newref } r \text{ in } M \mid !M \mid M := \text{true} & \checkmark
\end{array}$$

Interpretation of $\mathbb{R}$-IPA

Theorem

Games and $\mathbb{R}$-strategies with $\odot, \otimes, \perp$ form a compact closed category.

In fact, well-threaded negative games and $\mathbb{R}$-strategies form a symmetric monoidal closed category (smcc) with products.

$$[\![\Gamma \vdash M : A]\!] : [\![\Gamma]\!] \multimap [\![A]\!]$$

$$\begin{array}{rcll}
M, N & := & \mid x \mid \lambda x.t \mid MN & \checkmark \\
& & \mid \mathsf{true} \mid \mathsf{false} \mid \mathsf{ifte}\ b\ M\ N & \checkmark \\
& & \mid \mathsf{skip} \mid M; N \mid M \mid\mid N \mid \perp & \checkmark \\
& & \mid \mathsf{wait}(\alpha) & \checkmark \\
& & \mid \mathsf{newref}\ r\ \mathsf{in}\ M \mid !M \mid M := \mathsf{true} & \checkmark
\end{array}$$

Computational adequacy [3]

Theorem

Soundness: If $M \Downarrow^{t} v$ then $q^{x} \rightharpoonup v^{x+t'} \in [\![M]\!]$ with $t' \leq t$.

Resource bimonoid

$$\text{wait}(\alpha),\ \ \alpha \in \mathbb{R} \qquad \rightsquigarrow \qquad \text{consume}(\alpha),\ \ \alpha \in \mathcal{R}$$

Theorem

Adequacy: If $q^{x} \rightharpoonup v^{x+t} \in [\![M]\!]$ then $M \Downarrow^{t} v$.

[3] [ACL19] A., Clairambault and Laurent. Resource-tracking concurrent games

Computational adequacy [3]

Theorem

Soundness: If $M \Downarrow^{t} v$ then $q^{x} \rightharpoonup v^{x+t'} \in [\![M]\!]$ with $t' \leq t$.

Resource bimonoid

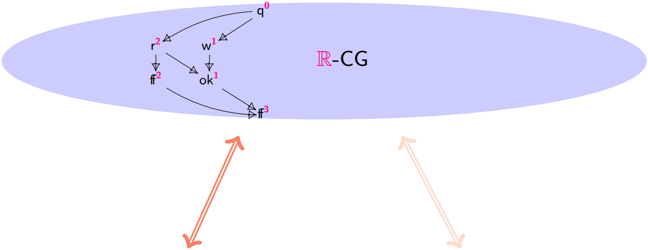
$$\mathsf{wait}(\alpha),\ \ \alpha \in \mathbb{R} \qquad \rightsquigarrow \qquad \mathsf{consume}(\alpha),\ \ \alpha \in \mathcal{R}$$

Theorem

Adequacy: If $q^{x} \rightharpoonup v^{x+t} \in [\![M]\!]$ then $M \Downarrow^{t} v$.

[3] [ACL19] A., Clairambault and Laurent. Resource-tracking concurrent games

Annotated concurrent games



ℝ-IPA

```
newref (r,false) in
    wait(2)  ||  wait(1)
    !r       ||  r := true
             ||  wait(2)
```

$\exists x \forall y \neg D(x) \vee D(y)$

$x := c$ / $x := y$

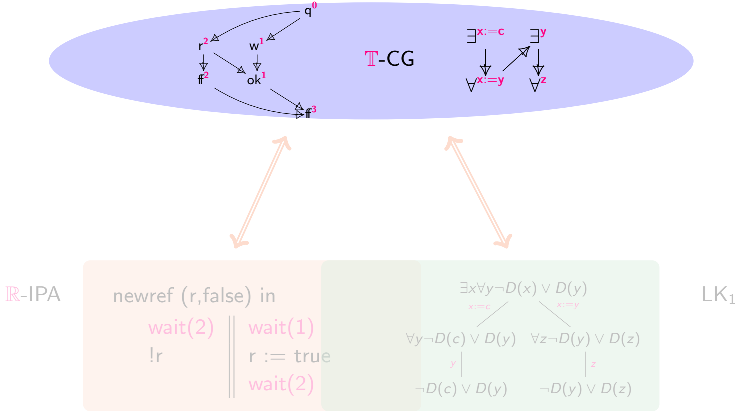
$\forall y \neg D(c) \vee D(y)$ $\quad$ $\forall z \neg D(y) \vee D(z)$

y / z

$\neg D(c) \vee D(y)$ $\quad$ $\neg D(y) \vee D(z)$

LK_1

Annotated concurrent games



Concurrent games: CG

Definition

A game is an event structure with polarity: $(|A|, \leq_A, \#_A, \mathrm{pol}_A)$

A play $(q, \leq_q) : A$ is a partial order s.t.:

* (rule respecting) $\mathscr{C}(q) \subseteq \mathscr{C}(A)$
* (courteous) $a \rightarrowtriangle b$

A strategy is a down-closed set of annotated plays (with extra conditions).

Theorem

Games and strategies with $\odot, \otimes, \perp$ form a compact closed category.

Annotated concurrent games: $\mathbb{R}$-CG

Definition

A game is an event structure with polarity: $(|A|, \leq_A, \#_A, \mathrm{pol}_A)$

A play $(q, \leq_q) : A$ is a partial order s.t.:

- (rule respecting) $\mathscr{C}(q) \subseteq \mathscr{C}(A)$
- (courteous) $a \rightarrow b$

A $\mathbb{R}$-annotation for q is a mapping $\lambda : (b \in |q|^P) \longrightarrow (\mathbb{R}^{[s]^O} \rightarrow \mathbb{R})$.

A $\mathbb{R}$-strategy is a down-closed set of $\mathbb{R}$-annotated plays (with extra conditions).

Theorem

Games and $\mathbb{R}$-strategies with $\odot, \otimes, \perp$ form a compact closed category.

Annotated concurrent games: $\mathbb{T}$-CG

Equational theory: $\mathbb{T} = (\theta, \Sigma, \equiv)$

Definition

A $\mathbb{T}$-game is an event structure with polarity: $(|A|, \leq_A, \#_A, \mathrm{pol}_A)$ together with a typing function $|A| \to \theta$.

A play $(q, \leq_q) : A$ is a partial order s.t.:

- (rule respecting) $\mathscr{C}(q) \subseteq \mathscr{C}(A)$
- (courteous) $a \rightarrowtriangle b$

A $\mathbb{T}$-annotation for q is a mapping $\lambda : (b \in |q|^P) \longrightarrow \mathbb{T}\,(\theta([s]_O), \theta(s))$.

A $\mathbb{T}$-strategy is a down-closed set of $\mathbb{T}$-annotated plays (with extra conditions).

Theorem

$\mathbb{T}$-games and $\mathbb{T}$-strategies with $\odot, \otimes, \perp$ form a compact closed category.

Annotated concurrent games: from $\mathbb{T}$-CG to $\mathbb{R}$-CG

- Real functions: $\mathbb{R}$-CG

$$\theta = \{\mathbb{R}\}$$

$$\mathsf{f} \in \Sigma^n \text{ iff } f : \mathbb{R}^n \to \mathbb{R} \qquad \rightsquigarrow \qquad \forall t \in \mathsf{Tm}(\mathcal{V}),\ \overline{t} : \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}$$

$$t_1 \equiv t_2 \in \mathsf{Tm}(\mathcal{V}) \text{ iff } \overline{t_1} = \overline{t_2} \qquad \rightsquigarrow \qquad \overline{t_1[t_2/x]} = \overline{t_1} \circ_x \overline{t_2}$$

- Cartesian category: $\mathcal{C}$-CG $\qquad \theta = \mathcal{C}_0 \qquad \ldots$

- Terms: Σ-CG $\qquad \mathbb{T} = (\{\bullet\}, \Sigma, \emptyset)$

Annotated concurrent games: from $\mathbb{T}$-CG to $\mathbb{R}$-CG

- Real functions: $\mathbb{R}$-CG

$$\theta = \{\mathbb{R}\}$$

$$\mathsf{f} \in \Sigma^n \text{ iff } f : \mathbb{R}^n \to \mathbb{R} \quad \rightsquigarrow \quad \forall t \in \mathsf{Tm}(\mathcal{V}),\ \overline{t} : \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}$$

$$t_1 \equiv t_2 \in \mathsf{Tm}(\mathcal{V}) \text{ iff } \overline{t_1} = \overline{t_2} \quad \rightsquigarrow \quad \overline{t_1[t_2/x]} = \overline{t_1} \circ_x \overline{t_2}$$

- Cartesian category: $\mathcal{C}$-CG $\quad \theta = \mathcal{C}_0 \quad \ldots$

- Terms: Σ-CG $\quad \mathbb{T} = (\{\bullet\}, \Sigma, \emptyset)$

Annotated concurrent games: from $\mathbb{T}$-CG to $\mathbb{R}$-CG

- Real functions: $\mathbb{R}$-CG

$$\theta = \{\mathbb{R}\}$$

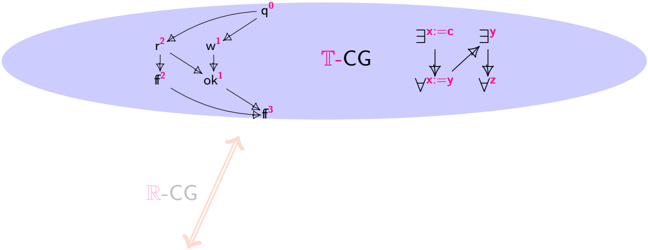
$$\mathsf{f} \in \Sigma^n \text{ iff } f : \mathbb{R}^n \to \mathbb{R} \quad \rightsquigarrow \quad \forall t \in \mathsf{Tm}(\mathcal{V}),\ \overline{t} : \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}$$

$$t_1 \equiv t_2 \in \mathsf{Tm}(\mathcal{V}) \text{ iff } \overline{t_1} = \overline{t_2} \quad \rightsquigarrow \quad \overline{t_1[t_2/x]} = \overline{t_1} \circ_x \overline{t_2}$$

- Cartesian category: $\mathcal{C}$-CG $\qquad \theta = \mathcal{C}_0 \qquad \ldots$

- Terms: Σ-CG $\qquad \mathbb{T} = (\{\bullet\}, \Sigma, \emptyset)$

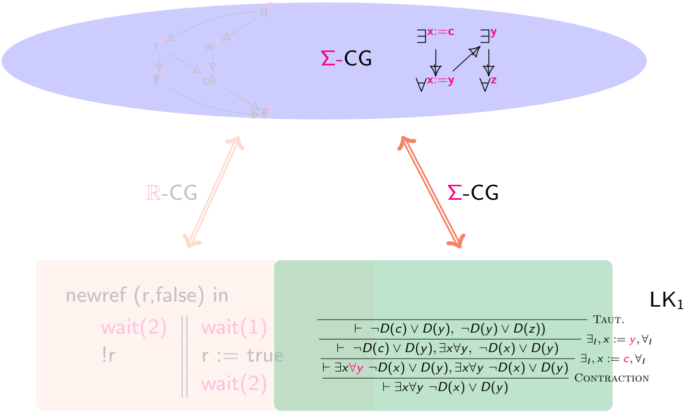
Annotated concurrent games for Herbrand's theorem




```
newref (r,false) in
   wait(2)  ||  wait(1)
   !r       ||  r := true
            ||  wait(2)
```

LK_1

Annotated concurrent games for Herbrand's theorem



Herbrand's witnesses

Herbrand's theorem (Simple)

A purely existential formula $\exists x\ \varphi(x)$ is valid in classical logic iff there is a finite set of witnesses $t_1, \ldots, t_n \in Tm_\Sigma$ s.t. $\models \varphi(t_1) \vee \ldots \vee \varphi(t_n)$.

Example $\models \exists x\ \neg D(x) \vee D(f(x))$

$$\models (\neg D(c) \vee D(f(c))) \vee (\neg D(f(c)) \vee D(f(f(c))))$$

Herbrand's witnesses

Herbrand's theorem (Simple)

A purely existential formula $\exists x\ \varphi(x)$ is valid in classical logic iff there is a finite set of witnesses $t_1, \ldots, t_n \in Tm_\Sigma$ s.t. $\models \varphi(t_1) \vee \ldots \vee \varphi(t_n)$.

Example $\models \exists x\ \neg D(x) \vee D(f(x))$

$$\models (\neg D(c) \vee D(f(c))) \vee (\neg D(f(c)) \vee D(f(f(c))))$$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash\ \neg D(c) \vee D(f(c)),\ \neg D(f(c)) \vee D(f(f(c)))}\ \textsc{Prop. Tautology}}{\vdash\ \neg D(c) \vee D(f(c)),\ \exists x\ \neg D(x) \vee D(f(x))}\ \exists_I, x := f(c)}{\vdash \exists x\ \neg D(x) \vee D(f(x)),\ \exists x\ \neg D(x) \vee D(f(x))}\ \exists_I, x := c}{\vdash \exists x\ \neg D(x) \vee D(f(x))}\ \textsc{Contraction}$$

Herbrand proofs

Herbrand's theorem (General)

A 1^{st} order formula φ is valid in classical logic iff it has a Herbrand proof.

Example $\models \exists x \forall y,\ \neg D(x) \vee D(y)$ (DF)

A proof for DF:

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{\vdash\ \neg D(c) \vee D(y),\ \neg D(y) \vee D(z))}\ \textsc{Prop. Tautology}}{\vdash\ \neg D(c) \vee D(y), \exists x \forall y,\ \neg D(x) \vee D(y)}\ \exists_I, x := y, \forall_I}{\vdash \exists x \forall y\ \neg D(x) \vee D(y), \exists x \forall y\ \neg D(x) \vee D(y)}\ \exists_I, x := c, \forall_I}{\vdash \exists x \forall y\ \neg D(x) \vee D(y)}\ \textsc{Contraction}
$$

[Mil87] Miller. A compact representation of proofs.

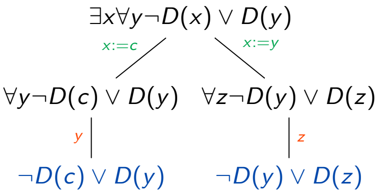
Herbrand proofs: Miller's **expansion trees**

Herbrand's theorem (Miller, 1987)

A 1^{st} *order formula* φ *is valid in classical logic iff it has an expansion tree.*

Example $\models \exists x \forall y,\ \neg D(x) \vee D(y)$ (DF)

An expansion tree for DF:



[Mil87] Miller. A compact representation of proofs.

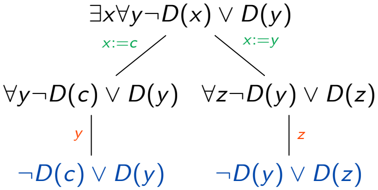
Herbrand proofs: Miller's **expansion trees**

Herbrand's theorem (Miller, 1987)

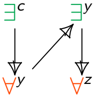
A 1^{st} order formula φ is valid in classical logic iff it has an expansion tree.

Example $\models \exists x \forall y,\ \neg D(x) \vee D(y)$ (DF)

An expansion tree for DF:



acyclicity



validity

$$\models (\neg D(c) \vee D(y)) \vee (\neg D(y) \vee D(z))$$

[Mil87] Miller. A compact representation of proofs.

Herbrand proofs: Miller's **expansion trees**

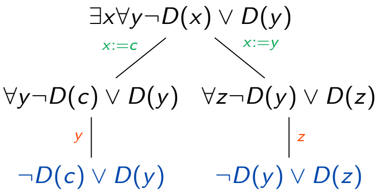
Herbrand's theorem (Miller, 1987)

A 1^{st} *order formula* φ *is valid in classical logic iff it has an expansion tree.*

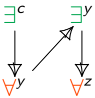
Proof: By translation from the cut-free sequent calculus. $\rightarrow$ not compositional.

Example $\models \exists x \forall y,\ \neg D(x) \vee D(y)$ (DF)

An expansion tree for DF:



acyclicity



validity

$$\models (\neg D(c) \vee D(y)) \quad \vee \quad (\neg D(y) \vee D(z))$$

[Mil87] Miller. A compact representation of proofs.

Composable Expansion Trees?

Syntactic approaches: Heijltjes, [4] Hetzl and Weller, [5] McKinley, [6] via notions of Herbrand proofs with cuts.

$$\frac{\pi}{\vdash \varphi} \quad \rightsquigarrow \quad [\![\pi]\!] : [\![\varphi]\!] \qquad\qquad \sigma = \sigma_1 \odot \sigma_2$$

Contribution [7] (semantic approach): Expansion trees as strategies in a concurrent game model (categories of winning Σ-strategies).

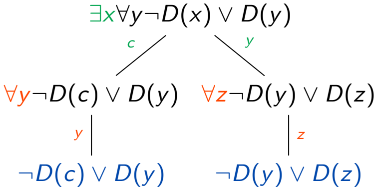
Herbrand's theorem (Compositional Herbrand's theorem)

A 1^{st} order formula φ is valid iff there is a winning Σ-strategy: $\sigma : [\![\varphi]\!]$.

[4] [Hei10] Heijltjes. Classical proof forestry.
[5] [HW13] Hetzl and Weller. Expansion trees with cut.
[6] [MCK13] McKinley. Proof nets for Herbrand's theorem
[7] [ACHW18] A., Clairambault, Hyland, Winskel. The True Concurrency of Herbrand's Theorem

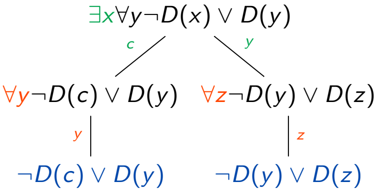
From expansion trees to Σ-strategies

An implicit two-player game played on the formula between ∃loïse and ∀bélard:

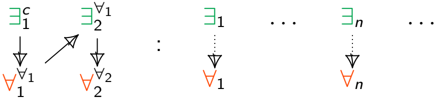


From expansion trees to Σ-strategies

An implicit two-player game played on the formula between ∃loïse and ∀bélard:

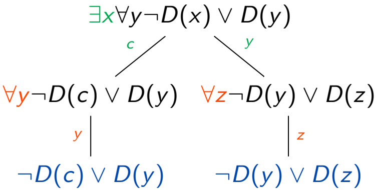


An interpretation of formulas as games and proofs as Σ-strategies:

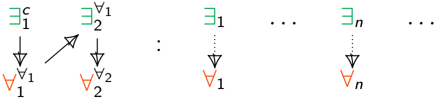


From expansion trees to Σ-strategies

An implicit two-player game played on the formula between ∃loïse and ∀bélard:

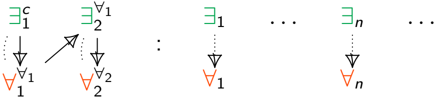


An interpretation of formulas as games and proofs as winning Σ-strategies:



Example of winning conditions

Consider the Σ-strategy $\sigma : [\![\exists x\ \forall y\ \neg D(x) \vee D(y)]\!]$ over DF

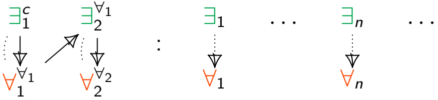


Validity in expansion trees:

$$\models (\neg D(c) \vee D(\forall_1)) \quad \vee \quad (\neg D(\forall_1) \vee D(\forall_2))$$

Example of winning conditions

Consider the Σ-strategy $\sigma : [\![\exists x\ \forall y\ \neg D(x) \vee D(y)]\!]$ over DF



Validity in expansion trees:

$$\models (\neg D(c) \vee D(\forall_1)) \quad \vee \quad (\neg D(\forall_1) \vee D(\forall_2))$$

Can be decomposed into

$$\models \quad \underbrace{(\neg D(\exists_1) \vee D(\forall_1)) \quad \vee \quad (\neg D(\exists_2) \vee D(\forall_2))}_{\text{Winning conditions}, \mathcal{W}_{\mathrm{DF}}(|\sigma|)} \quad \underbrace{[\exists_1 \mapsto c; \exists_2 \mapsto \forall_1]}_{\text{Labelling}, \lambda_\sigma}$$

Winning conditions on arenas

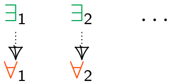
Definition

A game $\mathcal{A}$ is an arena A, together with winning conditions:

$$\mathcal{W}_{\mathcal{A}} : (x \in \mathscr{C}(A)) \mapsto \mathrm{QF}_{\Sigma}(x)$$

where $\mathrm{QF}_{\Sigma}(x)$ is the set of **quantifier-free** formulas on signature Σ and free variables in x, extended with **countable** conjunctions and disjunctions.

To each configuration of $[\![\exists x\ \forall y\ \neg D(x) \vee D(y)]\!]$, we associate a formula:



σ is a winning on x if $\quad \models \mathcal{W}_A(x)[\lambda_\sigma]$.

Winning conditions on arenas

Definition

A game $\mathcal{A}$ is an arena A, together with winning conditions:

$$\mathcal{W}_{\mathcal{A}} : (x \in \mathscr{C}(A)) \mapsto \mathrm{QF}_{\Sigma}(x)$$

where $\mathrm{QF}_{\Sigma}(x)$ is the set of **quantifier-free** formulas on signature Σ and free variables in x, extended with **countable** conjunctions and disjunctions.

To each configuration of $[\![\exists x\ \forall y\ \neg D(x) \vee D(y)]\!]$, we associate a formula:

$$\mapsto \qquad \bot$$

σ is a winning on x if $\models \mathcal{W}_A(x)[\lambda_\sigma]$.

Winning conditions on arenas

Definition

A game $\mathcal{A}$ is an arena A, together with winning conditions:

$$\mathcal{W}_{\mathcal{A}} : (x \in \mathscr{C}(A)) \mapsto \mathrm{QF}_{\Sigma}(x)$$

where $\mathrm{QF}_{\Sigma}(x)$ is the set of **quantifier-free** formulas on signature Σ and free variables in x, extended with **countable** conjunctions and disjunctions.

To each configuration of $[\![\exists x \; \forall y \; \neg D(x) \vee D(y)]\!]$, we associate a formula:

$\exists_1$ $\mapsto$ $\top$

σ is a winning on x if $\models \mathcal{W}_A(x)[\lambda_\sigma]$.

Winning conditions on arenas

Definition

A game $\mathcal{A}$ is an arena A, together with winning conditions:

$$\mathcal{W}_{\mathcal{A}} : (x \in \mathscr{C}(A)) \mapsto \mathrm{QF}_{\Sigma}(x)$$

where $\mathrm{QF}_{\Sigma}(x)$ is the set of **quantifier-free** formulas on signature Σ and free variables in x, extended with **countable** conjunctions and disjunctions.

To each configuration of $[\![\exists x\ \forall y\ \neg D(x) \vee D(y)]\!]$, we associate a formula:

$$\mapsto \qquad \neg D(\exists_1) \vee D(\forall_1)$$

σ is a winning on x if $\models \mathcal{W}_A(x)[\lambda_\sigma]$.

Winning conditions on arenas

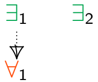
Definition

A game $\mathcal{A}$ is an arena A, together with winning conditions:

$$\mathcal{W}_{\mathcal{A}} : (x \in \mathscr{C}(A)) \mapsto \mathrm{QF}_{\Sigma}(x)$$

where $\mathrm{QF}_{\Sigma}(x)$ is the set of **quantifier-free** formulas on signature Σ and free variables in x, extended with **countable** conjunctions and disjunctions.

To each configuration of $[\![\exists x\ \forall y\ \neg D(x) \vee D(y)]\!]$, we associate a formula:



$$\mapsto \qquad (\neg D(\exists_1) \vee D(\forall_1)) \quad \vee \quad \top$$

σ is a winning on x if $\quad \models \mathcal{W}_A(x)[\lambda_\sigma]$.

Winning conditions on arenas

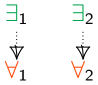
Definition

A game $\mathcal{A}$ is an arena A, together with winning conditions:

$$\mathcal{W}_{\mathcal{A}} : (x \in \mathscr{C}(A)) \mapsto \mathrm{QF}_{\Sigma}(x)$$

where $\mathrm{QF}_{\Sigma}(x)$ is the set of **quantifier-free** formulas on signature Σ and free variables in x, extended with **countable** conjunctions and disjunctions.

To each configuration of $[\![\exists x \, \forall y \, \neg D(x) \vee D(y)]\!]$, we associate a formula:



$$\mapsto \qquad (\neg D(\exists_1) \vee D(\forall_1)) \quad \vee \quad (\neg D(\exists_2) \vee D(\forall_2))$$

σ is a winning on x if $\models \mathcal{W}_A(x)[\lambda_\sigma]$.

Winning conditions on arenas

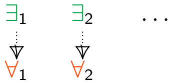
Definition

A game $\mathcal{A}$ is an arena A, together with winning conditions:

$$\mathcal{W}_{\mathcal{A}} : (x \in \mathscr{C}(A)) \mapsto \mathrm{QF}_{\Sigma}(x)$$

where $\mathrm{QF}_{\Sigma}(x)$ is the set of **quantifier-free** formulas on signature Σ and free variables in x, extended with **countable** conjunctions and disjunctions.

To each configuration of $[\![\exists x \, \forall y \, \neg D(x) \vee D(y)]\!]$, we associate a formula:



$$\mapsto \quad (\neg D(\exists_1) \vee D(\forall_1)) \quad \vee \quad (\neg D(\exists_2) \vee D(\forall_2)) \quad \vee \quad \ldots$$

σ is a winning on x if $\quad \models \mathcal{W}_A(x)[\lambda_\sigma]$.

Winning conditions on arenas

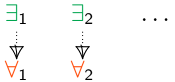
Definition

A game $\mathcal{A}$ is an arena A, together with winning conditions:

$$\mathcal{W}_{\mathcal{A}} : (x \in \mathscr{C}(A)) \mapsto \mathrm{QF}_{\Sigma}(x)$$

where $\mathrm{QF}_{\Sigma}(x)$ is the set of **quantifier-free** formulas on signature Σ and free variables in x, extended with **countable** conjunctions and disjunctions.

To each configuration of $[\![\exists x\ \forall y\ \neg D(x) \vee D(y)]\!]$, we associate a formula:



$$\mapsto \quad (\neg D(\exists_1) \vee D(\forall_1)) \quad \vee \quad (\neg D(\exists_2) \vee D(\forall_2)) \quad \vee \quad \ldots$$

σ is a winning on x if $\quad \models \mathcal{W}_A(x)[\lambda_\sigma]$.

Winning conditions on arenas

Definition

A game $\mathcal{A}$ is an arena A, together with winning conditions:

$$\mathcal{W}_{\mathcal{A}} : (x \in \mathscr{C}(A)) \mapsto \mathrm{QF}_{\Sigma}(x)$$

where $\mathrm{QF}_{\Sigma}(x)$ is the set of **quantifier-free** formulas on signature Σ and free variables in x, extended with **countable** conjunctions and disjunctions.

Definition

A Σ-strategy $\sigma : A$ is winning on $\mathcal{W}_{\mathcal{A}}$ iff for all $x \in \mathcal{C}^{\infty}(\sigma)$ $\exists$-**maximal**,

$$\models \mathcal{W}_{A}(x)[\lambda_{\sigma}]$$

→ Two new constructors on games: $\otimes$ (conjunction) and $⅋$ (disjunction)

with units $1 = (\emptyset, \mathcal{W}_1(\emptyset) = \top)$ $\quad \bot = (\emptyset, \mathcal{W}_{\bot}(\emptyset) = \bot)$

→ Winning strategies $\sigma : \mathcal{A}^{\perp} ⅋ \mathcal{B}$ are **stable under composition (⋆-autonomous category).**

LK_1 interpretation

Propositional connectives (MLL $*$-autonomous model)

Quantifiers

$$[\![\exists x \varphi]\!]_{\mathcal{V}} = \exists_x . [\![\varphi]\!]_{\mathcal{V} \uplus \{x\}} \qquad [\![\forall x \varphi]\!]_{\mathcal{V}} = \forall_x . [\![\varphi]\!]_{\mathcal{V} \uplus \{x\}}$$

Weakening: for any formula φ, $w_{[\![\varphi]\!]} : \bot \longrightarrow [\![\varphi]\!]$

Contraction: for any formula φ, $c_{[\![\varphi]\!]} : [\![\varphi]\!] \parr [\![\varphi]\!] \longrightarrow [\![\varphi]\!]$

LK_1 interpretation

Propositional connectives (MLL $*$-autonomous model)

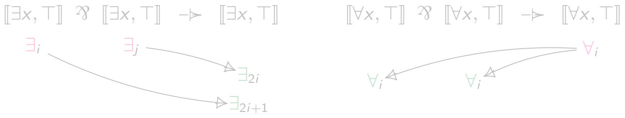
Quantifiers

$$[\![\exists x \varphi]\!]_{\mathcal{V}} = \exists_x . [\![\varphi]\!]_{\mathcal{V} \uplus \{x\}} \qquad [\![\forall x \varphi]\!]_{\mathcal{V}} = \forall_x . [\![\varphi]\!]_{\mathcal{V} \uplus \{x\}}$$

Weakening: for any formula φ, $w_{[\![\varphi]\!]} : \bot \to [\![\varphi]\!]$

$$\mathrm{W}\ \frac{\vdash^{\mathcal{V}} \Gamma}{\vdash^{\mathcal{V}} \Gamma, \varphi}$$

Contraction: for any formula φ, $c_{[\![\varphi]\!]} : [\![\varphi]\!] \parr [\![\varphi]\!] \to [\![\varphi]\!]$



LK_1 interpretation

Propositional connectives (MLL $*$-autonomous model)

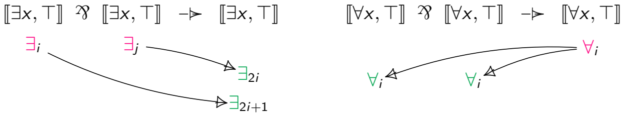
Quantifiers

$$[\![\exists x\varphi]\!]_{\mathcal{V}} = \bigparr_{n\in\omega} \exists_x . [\![\varphi]\!]_{\mathcal{V}\uplus\{x\}} \qquad [\![\forall x\varphi]\!]_{\mathcal{V}} = \bigotimes_{n\in\omega} \forall_x . [\![\varphi]\!]_{\mathcal{V}\uplus\{x\}}$$

Weakening: for any formula φ, $w_{[\![\varphi]\!]} : \bot \longrightarrow [\![\varphi]\!]$

Contraction: for any formula φ, $c_{[\![\varphi]\!]} : [\![\varphi]\!] \parr [\![\varphi]\!] \longrightarrow [\![\varphi]\!]$

$$\mathrm{C}\ \frac{\vdash^{\mathcal{V}} \Gamma, \varphi, \varphi}{\vdash^{\mathcal{V}} \Gamma, \varphi}$$



Back to Herbrand's proofs

Herbrand's theorem (Compositional Herbrand's theorem)

A 1^{st} order formula φ is valid iff there is a winning Σ-strategy: $\sigma : [\![\varphi]\!]$.

Proof:

$\Rightarrow$ Interpret the classical sequent calculus LK_1.

$\Leftarrow$ Winning strategies ressemble expansion trees ...

Lemma (Compactness)

From every winning strategy $\sigma : [\![\varphi]\!]$ one can **effectively extract a finite expansion tree** *for φ.*

Back to Herbrand's proofs

Herbrand's theorem (Compositional Herbrand's theorem)

A 1^{st} order formula φ is valid iff there is a winning Σ-strategy: $\sigma : [\![\varphi]\!]$.

Proof:

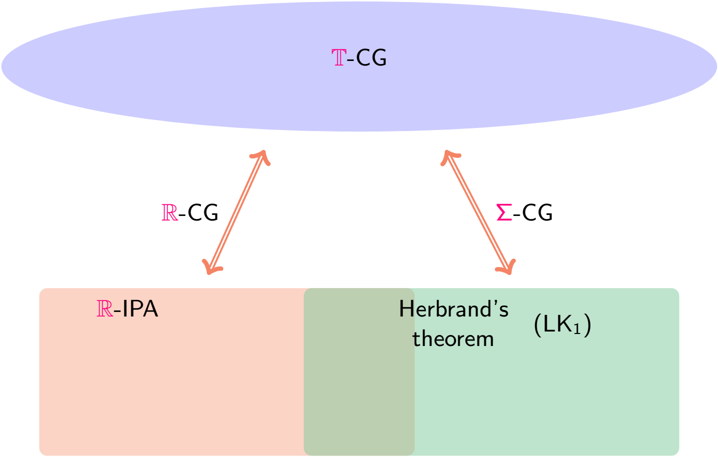
$\Rightarrow$ Interpret the classical sequent calculus LK_1.

$\Leftarrow$ Winning strategies ressemble expansion trees ...

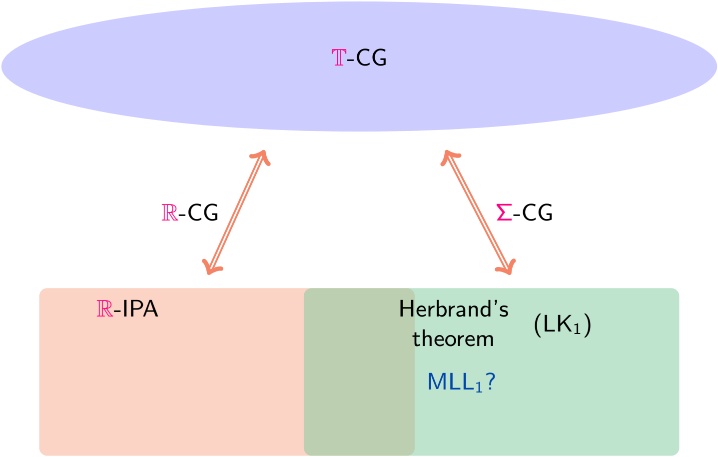
Lemma (Compactness)

From every winning strategy $\sigma : [\![\varphi]\!]$ one can **effectively extract a finite expansion tree** *for φ.*

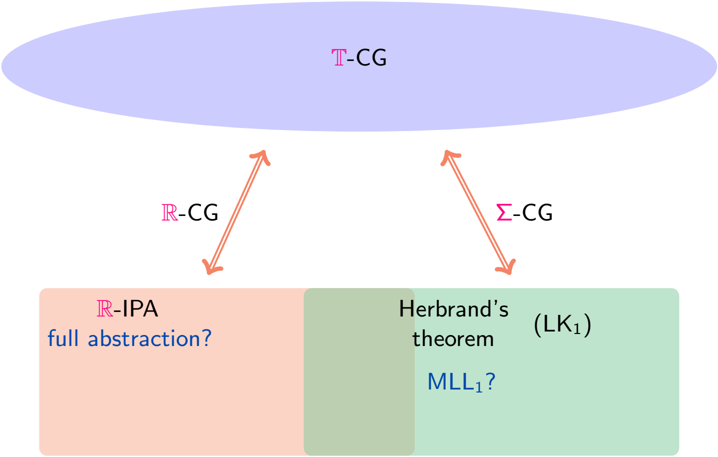
Conclusion



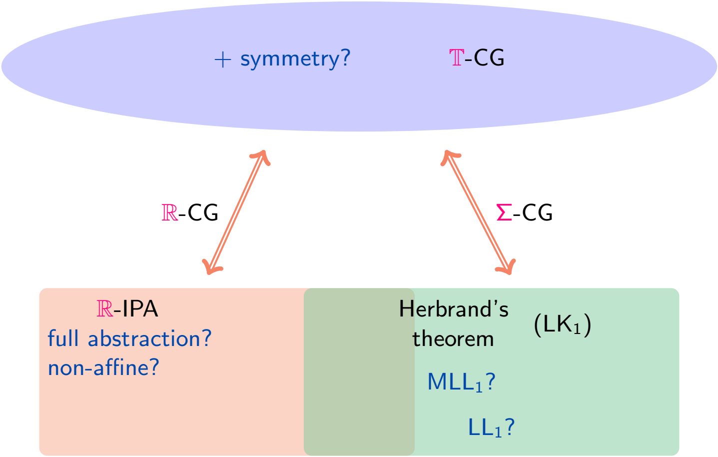
Perspectives



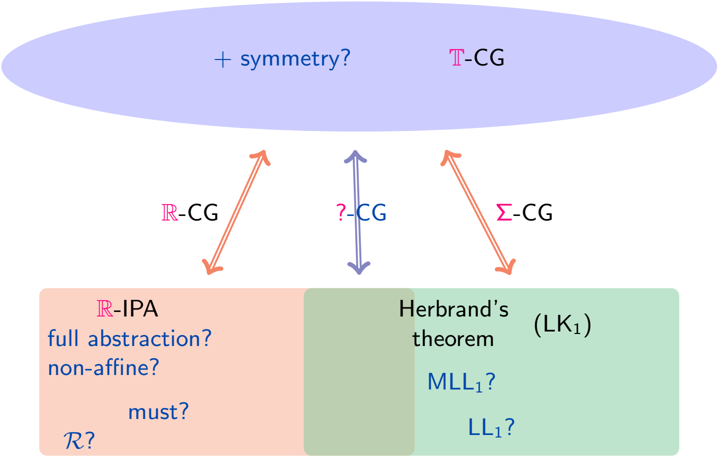
Perspectives



Perspectives



Perspectives



Perspectives