The parallel intensionally fully abstract games model of PCF

Simon Castellan[1], Pierre Clairambault[1], Glynn Winskel[2]

[1]LIP, ENS Lyon

[2]Computer Laboratory, University of Cambridge

May 12th 2016 – CHoCoLa

An implementation for `if`

How to describe the behaviour of **if** : $\mathbb{B} \to \mathbb{B} \to \mathbb{B} \to \mathbb{B}$?

An implementation for `if`

How to describe the behaviour of **if** : $\mathbb{B} \to \mathbb{B} \to \mathbb{B} \to \mathbb{B}$?

- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b.
- When the evaluation returns `ttrue`:
  - Evaluate t.
  - When the evaluation returns b_1:
    - Return b_1 to the caller.
- When the evaluation returns `false`:
  - Evaluate u.
  - When the evaluation returns b_2:
    - Return b_2 to the caller.

Alternation between *actions* and *information from the environment*.

A concurrent implementation for `if`

An alternative implementation for `if`:

A concurrent implementation for `if`

An alternative implementation for `if`:

- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b, t, u in parallel.
- When the evaluation of b returns `ttrue`,
  and when the evaluation of t returns b_1:
  - Return b_1 to the caller.
- When the evaluation of b returns `false`,
  and when the evaluation of u returns b_2:
  - Return b_2 to the caller.

Questions:

- Is there a context that can distinguish these implementations?

A concurrent implementation for `if`

An alternative implementation for `if`:

- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b, t, u in parallel.
- When the evaluation of b returns `ttrue`,
  and when the evaluation of t returns b_1:
  - Return b_1 to the caller.
- When the evaluation of b returns `false`,
  and when the evaluation of u returns b_2:
  - Return b_2 to the caller.

Questions:

- Is there a context that can distinguish these implementations?
  yes: $(\textbf{if}\ \texttt{tt}\ \textbf{then}\ ()\ \textbf{else}\ (x := 1)); !x$

A concurrent implementation for `if`

An alternative implementation for `if`:

- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b, t, u in parallel.
- When the evaluation of b returns `ttrue`,
  and when the evaluation of t returns b_1:
  - Return b_1 to the caller.
- When the evaluation of b returns `false`,
  and when the evaluation of u returns b_2:
  - Return b_2 to the caller.

Questions:

- Is there a context that can distinguish these implementations?
  yes: $(\textbf{if}\ \texttt{tt}\ \textbf{then}\ ()\ \textbf{else}\ (x := 1)); !x$
- Is there a pure context?

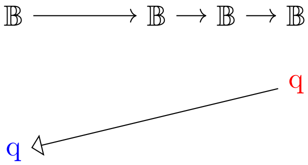
Formalizing implementations: P-view trees

$$\mathbb{B} \longrightarrow \mathbb{B} \to \mathbb{B} \to \mathbb{B}$$

$$\mathrm{q}$$

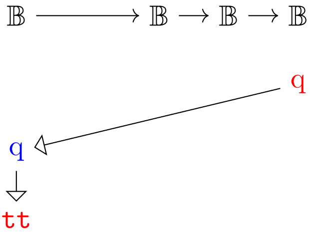
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:

Formalizing implementations: P-view trees



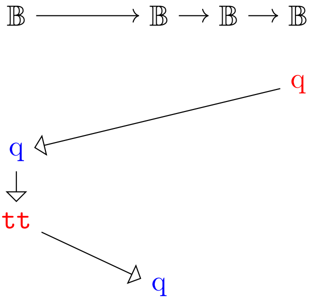
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b.

Formalizing implementations: P-view trees



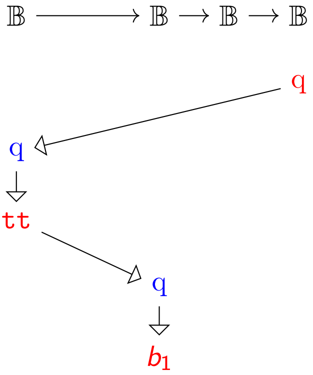
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b.
- When b evaluates to `ttrue`:

Formalizing implementations: P-view trees



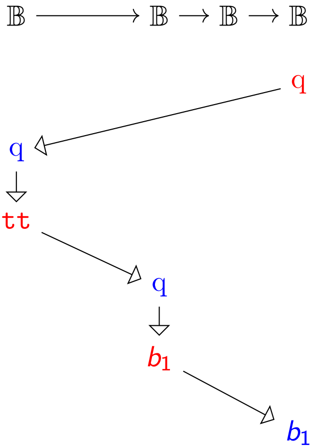
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b.
- When b evaluates to `ttrue`:
  - Evaluate t.

Formalizing implementations: P-view trees



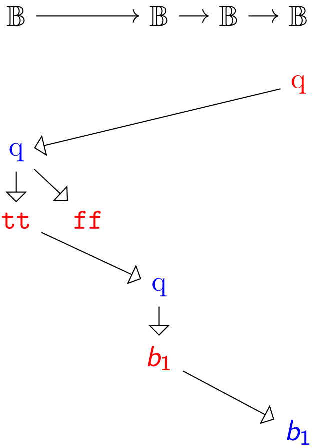
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b.
- When b evaluates to `ttrue`:
  - Evaluate t.
  - When t evaluates to b_1:

Formalizing implementations: P-view trees



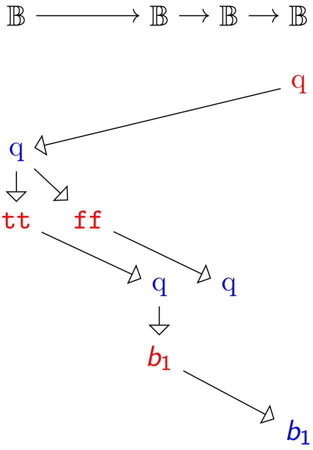
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b.
- When b evaluates to `ttrue`:
  - Evaluate t.
  - When t evaluates to b_1:
    - Return b_1 to the caller.

Formalizing implementations: P-view trees



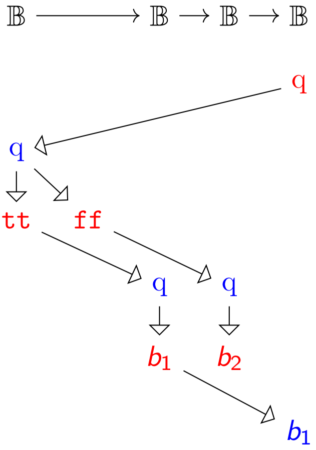
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b.
- When b evaluates to `ttrue`:
  - Evaluate t.
  - When t evaluates to b_1:
    - Return b_1 to the caller.
- When it evaluates to `false`:

Formalizing implementations: P-view trees



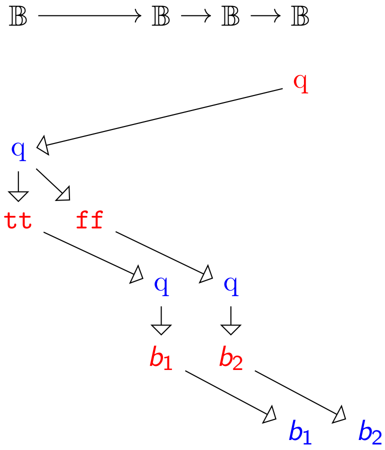
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b.
- When b evaluates to `ttrue`:
  - Evaluate t.
  - When t evaluates to b_1:
    - Return b_1 to the caller.
- When it evaluates to `false`:
  - Evaluate u.

Formalizing implementations: P-view trees



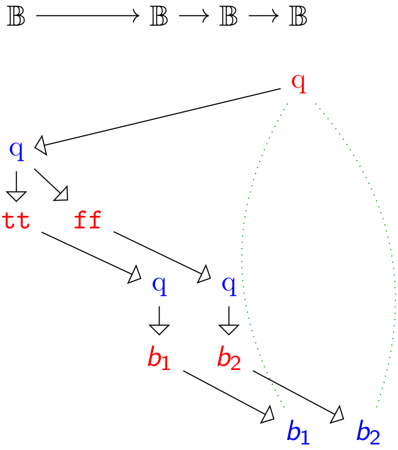
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b.
- When b evaluates to `ttrue`:
  - Evaluate t.
  - When t evaluates to b_1:
    - Return b_1 to the caller.
- When it evaluates to `false`:
  - Evaluate u.
  - When u evaluates to b_2:
    - Return b_2 to the caller.

Formalizing implementations: P-view trees



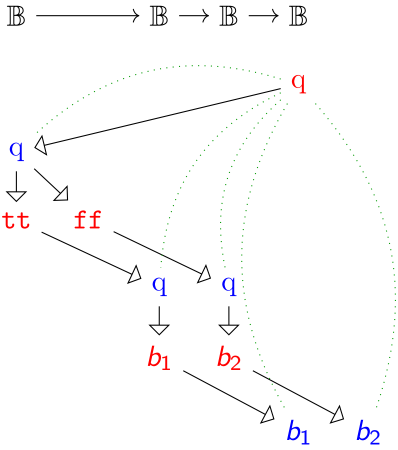
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b.
- When b evaluates to `ttrue`:
  - Evaluate t.
  - When t evaluates to b_1:
    - Return b_1 to the caller.
- When it evaluates to `false`:
  - Evaluate u.
  - When u evaluates to b_2:
    - Return b_2 to the caller.

Formalizing implementations: P-view trees



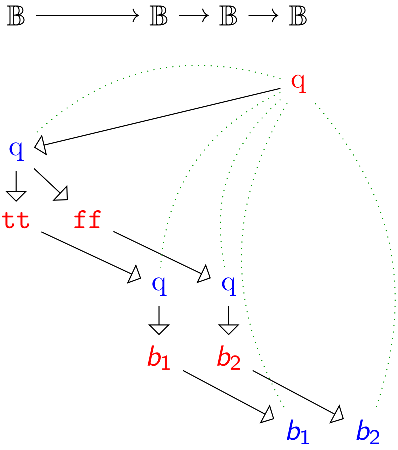
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b.
- When b evaluates to `ttrue`:
  - Evaluate t.
  - When t evaluates to b_1:
    - Return b_1 to the caller.
- When it evaluates to `false`:
  - Evaluate u.
  - When u evaluates to b_2:
    - Return b_2 to the caller.

Formalizing implementations: P-view trees



- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b.
- When b evaluates to `ttrue`:
  - Evaluate t.
  - When t evaluates to b_1:
    - Return b_1 to the caller.
- When it evaluates to `false`:
  - Evaluate u.
  - When u evaluates to b_2:
    - Return b_2 to the caller.

Formalizing implementations: P-view trees



- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b.
- When b evaluates to `ttrue`:
  - Evaluate t.
  - When t evaluates to b_1:
    - Return b_1 to the caller.
- When it evaluates to `false`:
  - Evaluate u.
  - When u evaluates to b_2:
    - Return b_2 to the caller.

Innocent Hyland-Ong game semantics: composition of P-view trees.

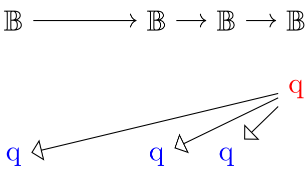
Towards P-view dags

$$\mathbb{B} \longrightarrow \mathbb{B} \to \mathbb{B} \to \mathbb{B}$$

$$\mathrm{q}$$

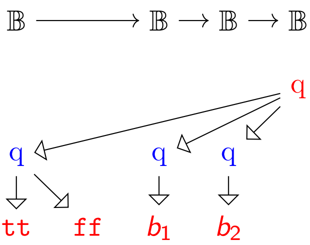
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:

Towards P-view dags



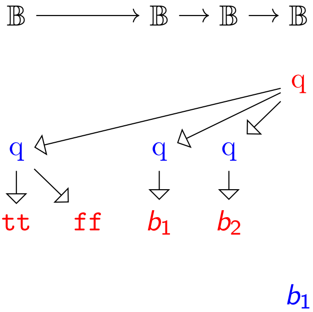
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b, t, u in parallel.

Towards P-view dags



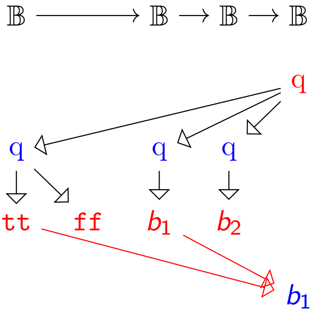
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b, t, u in parallel.

Towards P-view dags



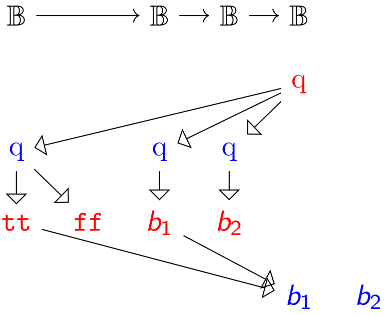
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b, t, u in parallel.
  - Return b_1 to the caller.

Towards P-view dags



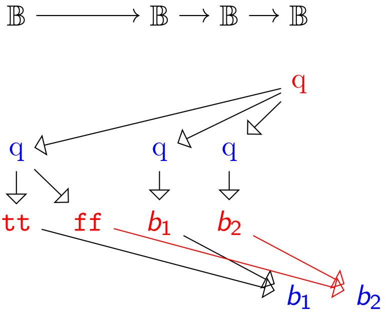
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b, t, u in parallel.
- When b evaluates to `ttrue`, **and** when t evaluates to b_1:
  - Return b_1 to the caller.

Towards P-view dags



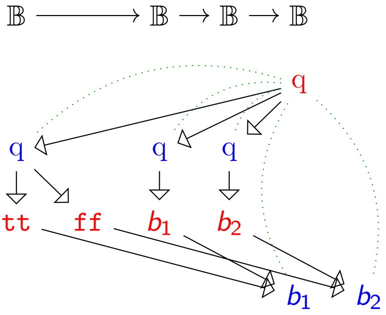
- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b, t, u in parallel.
- When b evaluates to `ttrue`, **and** when t evaluates to b_1:
  - Return b_1 to the caller.
  - Return b_2 to the caller.

Towards P-view dags



- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b, t, u in parallel.
- When b evaluates to `ttrue`, **and** when t evaluates to b_1:
  - Return b_1 to the caller.
- When b evaluates to `false`, **and** when u evaluates to b_2:
  - Return b_2 to the caller.

Towards P-view dags



- Given $b : \mathbb{B}$ and $t, u : \mathbb{B}$, do:
- Evaluate b, t, u in parallel.
- When b evaluates to `ttrue`, **and** when t evaluates to b_1:
  - Return b_1 to the caller.
- When b evaluates to `false`, **and** when u evaluates to b_2:
  - Return b_2 to the caller.

How to compose them?

A language of pure contexts: PCF

As a language describing pure contexts, we use **PCF**:

$$
\begin{aligned}
A, B &::= \mathbb{B} \mid \mathbb{N} \mid A \to B \\
t, u &::= \lambda x.\ t \mid t\ u \mid x \\
&\quad \mid \mathtt{tt}^{\mathbb{B}} \mid \mathtt{ff}^{\mathbb{B}} \mid \mathtt{if}^{\mathbb{B}\to\mathbb{B}\to\mathbb{B}} \\
&\quad \mid \bar{n}^{\mathbb{N}} \mid \mathtt{succ}^{\mathbb{N}\to\mathbb{N}} \mid \mathtt{pred}^{\mathbb{N}\to\mathbb{N}} \mid \mathtt{zero?}^{\mathbb{N}\to\mathbb{B}} \\
&\quad \mid Y^{(A\to A)\to A}
\end{aligned}
$$

Big-step semantics. $t \Downarrow k$ ($\vdash t : \mathbb{N}$, $k \in \mathbb{N}$)

This talk

1. Give a **compositional account** of "P-view dags".
   Composition of *linear terms* to focus on concurrency aspects.

2. Understand which dags arise from **pure terms**.
   Generalize the notion of innocence and well-bracketing.

3. Build a **model of PCF** using "P-view dags".
   Composing them by unfolding (something else than plays).

4. Prove **intensional full-abstraction**.
   Contextual equivalence coincide in the syntax and semantics.

I. Composing linear concurrent strategies

Representation of types: arenas

Arenas are semantic representations of types as partial orders:

Definition (Arenas)

An **arena** is a tuple $(A, \leq_A, \lambda : A \to \{O, P\} \times \{Q, A\})$ where:

- (*Alternation*) If $a \rightarrowtriangle_A a'$ then a and a' have different polarities
- (*Forest*) If $b, c \in A$ are below $a \in A$, they are comparable.
- (*Answers*) Answers are never minimal and always maximal.

$a \rightarrowtriangle_A a'$: $a < a'$ with no events in between ($\vdash_A$ in game semantics.)

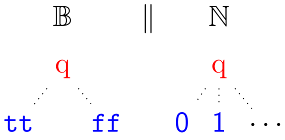
Examples:

$$\mathbb{B} \qquad\qquad \mathbb{N}$$

$$\begin{tikzcd} & \mathrm{q} \arrow[dl] \arrow[dr] & \\ \mathtt{tt} & & \mathtt{ff} \end{tikzcd} \qquad \begin{tikzcd} & \mathrm{q} \arrow[dl] \arrow[d] \arrow[dr] & \\ 0 & 1 & \cdots \end{tikzcd}$$

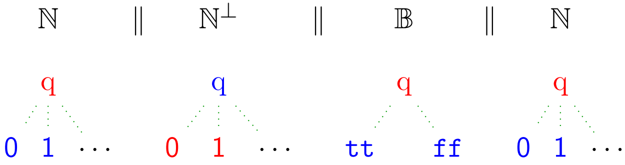
Constructions on arenas

- **Dual**: $A^\perp$ with the labelling λ^{OP} reversed.
- **Product**: $A \parallel B$ is obtained by putting A and B side-by-side:



- **Linear arrow:** $A \multimap B$ is defined as $A^\perp \parallel B$

Example: $\mathbb{N} \multimap ((\mathbb{N} \multimap \mathbb{B}) \parallel \mathbb{N}) = \mathbb{N} \parallel \mathbb{N}^\perp \parallel \mathbb{B} \parallel \mathbb{N}$.



Linear strategies

Strategy: causal enrichment of the arena with links $a \rightarrowtriangle b$.

Definition (Linear strategy)

A **linear strategy** on A is a partial order $(S, \leq_S)$ such that:

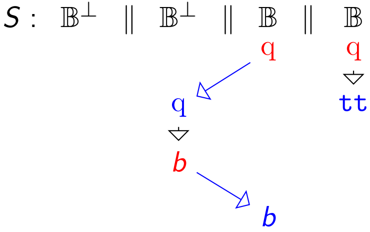
- (*Rule-obeying*) S is a down-closed subset of A and $\leq_A \subseteq \leq_S$
- (*Receptivity*) If $s \in S^+$ and $s \rightarrowtriangle a \in A$ then $a \in S$
- (*Courtesy*) If $s \rightarrowtriangle_S s'$ and not $s \rightarrowtriangle_A s'$ then s is negative and s' positive.

We write $S : A$.

The justifier of $s \in S$ is given by its predecessor for $\leq_A$.

Examples of linear strategies

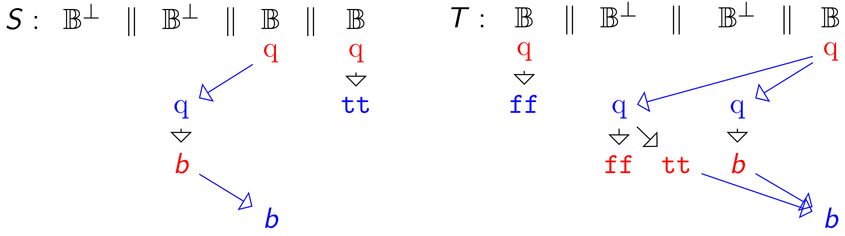
$$S = [\![\lambda b.((\lambda b'.b'), \mathtt{tt})]\!] \quad : \mathbb{B} \multimap ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B})$$
$$= \mathbb{B}^\perp \parallel \mathbb{B}^\perp \parallel \mathbb{B} \parallel \mathbb{B}$$



Examples of linear strategies

$$S = [\![\lambda b.((\lambda b'.b'), \mathtt{tt})]\!] \qquad : \mathbb{B} \multimap ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B})$$
$$= \mathbb{B}^\perp \parallel \mathbb{B}^\perp \parallel \mathbb{B} \parallel \mathbb{B}$$

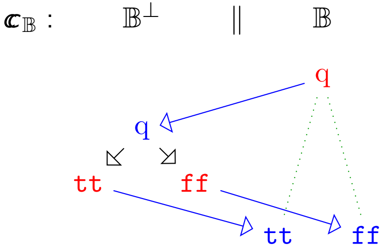
$$T = [\![\lambda(f, b).\mathtt{if}\ f\ \mathtt{ff}\ \mathtt{then}\ b\ \mathtt{else}\ \perp]\!] \qquad : ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \multimap \mathbb{B}$$
$$= \mathbb{B} \parallel \mathbb{B}^\perp \parallel \mathbb{B}^\perp \parallel \mathbb{B}$$



The copycat strategy

A particular linear strategy $A^{\perp} \parallel A$: **copycat**.

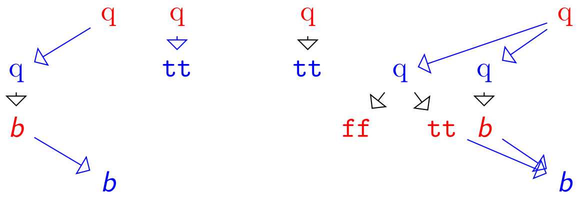
It delays a positive move by the corresponding negative move:



copycat will behave as identity wrt composition of strategies.

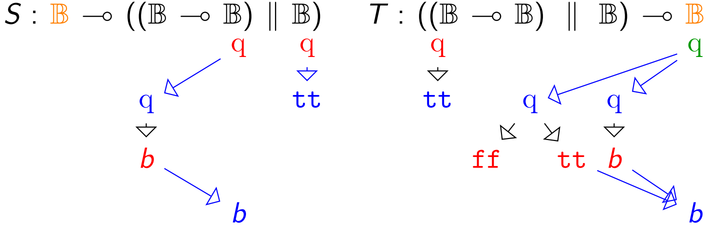
A glimpse on the composition process

$S : \mathbb{B} \multimap ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \qquad T : ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \multimap \mathbb{B}$



$S \circledast T : \mathbb{B} \multimap ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \multimap \mathbb{B}$

A glimpse on the composition process

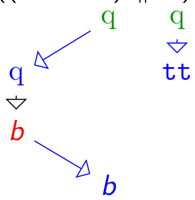


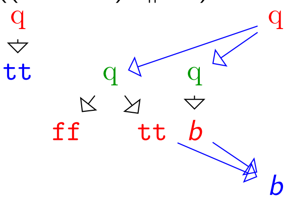
$$S \circledast T : \mathbb{B} \multimap ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \multimap \mathbb{B}$$

q

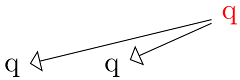
A glimpse on the composition process

$S : \mathbb{B} \multimap ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B})$ $\quad$ $T : ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \multimap \mathbb{B}$



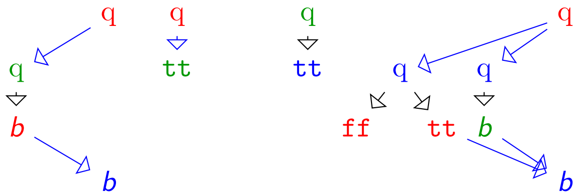


$S \circledast T : \mathbb{B} \multimap ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \multimap \mathbb{B}$

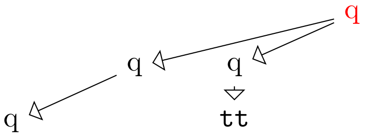


A glimpse on the composition process

$S : \mathbb{B} \multimap ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B})$ $\quad T : ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \multimap \mathbb{B}$

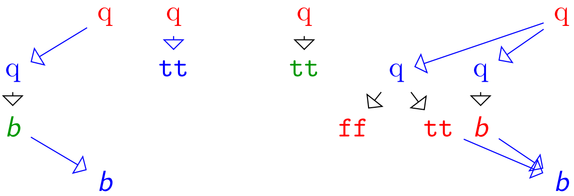


$S \circledast T : \mathbb{B} \multimap ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \multimap \mathbb{B}$

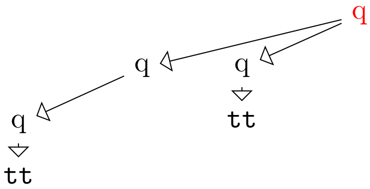


A glimpse on the composition process

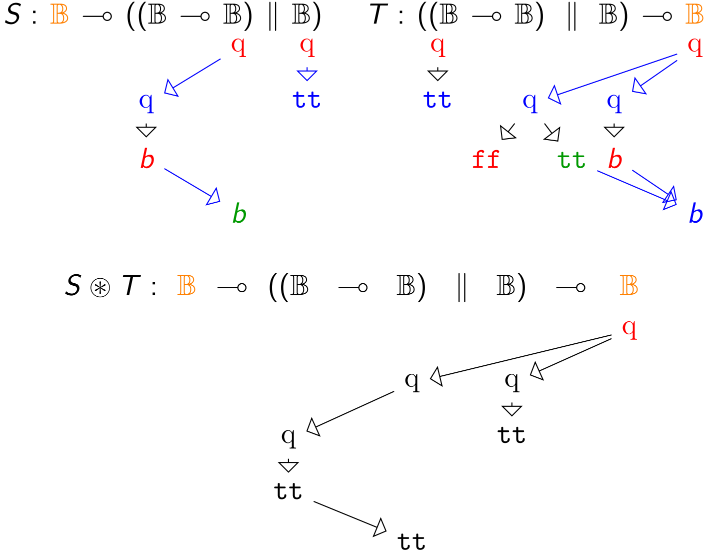
$S : \mathbb{B} \multimap ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \qquad T : ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \multimap \mathbb{B}$



$S \circledast T : \mathbb{B} \multimap ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \multimap \mathbb{B}$

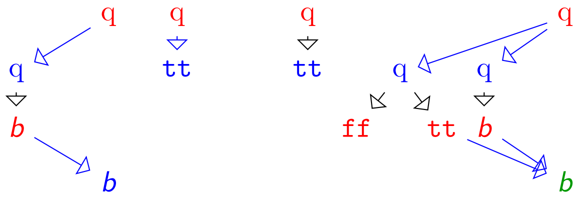


A glimpse on the composition process

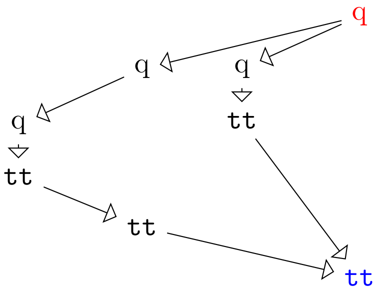


A glimpse on the composition process

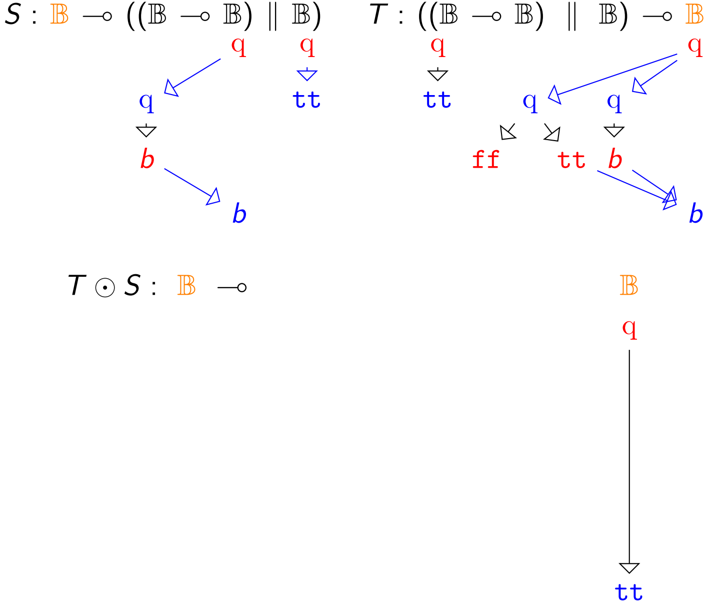
$S : \mathbb{B} \multimap ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \qquad T : ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \multimap \mathbb{B}$



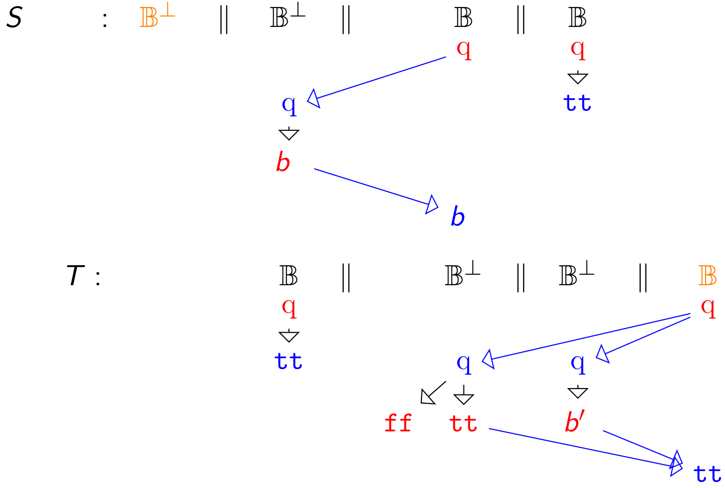
$S \circledast T : \mathbb{B} \multimap ((\mathbb{B} \multimap \mathbb{B}) \parallel \mathbb{B}) \multimap \mathbb{B}$



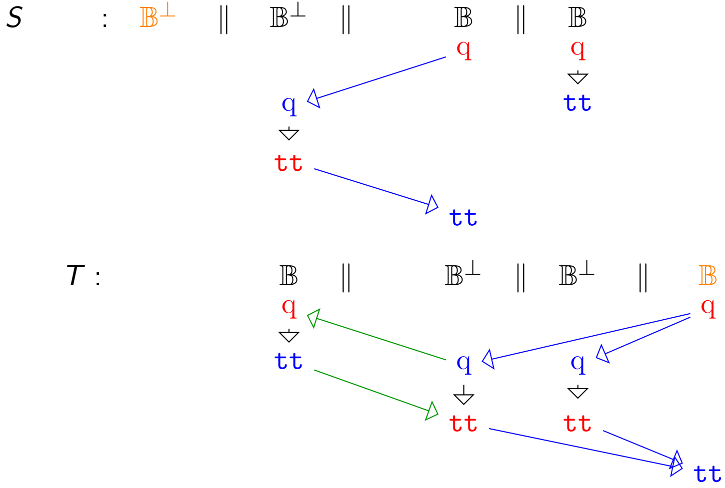
A glimpse on the composition process



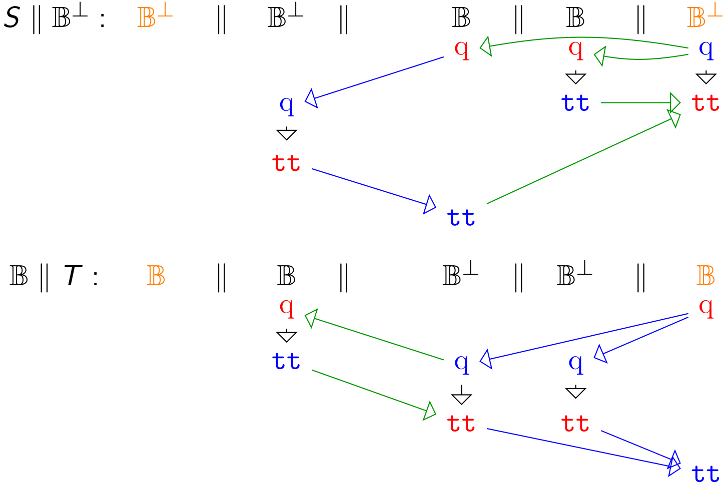
A global view on interaction



A global view on interaction



A global view on interaction



$\rightarrow$ Interaction of strategies on dual arenas.

Interaction

Assume S a linear strategy on A and T on $A^{\perp}$.

The pre-order I. We define the following pre-order I as follows:

- *Events*: $S \cap T$
- *Causality*: Transitive closure of $(\leq_S \cup \leq_T) \cap I^2$

Interaction

Assume S a linear strategy on A and T on $A^\perp$.

The pre-order I. We define the following pre-order I as follows:

- *Events*: $S \cap T$
- *Causality*: Transitive closure of $(\leq_S \cup \leq_T) \cap I^2$

Causal loops. I is not a partial-order in general. For example

$$A = \textbf{Drug} \quad \textbf{Money}$$

$$S : A = \textbf{Drug} \rightarrow \textbf{Money} \qquad T : A^\perp = \textbf{Drug} \leftarrow \textbf{Money}$$

I is $\textbf{Drug} \rightleftarrows \textbf{Money}$: there is a **causal loop**.

To remove those loops, we introduce the notion of *secured events*.

Secured events

An event $e \in I$ is **secured** when there exists $e_0, \ldots, e_n = e$ in I such that

$$\emptyset \subseteq \{e_0\} \subseteq \{e_0, e_1\} \subseteq \ldots \subseteq \{e_0, \ldots, e_n\} = \downarrow e$$

and all the sets are down-closed in I.

Lemma

The set of secured events of I along with the preorder induced by $\leq_I$ is a partial-order $S \wedge T$.

When all events are secured, the interaction is **deadlock-free**.

Composition of linear strategies

Let $S : A^\perp \parallel B$ and $T : B^\perp \parallel C$.

1. *Interaction:* we compute the interaction
$S \circledast T = (S \parallel C^\perp) \wedge (A \parallel T)$.
The events live in $A \parallel B \parallel C$.

2. *Hiding:* we define the linear strategy $T \odot S$ as follows:

Events	The events $e \in S \circledast T$ that live in A or C.
Causality	Induced by $\leq_{(S \parallel C^\perp) \circledast (A \parallel T)}$

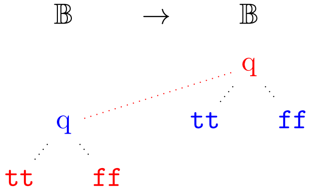
Theorem (Rideau-Winskel)

This composition is associative and linear strategies and arenas form a compact-closed category.

Negative category

PCF being call-by-name, types should be **negative arenas** (minimal events are negative)
But $\multimap$ does not preserve negativity!

Workaround: We use the usual arrow construction on arenas:



Under mild hypothesis on strategies:

$$\{S : A \multimap B\} \simeq \{S : A \rightarrow B\}$$

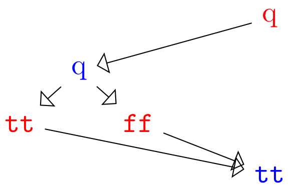
$\rightarrow$ Monoidal-closed category of negative arenas and "nice" strategies.

II. Innocent and well-bracketed strategies

Non-innocent behaviours

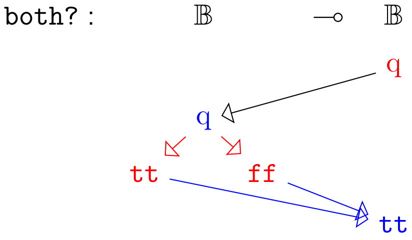
Which strategies arise as interpretation of pure (linear) terms?
For instance `both?` is not definable in a pure language:

`both?` : $\mathbb{B} \multimap \mathbb{B}$



Non-innocent behaviours

Which strategies arise as interpretation of pure (linear) terms?
For instance `both?` is not definable in a pure language:



Problem: Player is merging threads started by opponent.

A notion of concurrent innocence

A **thread** of a strategy $S : A$ is a sequence in S
$s_0 \rightarrowtriangle s_1 \rightarrowtriangle \ldots \rightarrowtriangle s_n$ with s_0 minimal.

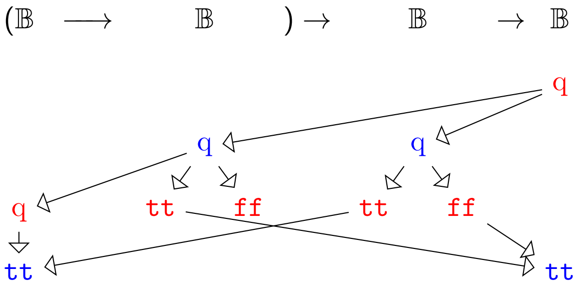
Definition

A linear strategy without the following pattern is **pre-innocent**:

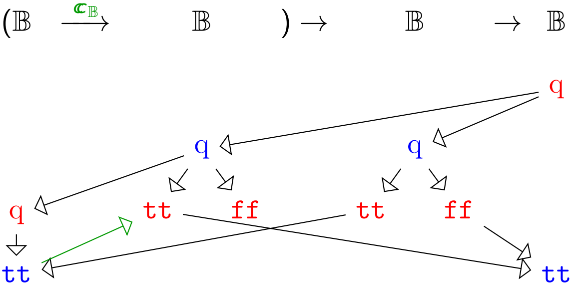
$$
\begin{tikzcd}
 & & & s_{i+1} \arrow[r] & \cdots \arrow[r] & s_{n-1} \arrow[dr] & \\
s_0 \arrow[r] & \cdots \arrow[r] & s_i \arrow[ur] \arrow[dr] & & & & s \\
 & & & s'_{i+1} \arrow[r] & \cdots \arrow[r] & s'_{p-1} \arrow[ur] &
\end{tikzcd}
$$

Player is only allowed to merge threads he started.

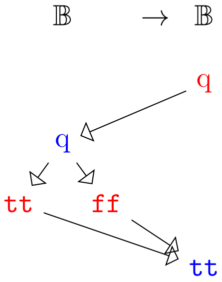
Non-stability by composition



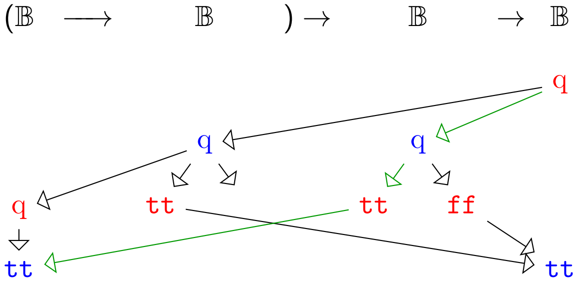
Non-stability by composition



Non-stability by composition



Non-stability by composition



This thread is not a linear strategy.

Visibility

To workaround that, we define:

Definition

- A strategy $S : A$ is **visible** when for every thread $s_0 \rightharpoonup \ldots \rightharpoonup s_n$, the set $\{s_0, \ldots, s_n\}$ is a linear strategy in A.
- A strategy is **innocent** when it is visible and pre-innocent.

These notions have interesting consequences:

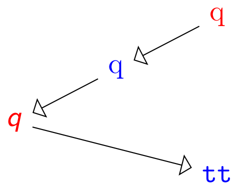
- Visibility and innocence are stable under composition
- Interactions of visible strategies are deadlock-free: $S \circledast T$ and $S \cap T$ coincide.
- Hence there is a functor from visible strategies to relations.

Non-well bracketed behaviours

There are still undefinable behaviours, related to questions/answers:

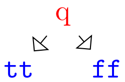
- *Not answering the pending question:*

$\texttt{strict?} : (\mathbb{B} \multimap \mathbb{B}) \multimap \mathbb{B}$



- *Answering twice the same question:*

$\texttt{fork} : \mathbb{B}$



Well-bracketed strategies

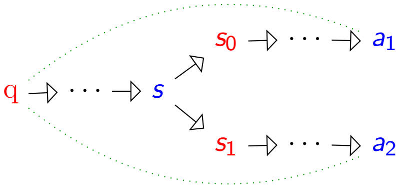
To ban these behaviours, we introduce **well-bracketing**:

Definition (Well-bracketing)

A strategy $S : A$ is **well-bracketed** when:

1. In any thread $s_0 \rightarrowtriangle \ldots \rightarrowtriangle s_n$ where s_n is an answer, its justifier is the last non-answered question.

2. If Player answers twice to the same question, they must live in threads started by Opponent.



A sub-category of innocent and well-bracketed strategies

Well-bracketing is stable under composition in presence of innocence. Hence:

Theorem

The following is a SMCC:

- ***Objects**:*
  Negative arenas
- ***Morphisms from** A **to** B:*
  Innocent, well-bracketed, linear strategies on $A \multimap B$ (or equivalently on $A \to B$)

Moreover, it is small enough:

Informal theorem

In this category, every strategy is definable by a PCF term up to observational equivalence.

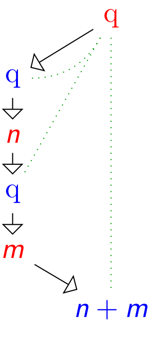
III. Expanded strategies

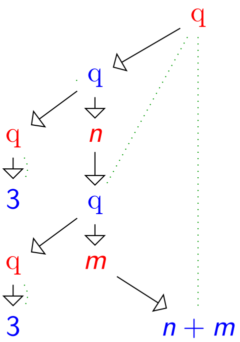
Tackling non-linearity

Linearity is built-in because our strategies are **subset** of arenas.
To relax that: ask only for a labelling map: $\mathrm{lbl} : S \to A$.

$[\![\lambda x^{\mathbb{N}}.x + x]\!]$ $\qquad$ $[\![\lambda f^{\mathbb{N}\to\mathbb{N}}.f\ 3 + f\ 3]\!]$.

$\mathbb{N} \longrightarrow \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \longrightarrow \mathbb{N}$





At higher-order, pointers become necessary to resolve ambiguity.

Can we compose such things?

$(\lambda x. x + x)$ $\qquad$ $(\lambda f. f\ 3 + f\ 3)$

$\mathbb{N} \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$

Can we compose such things?

$(\lambda x.x + x)$ $\qquad$ $(\lambda f.f\ 3 + f\ 3)$

$\mathbb{N} \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathrm{q}$

Can we compose such things?

$(\lambda x.x + x)$ $\quad$ $(\lambda f.f\ 3 + f\ 3)$

$\mathbb{N} \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$

q $\quad$ q

Can we compose such things?

$(\lambda x.x + x)$ $\qquad$ $(\lambda f.f\ 3 + f\ 3)$

$\mathbb{N} \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$

$$\begin{array}{cc} & \mathrm{q} \\ \mathrm{q} & \end{array} \qquad \mathrm{q}$$

Can we compose such things?

$(\lambda x.x + x)$ $\quad\quad$ $(\lambda f.f\ 3 + f\ 3)$

$\mathbb{N} \to \mathbb{N}$ $\quad\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\quad\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$

q $\quad$ q

q $\quad$ q

Can we compose such things?

$(\lambda x.x + x)$ $\qquad$ $(\lambda f.f\ 3 + f\ 3)$

$\mathbb{N} \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$

q $\qquad$ $\mathrm{q} \leftarrow \mathrm{q}$ $\qquad$ $\mathrm{q} \leftarrow \mathrm{q}$

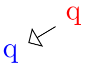
Can we compose such things?

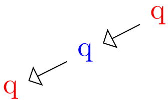
$(\lambda x.x + x)$	$(\lambda f.f\ 3 + f\ 3)$	
$\mathbb{N} \to \mathbb{N}$	$(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$	$(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$
$\mathrm{q} \leftarrow \mathrm{q}$	$\mathrm{q} \leftarrow \mathrm{q}$	$\mathrm{q} \leftarrow \mathrm{q}$

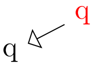
Can we compose such things?

$(\lambda x.x + x)$ $\quad$ $(\lambda f.f\ 3 + f\ 3)$

$\mathbb{N} \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



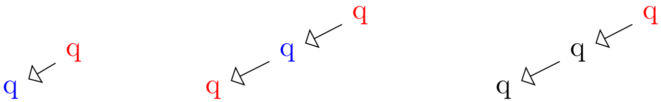




Can we compose such things?

$(\lambda x.x + x)$ $\quad$ $(\lambda f.f\ \color{green}{3} + f\ 3)$

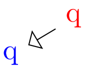
$\mathbb{N} \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$

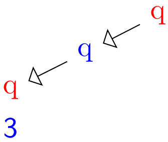


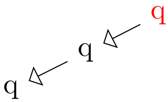
Can we compose such things?

$(\lambda x.x + x)$ $\quad$ $(\lambda f.f\ 3 + f\ 3)$

$\mathbb{N} \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



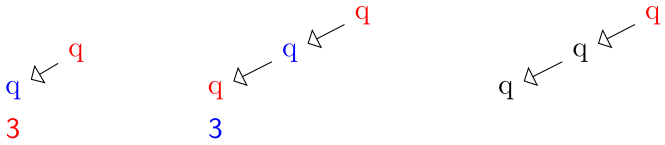




Can we compose such things?

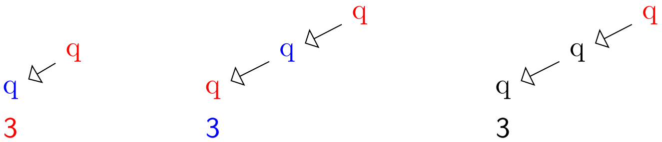
$(\lambda x.x + x)$ $\quad$ $(\lambda f.f\ 3 + f\ 3)$

$\mathbb{N} \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



Can we compose such things?

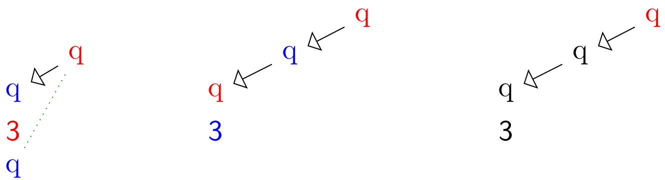
$(\lambda x.x + x)$	$(\lambda f.f\ 3 + f\ 3)$	
$\mathbb{N} \to \mathbb{N}$	$(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$	$(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



Can we compose such things?

$(\lambda x.x + x)$ $\quad$ $(\lambda f.f\ 3 + f\ 3)$

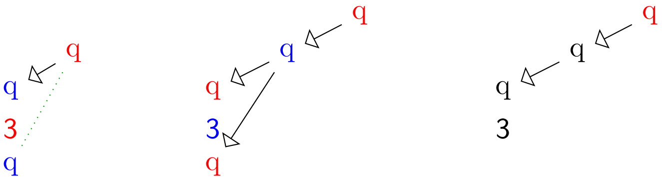
$\mathbb{N} \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



Can we compose such things?

$(\lambda x.x + x)$ $\quad$ $(\lambda f.f\ 3 + f\ 3)$

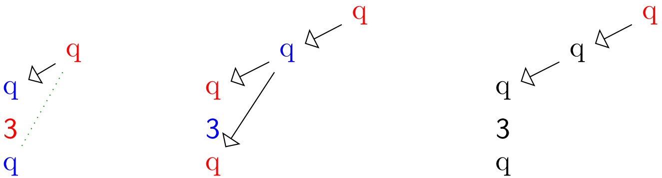
$\mathbb{N} \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



Can we compose such things?

$(\lambda x.x + x)$ $\qquad$ $(\lambda f.f\ 3 + f\ 3)$

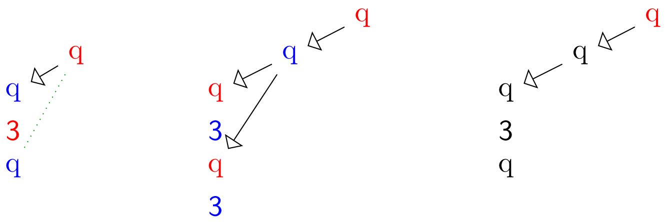
$\mathbb{N} \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



Can we compose such things?

$(\lambda x.x + x)$ $\quad$ $(\lambda f.f\ 3 + f\ 3)$

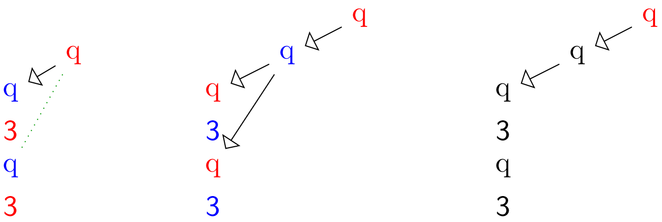
$\mathbb{N} \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



Can we compose such things?

$(\lambda x.x + x)$ $\qquad$ $(\lambda f.f\ 3 + f\ 3)$

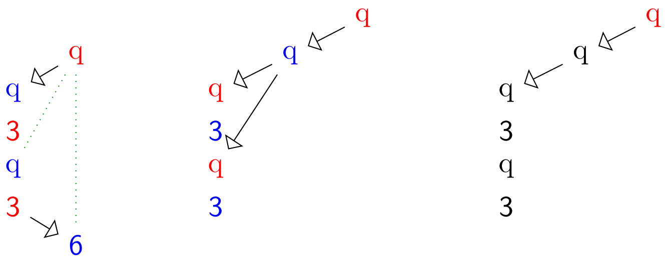
$\mathbb{N} \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



Can we compose such things?

$(\lambda x. x + x)$ $\qquad$ $(\lambda f. f\ 3 + f\ 3)$

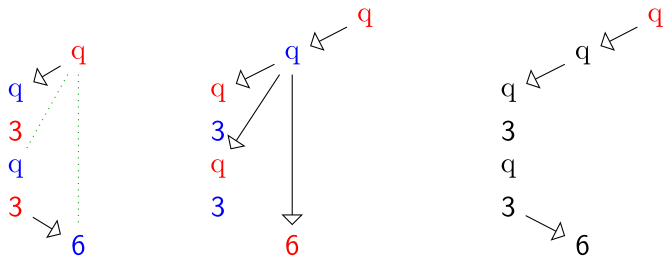
$\mathbb{N} \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



Can we compose such things?

$(\lambda x.x + x)$ $\quad$ $(\lambda f.f\ 3 + f\ 3)$

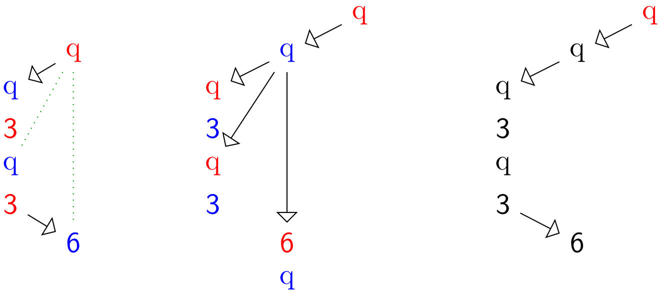
$\mathbb{N} \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



Can we compose such things?

$(\lambda x.x + x)$ $\quad$ $(\lambda f.f\ 3 + f\ 3)$

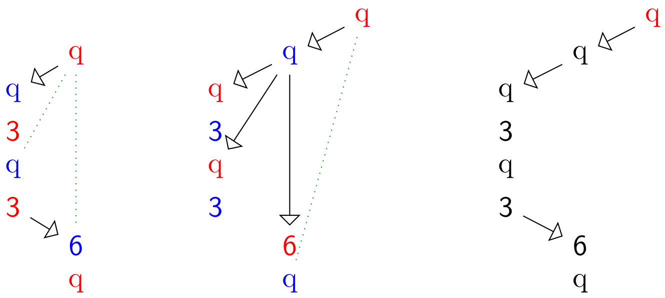
$\mathbb{N} \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



Can we compose such things?

$(\lambda x. x + x)$ $\quad$ $(\lambda f. f\ 3 + f\ 3)$

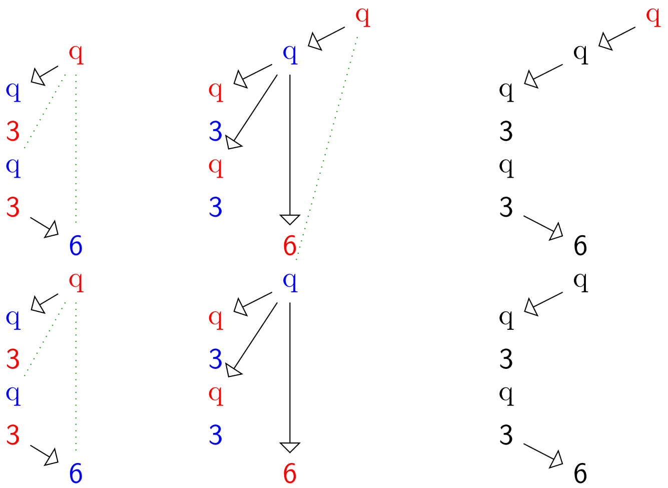
$\mathbb{N} \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\quad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



Can we compose such things?

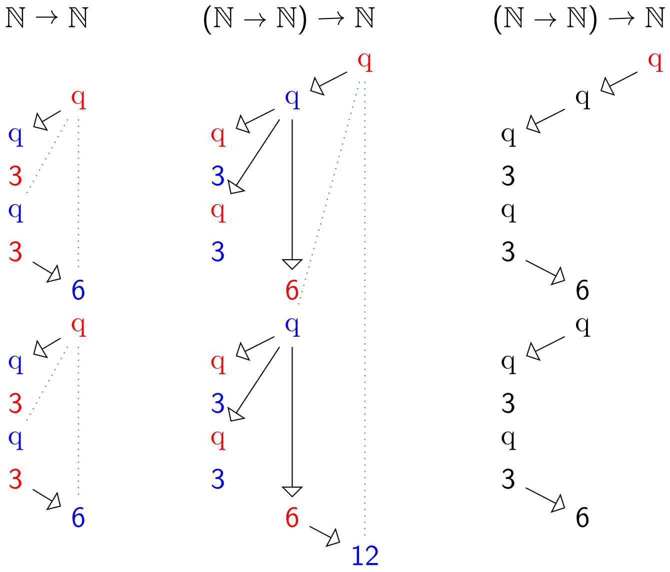
$(\lambda x.x + x)$ $\qquad$ $(\lambda f.f\ 3 + f\ 3)$

$\mathbb{N} \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



Can we compose such things?

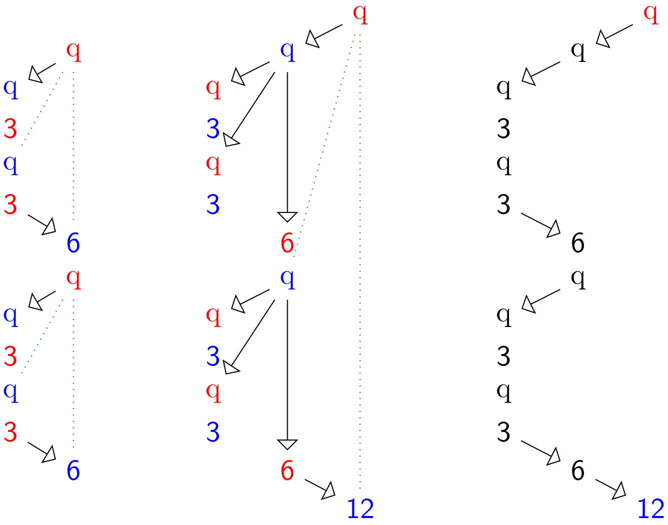
$(\lambda x.x + x)$ $\qquad$ $(\lambda f.f\ 3 + f\ 3)$



Can we compose such things?

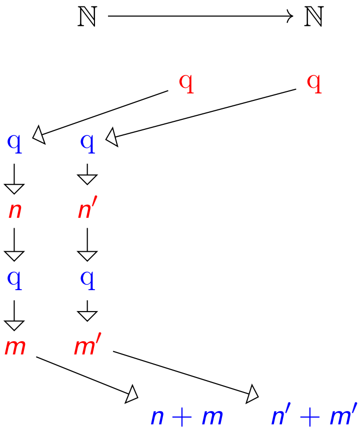
$(\lambda x.x + x)$ $\qquad$ $(\lambda f.f\ 3 + f\ 3)$

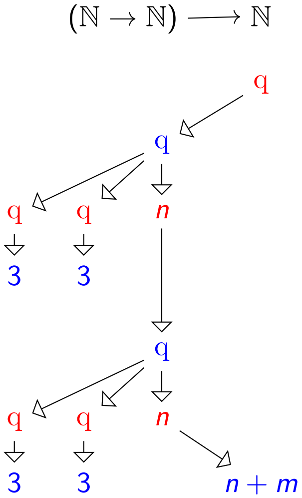
$\mathbb{N} \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$ $\qquad$ $(\mathbb{N} \to \mathbb{N}) \to \mathbb{N}$



The need for expanded strategies

It is as if we composed the following linear strategies:





On which arena do they live?

The expanded arena

Idea: deeply duplicate moves to accommodate non-linearity.

Definition (Expanded arena)

Let A be an arena. We define the arena $!A$ as follows:

- (**Events**) *index functions*: pairs (a, α): $a \in A$ and $\alpha :\downarrow a \to \mathbb{N}$
- (**Causality**) $(a, \alpha) \leq (b, \beta)$ iff $a \leq b$ and $\alpha \subseteq \beta$.
- (**Labelling**) Inherited from A

where $\downarrow a = \{a' \in A \mid a' \leq a\}$.

Down-closed subsets of $!A$ correspond to Boudes' *thick sub-trees*.

Example of expanded arena

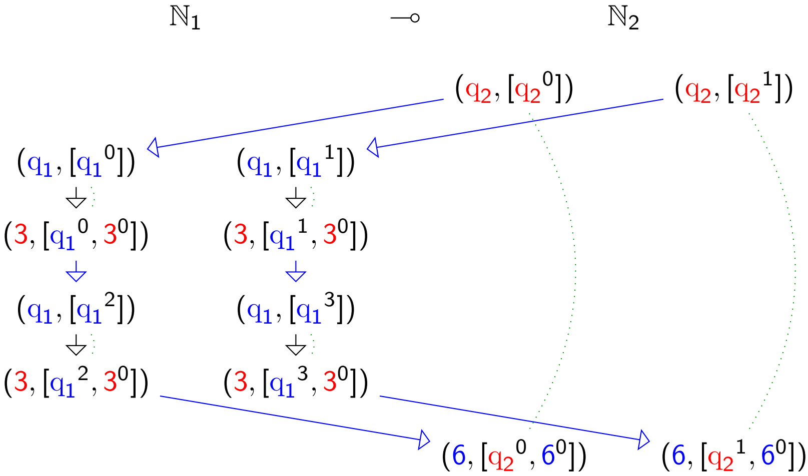
Take $A = \mathbb{N} \multimap \mathbb{N}$. Here is a down-closed subset of $!A$ corresponding to the previous example:

$$
\begin{tikzcd}
\mathbb{N}_1 & & \multimap & \mathbb{N}_2 \\
 & & (q_2, [{q_2}^0]) \arrow[dll] & (q_2, [{q_2}^1]) \arrow[dll] \\
(q_1, [{q_1}^0]) \arrow[d] & (q_1, [{q_1}^1]) \arrow[d] & & \\
(3, [{q_1}^0, 3^0]) \arrow[d] & (3, [{q_1}^1, 3^0]) \arrow[d] & & \\
(q_1, [{q_1}^2]) \arrow[d] & (q_1, [{q_1}^3]) \arrow[d] & & \\
(3, [{q_1}^2, 3^0]) \arrow[drr] & (3, [{q_1}^3, 3^0]) \arrow[drr] & & \\
 & & (6, [{q_2}^0, 6^0]) & (6, [{q_2}^1, 6^0])
\end{tikzcd}
$$

Non-linear strategies on A can be seen as linear strategies on $!A$.

Example of expanded arena

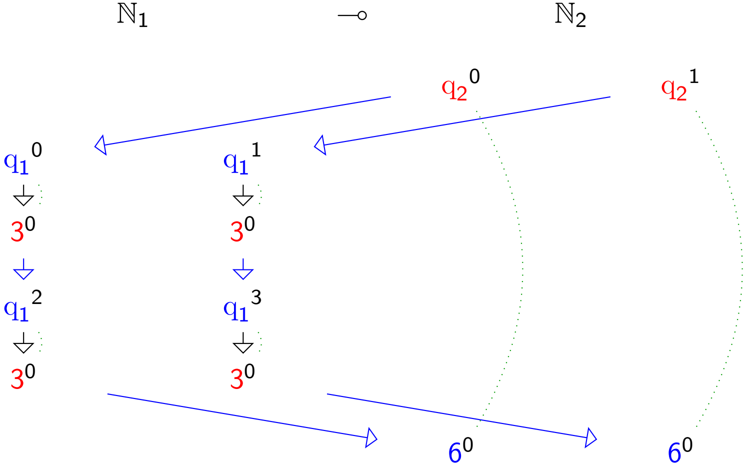
Take $A = \mathbb{N} \multimap \mathbb{N}$. Here is a down-closed subset of $!A$ corresponding to the previous example:



Non-linear strategies on A can be seen as linear strategies on $!A$.

Example of expanded arena

Take $A = \mathbb{N} \multimap \mathbb{N}$. Here is a down-closed subset of $!A$ corresponding to the previous example:



Non-linear strategies on A can be seen as linear strategies on $!A$.

Non-uniformity

In a strategy $S : !A$, positive copy indices are arbitrary.
$\rightarrow$ Consider strategies on $!A$ up to positive copy indices ($S \simeq T$)

Problem. $\simeq$ is not a congruence:

$$\mathbb{N}$$

$$\begin{array}{ccc} q^0 & q^1 & \cdots \\ \downarrow & \downarrow & \\ 0^0 & 1^0 & \cdots \end{array}$$

This strategy is not **uniform**: moves depend on negative indices.

A CCC of innocent strategies

There is a notion of uniform strategies on $!A$ that forbids this.

Theorem (C., Clairambault, Winskel)

The following is a cartesian closed category Inn supporting an interpretation of PCF:

- *(Objects) Negative arenas*
- *(Morphisms from A to B) **Uniform**, **innocent**, **well-bracketed** linear strategies from $!A$ to $!B$, **up to copy indices**.*

There is a bigger CCC of uniform and **single-threaded** strategies.

IV. Finite definability and intensional full-abstraction for PCF

Contextual equivalences

Contextual equivalence formalizes program indistinguishability:

- PCF terms: $t \simeq_{syn} u$ iff for all context $C[] : \mathbb{N}$,

$$C[t] \Downarrow k \text{ iff } C[u] \Downarrow k$$

- Strategies: $S \simeq_{sem} S' : A$ iff for all strategies $T : !A \multimap \mathbb{N}$,

$$k \in T \odot S \text{ iff } k \in T \odot S'$$

Intensional full abstraction: $t \simeq_{syn} u$ iff $[\![t]\!] \simeq_{sem} [\![u]\!]$.

Extensional behaviour of strategies

PCF terms can be seen as continuous functions. And strategies?

- **Base types.** Strategies on $\mathbb{B}$ diverge or give a single answer. Hence:

$$\downarrow : \mathrm{Inn}(\mathbb{B}) \cong \mathbb{B}_\perp : \uparrow$$

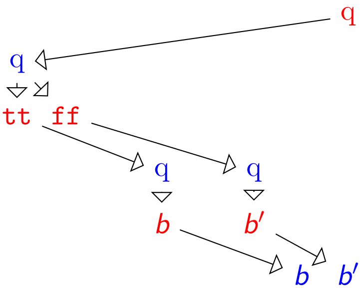
- **First-order types.** The previous isomorphism lifts to a map:

$$\begin{array}{llcr} \downarrow : \mathrm{Inn}(\mathbb{B}^k, \mathbb{B}) & \to [\ \mathbb{B}_\perp^k & \to & \mathbb{B}_\perp\ ] \\ S & \mapsto (b_1, \ldots, b_n) & \mapsto & \downarrow (S \odot \langle \uparrow b_1, \ldots, \uparrow b_n \rangle) \end{array}$$

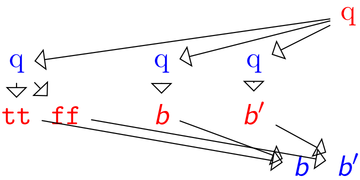
Intensional full-abstraction entails: $\downarrow S = \downarrow S'$ iff $S \simeq_{sem} S'$.

The two ifs are contextual equivalent

$\texttt{sif} : \mathbb{B} \multimap \mathbb{B} \multimap \mathbb{B} \multimap \mathbb{B}$



$\texttt{pif} : \mathbb{B} \multimap \mathbb{B} \multimap \mathbb{B} \multimap \mathbb{B}$



Both implement the same continuous function:

$$\begin{aligned} \mathbb{B} \times \mathbb{B} \times \mathbb{B} &\to \mathbb{B} \\ (\texttt{tt}, x, \bot) &\mapsto x \\ (\texttt{ff}, \bot, x) &\mapsto x \\ (\bot, \bot, \bot) &\mapsto \bot \end{aligned}$$

Outline of the proof of intensional full-abstraction

Theorem (C., Clairambault, Winskel)

Our concurrent interpretation of PCF inside Inn is intensionally fully-abstract.

Proof.

1. The model is sound and adequate: $t \Downarrow k$ if and only if $k \in [\![t]\!]$.
2. Hence $[\![t]\!] \simeq_{sem} [\![u]\!]$ implies $t \simeq_{syn} u$.
3. If $[\![t]\!] \not\simeq_{sem} [\![u]\!]$, they are distinguished by a *finite* strategy S.
4. Finite strategies can be represented by a PCF term up to contextual equivalence:
   - We prove the result for first-order types
   - And generalize by induction to higher-order types.
5. Hence S gives a context C dinstinguishing t and u
6. And $t \not\simeq_{syn} u$.

□

An aside on finite strategies

Strategies on $!A$ are infinite but uniformity ensues its behaviour depends on a (possibly) finite part:

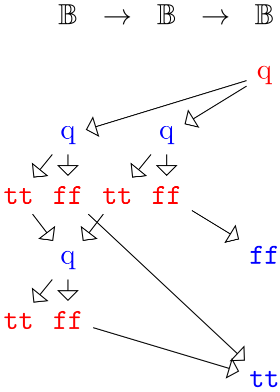
Definition (Reduced form)

Let $S : !A$ be a strategy. Define $r(S)$ to be the induced partial-order on $\{s \in S \mid \forall s_0^- \leq s, s_0 \text{ has copy index } 0\}$

- Reduction is **faithful** ensues that $r(S) \cong r(S')$ implies $S \simeq S'$ (Uniformity)
- A strategy is **finite** when $r(S)$ is.
- $r(S)$ is exactly the P-view dag of S.

Finite definability for the first-order

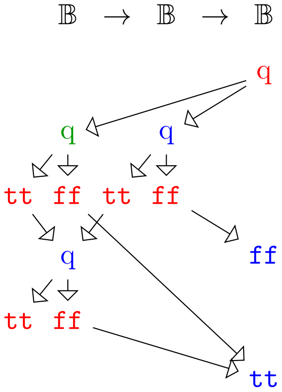
A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:



`t = λxy.`

Finite definability for the first-order

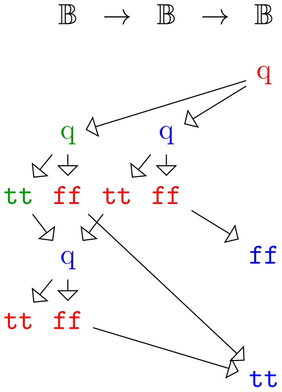
A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:




```
t = λxy.
 if x
```

Finite definability for the first-order

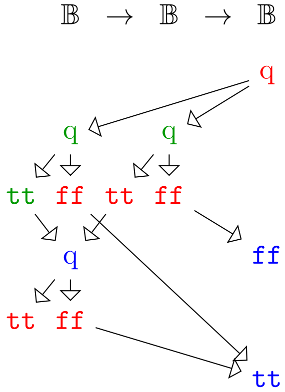
A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:




```
t = λxy.
 if x then
```

Finite definability for the first-order

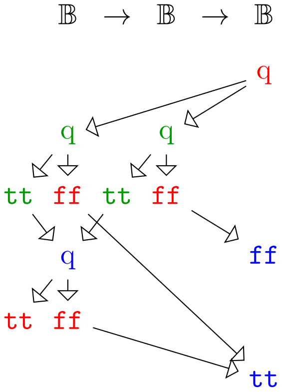
A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:




```
t = λxy.
 if x then if y
```

Finite definability for the first-order

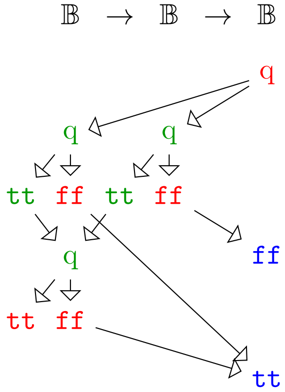
A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:




```
t = λxy.
 if x then if y then
```

Finite definability for the first-order

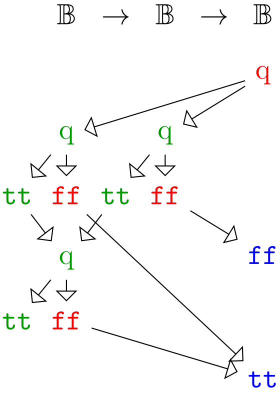
A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:




```
t = λxy.
 if x then if y then if x
```

Finite definability for the first-order

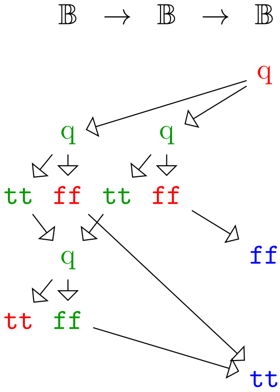
A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:




```
t = λxy.
 if x then if y then if x then ⊥
```

Finite definability for the first-order

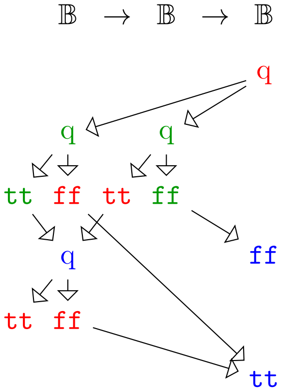
A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:



t = λxy.
 if x then if y then if x then $\perp$ else $\perp$

Finite definability for the first-order

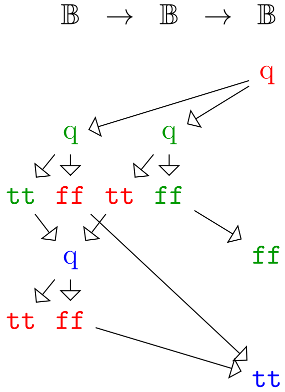
A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:



t = λxy.
 if x then if y then if x then $\perp$ else $\perp$
 else

Finite definability for the first-order

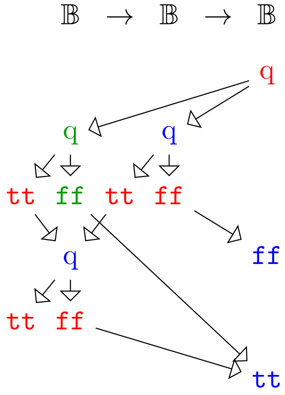
A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:




```
t = λxy.
 if x then if y then if x then ⊥ else ⊥
           else ff
```

Finite definability for the first-order

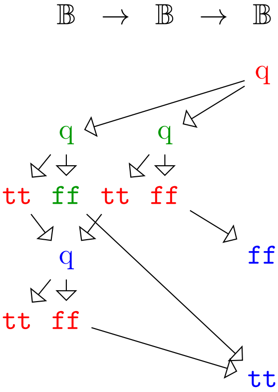
A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:




```
t = λxy.
 if x then if y then if x then ⊥ else ⊥
          else ff
     else
```

Finite definability for the first-order

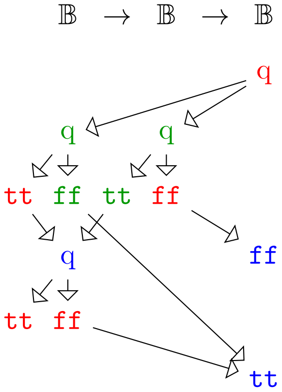
A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:




```
t = λxy.
 if x then if y then if x then ⊥ else ⊥
          else ff
     else if y
```

Finite definability for the first-order

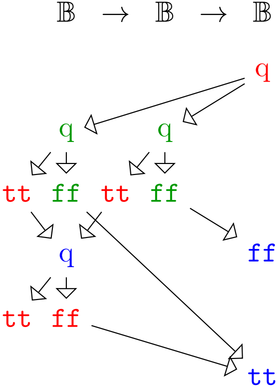
A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:




```
t = λxy.
 if x then if y then if x then ⊥ else ⊥
           else ff
      else if y then ⊥
```

Finite definability for the first-order

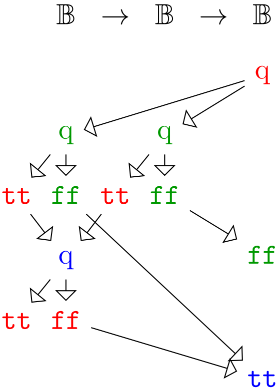
A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:



t = λxy.
 if x then if y then if x then $\perp$ else $\perp$
           else ff
      else if y then $\perp$ else

Finite definability for the first-order

A typical strategy on $!(\mathbb{B} \to \mathbb{B} \to \mathbb{B})$ looks like:




```
t = λxy.
 if x then if y then if x then ⊥ else ⊥
          else ff
      else if y then ⊥ else ff
```

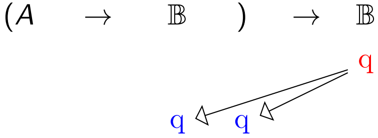
Finite definability at higher-order

What can happen on $(A \to \mathbb{B}) \to \mathbb{B}$?

$$(A \quad \to \quad \mathbb{B} \quad) \quad \to \quad \underset{\color{red}{\mathrm{q}}}{\mathbb{B}}$$

Finite definability at higher-order

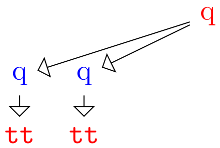
What can happen on $(A \to \mathbb{B}) \to \mathbb{B}$?



Finite definability at higher-order

What can happen on $(A \to \mathbb{B}) \to \mathbb{B}$?

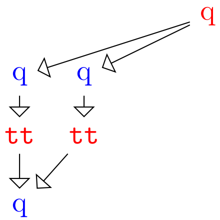
$$(A \quad \to \quad \mathbb{B} \quad) \quad \to \quad \mathbb{B}$$



Finite definability at higher-order

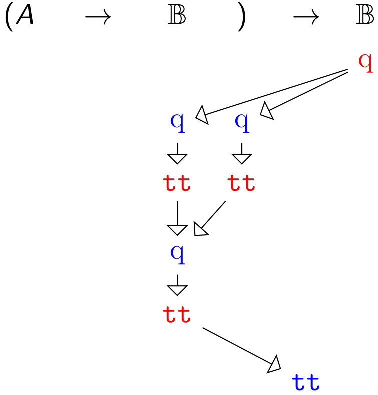
What can happen on $(A \to \mathbb{B}) \to \mathbb{B}$?

$$(A \quad \to \quad \mathbb{B} \quad) \quad \to \quad \mathbb{B}$$



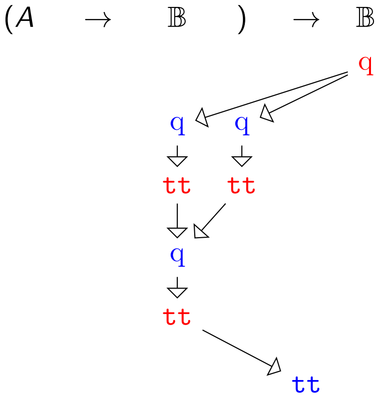
Finite definability at higher-order

What can happen on $(A \to \mathbb{B}) \to \mathbb{B}$?



Finite definability at higher-order

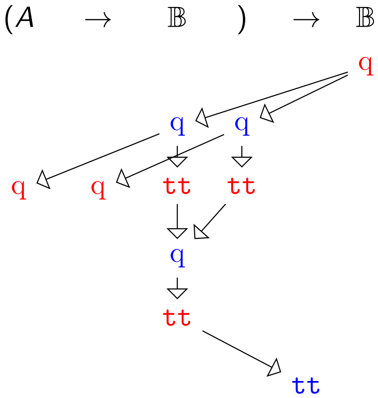
What can happen on $(A \to \mathbb{B}) \to \mathbb{B}$?



- This gives a strategy $S_f : !(\mathbb{B} \times \mathbb{B} \times \mathbb{B} \to \mathbb{B})$.

Finite definability at higher-order

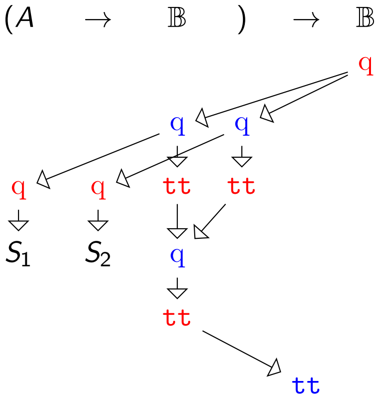
What can happen on $(A \to \mathbb{B}) \to \mathbb{B}$?



- This gives a strategy $S_f : !(\mathbb{B} \times \mathbb{B} \times \mathbb{B} \to \mathbb{B})$.

Finite definability at higher-order

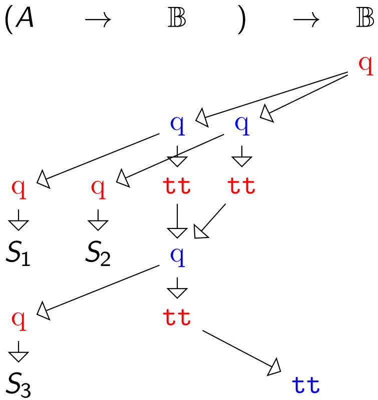
What can happen on $(A \to \mathbb{B}) \to \mathbb{B}$?



- This gives a strategy $S_f : !(\mathbb{B} \times \mathbb{B} \times \mathbb{B} \to \mathbb{B})$.

Finite definability at higher-order

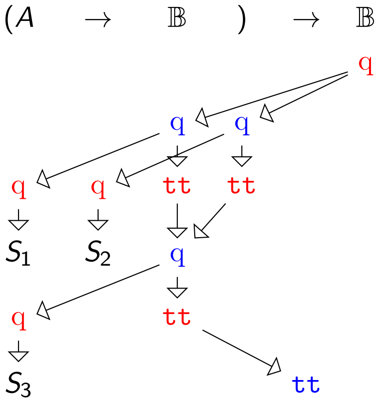
What can happen on $(A \to \mathbb{B}) \to \mathbb{B}$?



- This gives a strategy $S_f : !(\mathbb{B} \times \mathbb{B} \times \mathbb{B} \to \mathbb{B})$.

Finite definability at higher-order

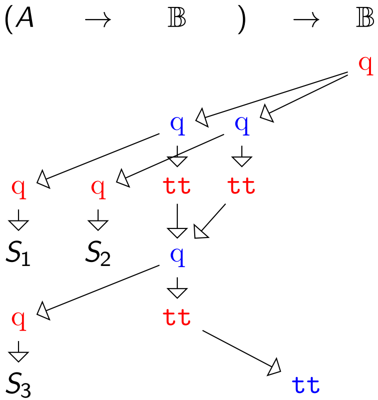
What can happen on $(A \to \mathbb{B}) \to \mathbb{B}$?



- This gives a strategy $S_f : !(\mathbb{B} \times \mathbb{B} \times \mathbb{B} \to \mathbb{B})$.
- This gives strategies $S_i : !((A \to \mathbb{B}) \to A)$.

Finite definability at higher-order

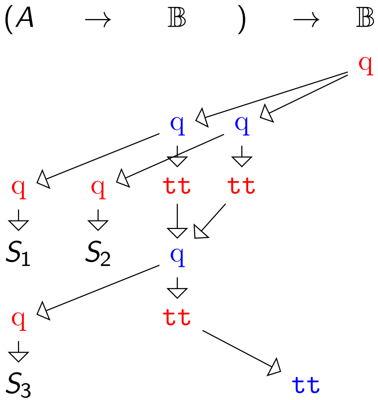
What can happen on $(A \to \mathbb{B}) \to \mathbb{B}$?



- This gives a strategy $S_f : !(\mathbb{B} \times \mathbb{B} \times \mathbb{B} \to \mathbb{B})$.
- This gives strategies $S_i : !((A \to \mathbb{B}) \to A)$.
- **Theorem.** $S = \lambda g^{A \to \mathbb{B}}.\ S_f\ (S_1\ g)\ (S_2\ g)\ (S_3\ g)$

Finite definability at higher-order

What can happen on $(A \to \mathbb{B}) \to \mathbb{B}$?



- This gives a strategy $S_f : !(\mathbb{B} \times \mathbb{B} \times \mathbb{B} \to \mathbb{B})$.
- This gives strategies $S_i : !((A \to \mathbb{B}) \to A)$.
- **Theorem.** $S = \lambda g^{A \to \mathbb{B}}.\ S_f\ (S_1\ g)\ (S_2\ g)\ (S_3\ g)$
- The S_i are smaller and S_f is first-order.

Conclusion

Summary.

- A compositional framework for concurrent strategies
- A CCC of innocent strategies supporting a concurrent interpretation PCF.
- A result of intensional full-abstraction.

Existing extensions / Work in progress.

- **Disjunctive causes.** There is a fully-abstract model for PCF+por (collapse to domains)
- **Non-determinism.** There is a CCC of concurrent non-deterministic strategies supporting languages like IPA.
- **Must-equivalence.** Hiding can be modified to remember divergence and obtain a model for must-equivalences.
- **Weak memory models.** Design models of shared memory concurrency that features weak memory behaviours.