Representing permutations without permutations

The expressive power of sequential intersection

Pierre VIAL
Équipe Gallinette
Inria (LS2N CNRS)

June 10, 2018

Outline

Context

Non-idempotent intersection types

Using type A *once* or *twice* not the same

Rigid vs. non-rigid paradigms

Proof red. deterministic vs. non-deterministic

Question 1

Rigid collapses on non-rigid

Is this collapse surjective?

(A, A, B) and (A, B, A) collapse on $[A, A, B]$

Question 2

In rigid fw., red. paths captured by permutations

$(A, B, A) \mapsto (A, A, B)$

Is it possible to capture red. paths without perm. ?

All this in a coinductive fw. (no productivity)!

Resource calculi (intuitions)

Girard (87), Boudol (93), Kfoury (96), Ehrhard-Régnier (03)

- **Bag arguments**: $t\,[u_1,\ldots,u_n]$ (and not $t\,u$)
- **Linear** substitution and reduction:
  if $t = x\,[x,y]$ then $x \rightsquigarrow [u_1,u_2]$ gives $u_1\,[u_2,y]$ or $u_2\,[u_1,y]$.
- Taylor expansion of a λ-term (linearization):
  TE of $t\,u$ = formal series involving $\tilde{t}[\,]$, $\tilde{t}[\tilde{u}]$, $\tilde{t}[\tilde{u},\tilde{u}]$, $\tilde{t}[\tilde{u},\tilde{u},\tilde{u}]\ldots$
- Adequation: Böhm tree and Taylor expansion.

Tsukada, Ong, Asada (LiCS17 and LiCS18)
$\rightsquigarrow$ "compositional enumeration problem"

- **Rigid** bags: $t\,(u_1,\ldots,u_n)$
- **Isomorphisms** to identify equivalent bags.
- Deterministic reduction.
- Adequation.

Plan

Terminal states and execution/reduction strategies

$$\underbrace{2 + 3 \times 5} \quad \longrightarrow \quad \underbrace{2 + 15} \quad \longrightarrow \quad 17$$

Reducible (non-terminal) states

Terminal state

Terminal states and execution/reduction strategies

$$\underbrace{2 + 3 \times 5} \longrightarrow \underbrace{2 + 15} \longrightarrow 17$$

Reducible (non-terminal) states

Terminal state

- Let $f(x) = x \times x \times x$. What is the value of $f(3+4)$?

Terminal states and execution/reduction strategies

$$\underbrace{2+3\times 5}_{} \longrightarrow \underbrace{2+15}_{} \longrightarrow 17$$

Reducible (non-terminal) states

Terminal state

- Let $f(x) = x \times x \times x$. What is the value of $f(3+4)$?

Kim (smart)

$$\begin{array}{lcl} f(3+4) & \to & f(7) \\ & \to & 7\times 7\times 7 \\ & \to & 49\times 7 \\ & \to & 343 \end{array}$$

Lee (not so)

$$\begin{array}{lcl} f(3+4) & \to & (3+4)\times(3+4)\times(3+4) \\ & \to & 7\times(3+4)\times(3+4) \\ & \to & 7\times 7\times(3+4) \\ & \to & 7\times 7\times 7 \\ & \to & 49\times 7 \\ & \to & 343 \end{array}$$

Thurston (don't be Thurston)

$$\begin{array}{lcl} f(3+4) & \to & (3+4)\times(3+4)\times(3+4) \\ & \to & 3\times(3+4)\times(3+4)+4\times(3+4)\times(3+4) \\ & \to & \text{dozens of computation steps} \\ \dots & & \dots\dots\dots\dots\dots\dots \\ & \to & 343 \end{array}$$

Terminal states and execution/reduction strategies

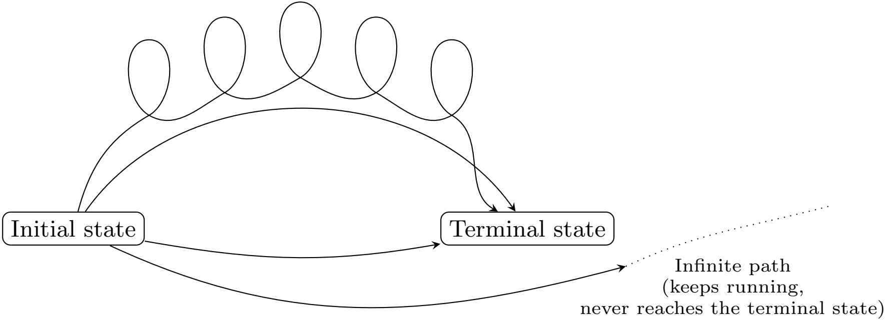
$$\underbrace{2+3\times 5} \longrightarrow \underbrace{2+15} \longrightarrow 17$$

Reducible (non-terminal) states

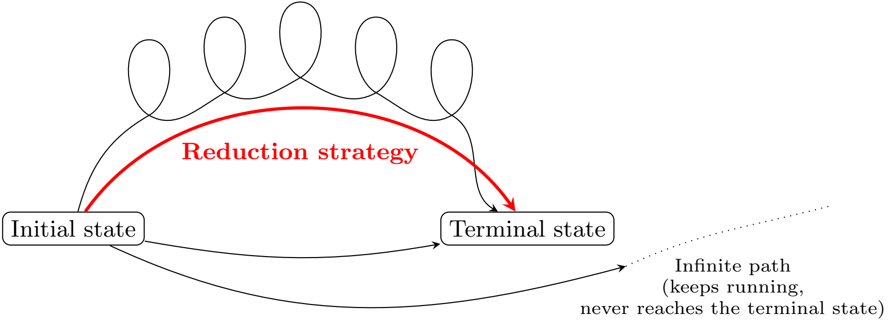
Terminal state

Terminal states and execution/reduction strategies

Terminal states and execution/reduction strategies



Terminal states and execution/reduction strategies



Reduction strategy

- **Choice** of a reduction path.
- Can be **complete** (w.r.t. termin.).
- Must be **certified**.

Goal

Equivalences of the form

"the program t is typable iff it can reach a terminal state"

Idea: **several** certificates to a same subprogram.

Intersections types (Coppo, Dezani, 1980)

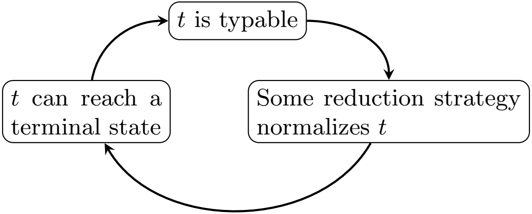
Goal

Equivalences of the form

"the program t is typable iff it can reach a terminal state"

Idea: **several** certificates to a same subprogram.

Proof: by the "circular" implications:



Intersections types (Coppo, Dezani, 1980)

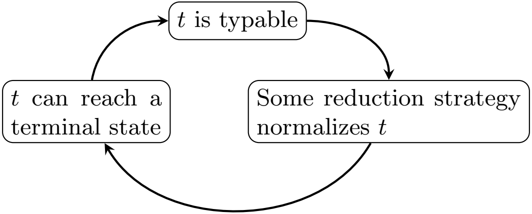
Goal

Equivalences of the form

"the program t is typable iff it can reach a terminal state"

Idea: **several** certificates to a same subprogram.

Proof: by the "circular" implications:



Intersection types

- Perhaps too expressive...
- ...but certify reduction strategies!

Non-idempotency

Computation causes duplication.

Computation causes duplication.

Non-idempotent intersection types

Disallow duplication for typing certificates.

$\rightsquigarrow$ Possibly many certificates for a subprogram.

$\rightsquigarrow$ Size of certificates decreases.

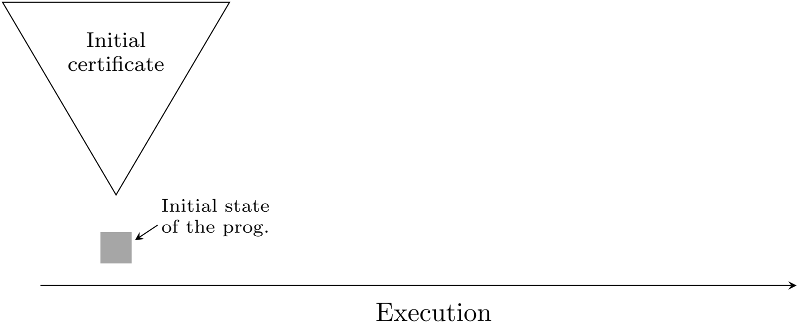
Non-idempotency

Computation causes duplication.

Non-idempotent intersection types

Disallow duplication for typing certificates.

- $\rightsquigarrow$ Possibly many certificates for a subprogram.
- $\rightsquigarrow$ Size of certificates decreases.

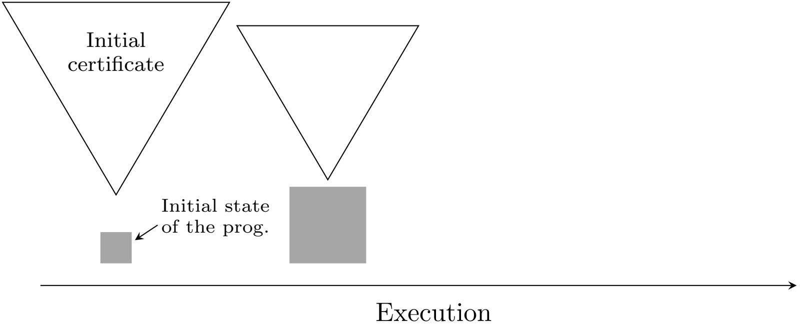


Computation causes duplication.

Non-idempotent intersection types

Disallow duplication for typing certificates.

- $\rightsquigarrow$ Possibly many certificates for a subprogram.
- $\rightsquigarrow$ Size of certificates decreases.

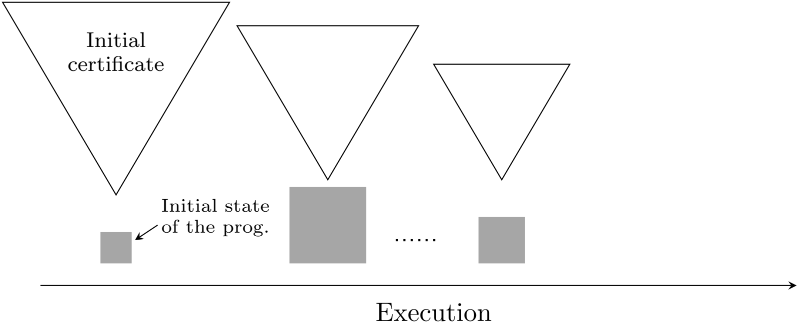


Computation causes duplication.

Non-idempotent intersection types

Disallow duplication for typing certificates.

- $\rightsquigarrow$ Possibly many certificates for a subprogram.
- $\rightsquigarrow$ Size of certificates decreases.



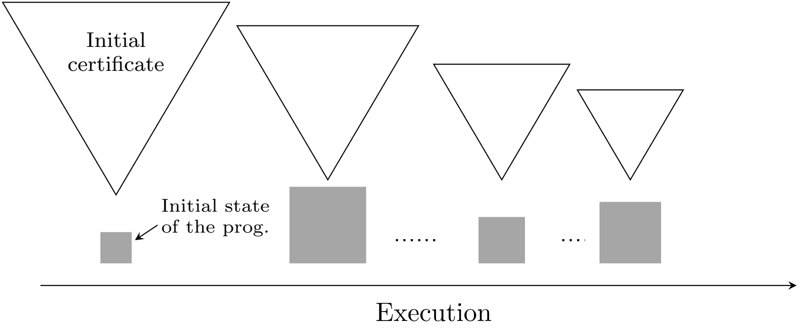
Non-idempotency

Computation causes duplication.

Non-idempotent intersection types

Disallow duplication for typing certificates.

- $\leadsto$ Possibly many certificates for a subprogram.
- $\leadsto$ Size of certificates decreases.



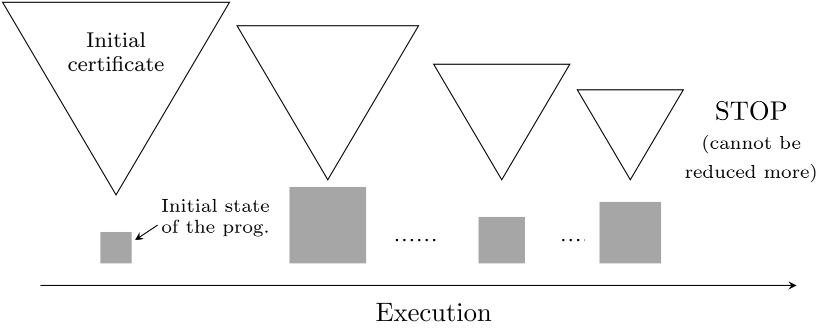
Non-idempotency

Computation causes duplication.

Non-idempotent intersection types

Disallow duplication for typing certificates.

- $\rightsquigarrow$ Possibly many certificates for a subprogram.
- $\rightsquigarrow$ Size of certificates decreases.

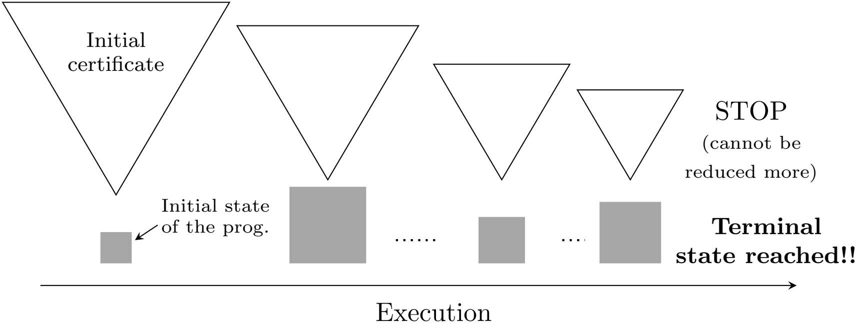


Computation causes duplication.

Non-idempotent intersection types

Disallow duplication for typing certificates.

- $\rightsquigarrow$ Possibly many certificates for a subprogram.
- $\rightsquigarrow$ Size of certificates decreases.



Computation causes duplication.

Non-idempotent intersection types

Disallow duplication for typing certificates.

- $\rightsquigarrow$ Possibly many certificates for a subprogram.
- $\rightsquigarrow$ Size of certificates decreases.

Non-idempotency

Computation causes duplication.

Non-idempotent intersection types

Disallow duplication for typing certificates.

$\rightsquigarrow$ Possibly many certificates for a subprogram.

$\rightsquigarrow$ Size of certificates decreases.

Comparative (dis)advantages

- Insanely difficult to type a particular program.
- Whole type system **easier** to study!
  - Easier proofs of **termination**!
  - Easier proofs of **characterization**!
  - Easier to certify a **reduction strategy**!

Intersection types (Coppo-Dezani 80)

- Type constructors: $o \in \mathscr{O}$, $\rightarrow$ and $\wedge$ (intersection).

Intersection types (Coppo-Dezani 80)

- Type constructors: $o \in \mathscr{O}$, $\rightarrow$ and $\wedge$ (intersection).
- *Strict* types: no $\wedge$ on the *right* h.s. of $\rightarrow$ (*e.g.*, $(A \wedge B) \rightarrow A$, not $A \rightarrow (B \wedge C)$)

  $\rightsquigarrow$ *no intro/elim. rules for* $\wedge$

- Type constructors: $o \in \mathscr{O}$, $\rightarrow$ and $\wedge$ (intersection).
- *Strict* types: no $\wedge$ on the *right* h.s. of $\rightarrow$ (*e.g.*, $(A \wedge B) \rightarrow A$, not $A \rightarrow (B \wedge C)$)
  $\rightsquigarrow$ *no intro/elim. rules for* $\wedge$
- $(A \wedge B) \wedge C \sim A \wedge (B \wedge C)$, $A \wedge B \sim B \wedge A$ (**assoc.** and **comm.**)
  $\rightsquigarrow$ *subtyping or permutation rules e.g.,*

$$\frac{\Gamma, x : A_1 \wedge A_2 \wedge \ldots \wedge A_n \vdash t : B \qquad \rho \in \mathfrak{S}_n}{\Gamma, x : A_{\rho(1)} \wedge \ldots \wedge A_{\rho(n)} \vdash t : B}\texttt{perm}$$

Intersection types (Coppo-Dezani 80)

- Type constructors: $o \in \mathscr{O}$, $\rightarrow$ and $\wedge$ (intersection).
- *Strict* types: no $\wedge$ on the *right* h.s. of $\rightarrow$ (*e.g.*, $(A \wedge B) \rightarrow A$, not $A \rightarrow (B \wedge C)$)

  $\rightsquigarrow$ *no intro/elim. rules for* $\wedge$
- $(A \wedge B) \wedge C \sim A \wedge (B \wedge C)$, $A \wedge B \sim B \wedge A$ (**assoc.** and **comm.**)

  $\rightsquigarrow$ *subtyping or permutation rules e.g.,*

$$\frac{\Gamma, x : A_1 \wedge A_2 \wedge \ldots \wedge A_n \vdash t : B \qquad \rho \in \mathfrak{S}_n}{\Gamma, x : A_{\rho(1)} \wedge \ldots \wedge A_{\rho(n)} \vdash t : B}\,\texttt{perm}$$

- **Idempotency?** $A \wedge A \sim A$ (Coppo-D) or not (Gardner 94-de Carvalho 07)

  idem: typing = qualitative info *non-idem: qual. and quant.*

Intersection types (Coppo-Dezani 80)

- Type constructors: $o \in \mathscr{O}$, $\to$ and $\wedge$ (intersection).
- *Strict* types: no $\wedge$ on the *right* h.s. of $\to$ (*e.g.*, $(A \wedge B) \to A$, not $A \to (B \wedge C)$)

  $\rightsquigarrow$ *no intro/elim. rules for* $\wedge$
- $(A \wedge B) \wedge C \sim A \wedge (B \wedge C)$, $A \wedge B \sim B \wedge A$ (**assoc.** and **comm.**)

  $\rightsquigarrow$ *subtyping or permutation rules e.g.,*

$$\frac{\Gamma, x : A_1 \wedge A_2 \wedge \ldots \wedge A_n \vdash t : B \qquad \rho \in \mathfrak{S}_n}{\Gamma, x : A_{\rho(1)} \wedge \ldots \wedge A_{\rho(n)} \vdash t : B}\texttt{perm}$$

- **Idempotency?** $A \wedge A \sim A$ (Coppo-D) or not (Gardner 94-de Carvalho 07)

  idem: typing = qualitative info *non-idem: qual. and quant.*
- Collapsing $A \wedge B \wedge C$ into $[A, B, C]$ (**multiset**) $\rightsquigarrow$ no need for perm rules etc.

$$[A, B, A] = [A, B, A] \neq [A, B] \qquad [A, B, A] = [A, B] + [A]$$

$$\begin{array}{llll} \textbf{(Strict Types)} & \tau, \sigma & := & o \in \mathscr{O} \mid \mathcal{I} \to \tau \\ \textbf{(Intersection Types)} & \mathcal{I} & := & [\sigma_i]_{i \in I} \end{array}$$

Strict types $\rightsquigarrow$ **syntax directed rules**:

$$\frac{}{x : [\tau] \vdash x : \tau}\texttt{ax} \qquad \frac{\Gamma; x : [\sigma_i]_{i \in I} \vdash t : \tau}{\Gamma \vdash \lambda x.t : [\sigma_i]_{i \in I} \to \tau}\texttt{abs}$$

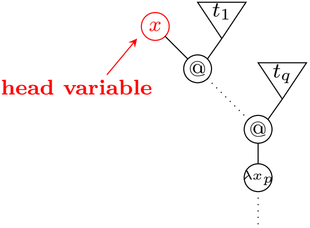
$$\frac{\Gamma \vdash t : [\sigma_i]_{i \in I} \to \tau \qquad (\Gamma_i \vdash u : \sigma_i)_{i \in I}}{\Gamma +_{i \in I} \Gamma_i \vdash t\, u : \tau}\texttt{app}$$

Remark

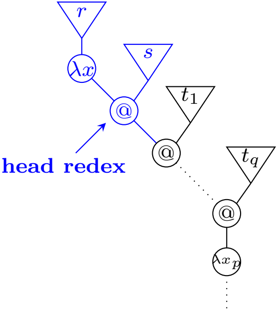
- **Relevant** system (no weakening)
- In **app**-rule, pointwise multiset sum *e.g.*,

$$(x : [\sigma]; y : [\tau]) + (x : [\sigma, \tau]) = x : [\sigma, \sigma, \tau]; y : [\tau]$$

Head Normalization (λ)

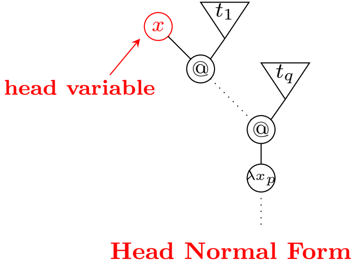


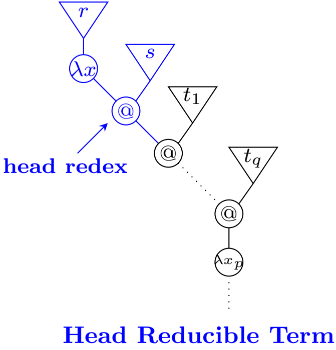
Head Normal Form



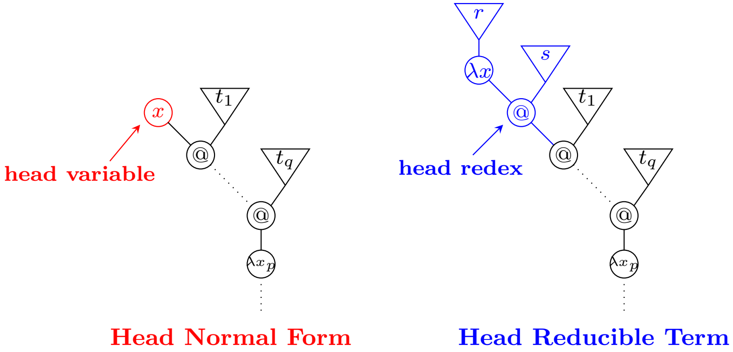
Head Reducible Term

Head Normalization (λ)





- t is **head normalizing (HN)** if $\exists$ reduction path from t to a HNF.



- t is **head normalizing (HN)** if $\exists$ reduction path from t to a HNF.
- The **head reduction strategy**: reducing head redexes while it is possible.

Head Normalization (λ)

- t is **head normalizing (HN)** if $\exists$ reduction path from t to a HNF.
- The **head reduction strategy**: reducing head redexes while it is possible.

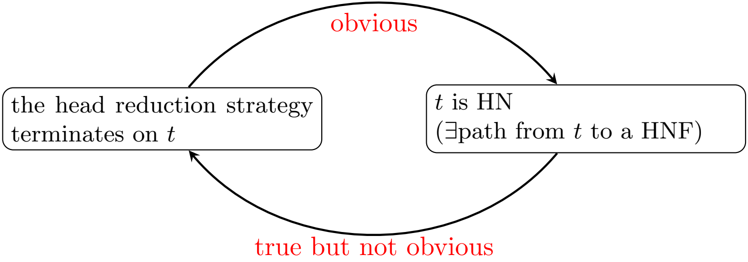
obvious

the head reduction strategy terminates on t

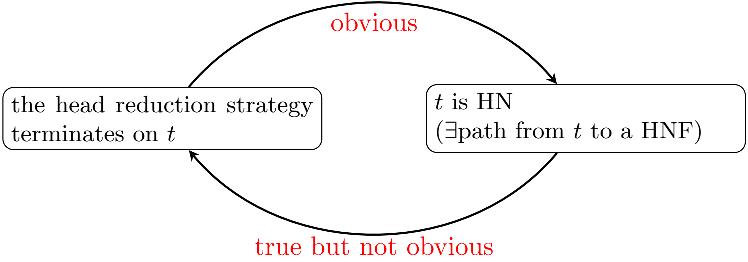
t is HN ($\exists$path from t to a HNF)

true but not obvious

- t is **head normalizing (HN)** if $\exists$ reduction path from t to a HNF.
- The **head reduction strategy**: reducing head redexes while it is possible.



- The **head reduction strategy**: reducing head redexes while it is possible.



Intersection types come to help!

- The **head reduction strategy**: reducing head redexes while it is possible.

Subject Reduction and Subject Expansion

A good intersection type system should enjoy:

Subject Reduction (SR):
Typing is stable under reduction.

Subject Expansion (SE):
Typing is stable under anti-reduction.

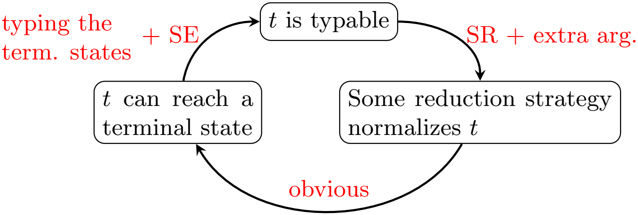
SE is usually not verified by simple or polymorphic type systems

A good intersection type system should enjoy:

Subject Reduction (SR):
Typing is stable under reduction.

Subject Expansion (SE):
Typing is stable under anti-reduction.

SE is usually not verified by simple or polymorphic type systems



Properties ($\mathscr{R}_0$)

- **Weighted Subject Reduction**
  - Reduction preserves types and environments, and...
  - ... *head* reduction strictly decreases the nodes of the deriv. tree (`size`).
- **Subject Expansion**
  - Anti-reduction preserves types and environments.

Theorem (de Carvalho)

Let t be a λ-term. Then equivalence between:

1. *t is typable (in $\mathscr{R}_0$)*
2. *t is HN*
3. *the head reduction strategy terminates on t ($\rightsquigarrow$ certification!)*

Bonus (quantitative information)

If Π types t, then $\texttt{size}(\Pi)$ bounds the number of **steps** of the head red. strategy on t

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$

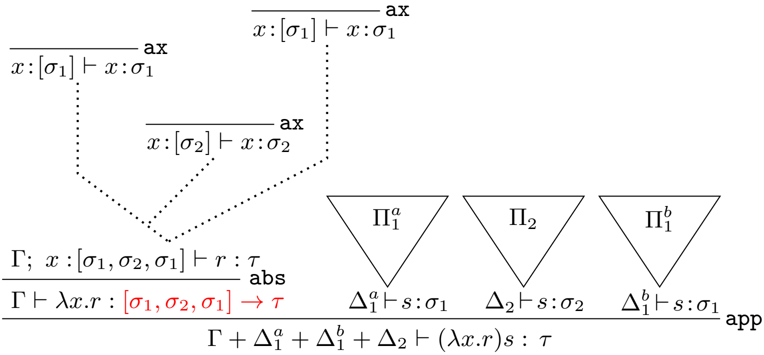
$$\dfrac{}{x:[\sigma_1]\vdash x:\sigma_1}\,\texttt{ax}$$

$$\dfrac{}{x:[\sigma_1]\vdash x:\sigma_1}\,\texttt{ax} \qquad \dfrac{}{x:[\sigma_2]\vdash x:\sigma_2}\,\texttt{ax}$$

$$\dfrac{\dfrac{\Gamma;\ x:[\sigma_1,\sigma_2,\sigma_1]\vdash r:\tau}{\Gamma\vdash\lambda x.r:[\sigma_1,\sigma_2,\sigma_1]\to\tau}\,\texttt{abs} \quad \begin{array}{c}\Pi_1^a\\ \Delta_1^a\vdash s:\sigma_1\end{array} \quad \begin{array}{c}\Pi_2\\ \Delta_2\vdash s:\sigma_2\end{array} \quad \begin{array}{c}\Pi_1^b\\ \Delta_1^b\vdash s:\sigma_1\end{array}}{\Gamma+\Delta_1^a+\Delta_1^b+\Delta_2\vdash(\lambda x.r)s:\tau}\,\texttt{app}$$

Subject reduction and expansion in $\mathscr{R}_0$

From a typing of $(\lambda x.r)s$ … to a typing of $r[s/x]$



From a typing of $(\lambda x.r)s$ … to a typing of $r[s/x]$

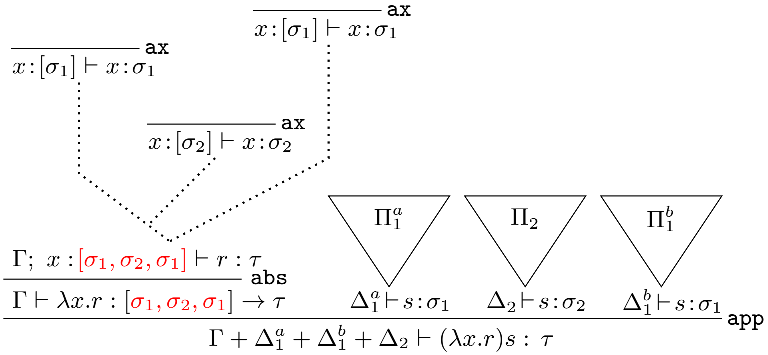
$$\frac{}{x:[\sigma_1]\vdash x:\sigma_1}\,\texttt{ax}$$

$$\frac{}{x:[\sigma_1]\vdash x:\sigma_1}\,\texttt{ax}$$

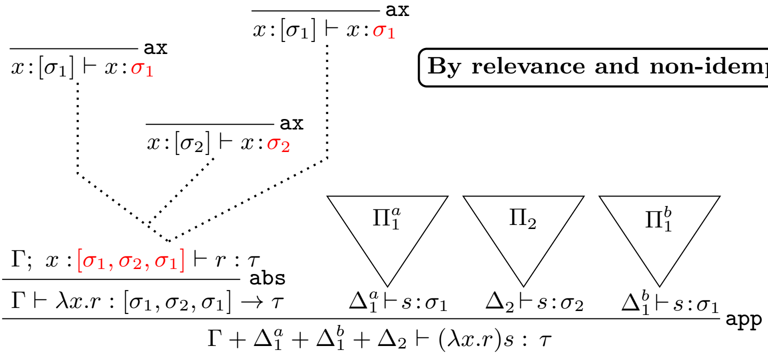
$$\frac{}{x:[\sigma_2]\vdash x:\sigma_2}\,\texttt{ax}$$

$$\dfrac{\dfrac{\Gamma;\ x:[\sigma_1,\sigma_2,\sigma_1]\vdash r:\tau}{\Gamma\vdash \lambda x.r:[\sigma_1,\sigma_2,\sigma_1]\to\tau}\,\texttt{abs} \qquad \overset{\Pi_1^a}{\Delta_1^a\vdash s:\sigma_1} \qquad \overset{\Pi_2}{\Delta_2\vdash s:\sigma_2} \qquad \overset{\Pi_1^b}{\Delta_1^b\vdash s:\sigma_1}}{\Gamma+\Delta_1^a+\Delta_1^b+\Delta_2\vdash(\lambda x.r)s:\ \tau}\,\texttt{app}$$

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$

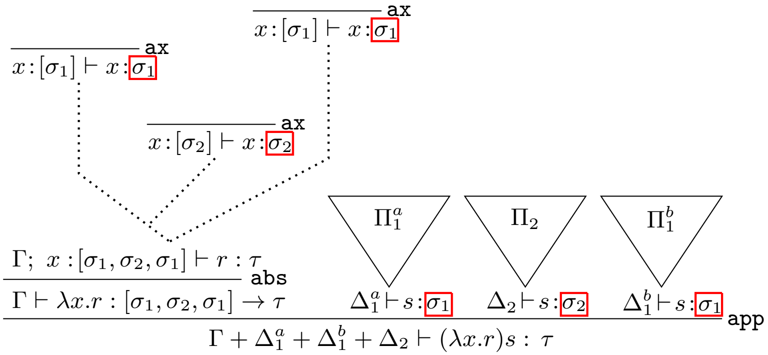


From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$

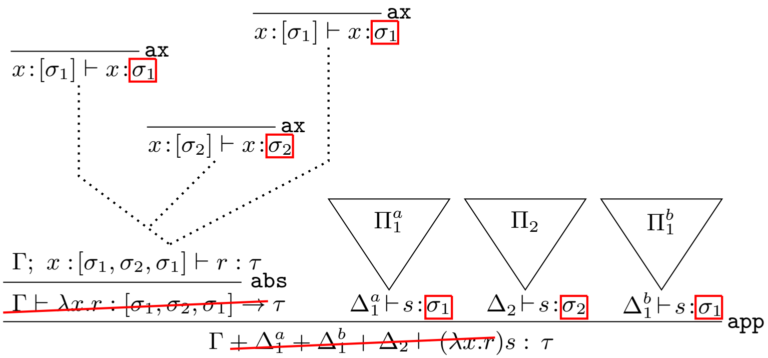


By relevance and non-idempotence !

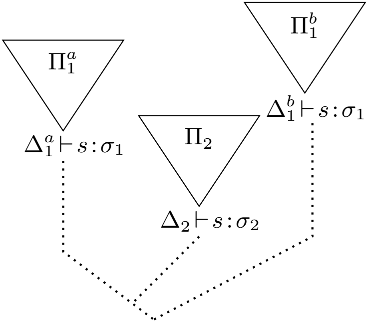
From a typing of $(\lambda x.r)s$ … to a typing of $r[s/x]$



From a typing of $(\lambda x.r)s$ … to a typing of $r[s/x]$

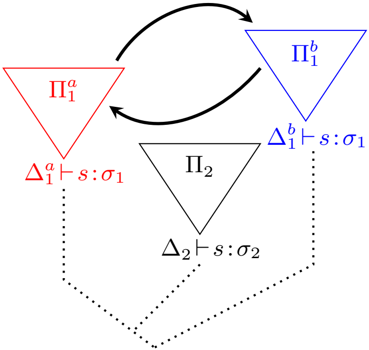


From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



$$\Gamma + \Delta_1^a + \Delta_1^b + \Delta_2 \vdash r[s/x] : \tau$$

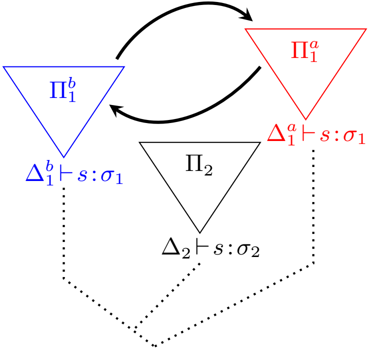
From a typing of $(\lambda x.r)s$ … to a typing of $r[s/x]$



Non-determinism of SR

$\Gamma + \Delta_1^a + \Delta_1^b + \Delta_2 \vdash r[s/x] : \tau$

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



Non-determinism of SR

$\Gamma + \Delta_1^a + \Delta_1^b + \Delta_2 \vdash r[s/x] : \tau$

Outline

Context

Non-idempotent intersection types

Using type A *once* or *twice* not the same

Rigid vs. non-rigid paradigms

Proof red. deterministic vs. non-deterministic

Question 1

Rigid collapses on non-rigid

Is this collapse surjective?

(A, A, B) and (A, B, A) collapse on $[A, A, B]$

Question 2

In rigid fw., red. paths captured by permutations

Is it possible to capture red. paths without perm. ?

$(A, B, A) \mapsto (A, A, B)$

All this in a coinductive fw. (no productivity)!

Plan

Motivations

- Multiset intersection:
  - $\oplus$ syntax-direction
  - $\ominus$ non-determinism of proof red.
  - $\ominus$ lack tracking: $[\sigma, \tau, \sigma] = [\underset{?}{\sigma}, \tau] + [\underset{?}{\sigma}]$.

- Multiset intersection:
  - $\oplus$ syntax-direction
  - $\ominus$ non-determinism of proof red.
  - $\ominus$ lack tracking: $[\sigma, \tau, \sigma] = [\underset{?}{\sigma}, \tau] + [\underset{?}{\sigma}]$.

- **Klop's Problem:** can the set of ∞-WN terms be characterized by an ITS ?

  Def: t is ∞-WN iff its Böhm tree does not contain $\perp$

  - **Tatsuta [07]:** an inductive ITS cannot do it.
  - Can a coinductive ITS characterize the set of ∞-WN terms?

- Multiset intersection:
  - $\oplus$ syntax-direction
  - $\ominus$ non-determinism of proof red.
  - $\ominus$ lack tracking: $[\sigma, \tau, \sigma] = [\underset{?}{\sigma}, \tau] + [\underset{?}{\sigma}]$.

- **Klop's Problem:** can the set of ∞-WN terms be characterized by an ITS ?

  Def: t is ∞-WN iff its Böhm tree does not contain $\perp$

  - **Tatsuta [07]:** an inductive ITS cannot do it.
  - Can a coinductive ITS characterize the set of ∞-WN terms?

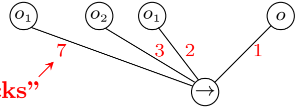
- *Answer:*
  - Impossible without tracking (need for a validity criterion).

    system $\mathscr{R}$ (*i.e.* $\mathscr{R}_0$ with a coinductive type grammar) does not work
  - **YES**, with inter. = **sequences** + **validity criterion**.

- **Strict Types:**

$$S_k, T \ ::= \ o \in \mathscr{O} \mid (k \cdot S_k)_{k\in K} \to T$$

- **Sequence Types** $(k \cdot S_k)_{k \in K}$
- *Example:* $(7 \cdot o_1, 3 \cdot o_2, 2 \cdot o_1) \to o$



7, 3, 2, 1 = **"tracks"**

- **Tracking:** $(3 \cdot \sigma, 5 \cdot \tau, 9 \cdot \sigma) = (3 \cdot \sigma, 5 \cdot \tau) \uplus (9 \cdot \sigma)$

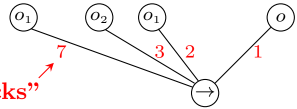
  vs. $[\sigma, \tau, \sigma] = [\sigma, \tau] + [\sigma]$

- **Strict Types:**

$$S_k, T \ ::= \ o \in \mathscr{O} \mid (k \cdot S_k)_{k \in K} \to T$$

- **Sequence Types** $\quad (k \cdot S_k)_{k \in K}$

- *Example:* $(7 \cdot o_1, 3 \cdot o_2, 2 \cdot o_1) \to o$



7, 3, 2, 1 = **"tracks"**

- **Tracking:** $(3 \cdot \sigma, 5 \cdot \tau, 9 \cdot \sigma) = (3 \cdot \sigma, 5 \cdot \tau) \uplus (9 \cdot \sigma)$

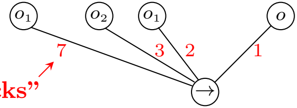
  vs. $[\sigma, \tau, \sigma] = [\sigma\,, \tau] + [\sigma\,]$

SEQUENTIAL INTERSECTION

- **Strict Types:**

$$S_k, T \ ::= \ o \in \mathscr{O} \mid (k \cdot S_k)_{k \in K} \to T$$

- **Sequence Types** $\quad (k \cdot S_k)_{k \in K}$
- *Example:* $(7 \cdot o_1, 3 \cdot o_2, 2 \cdot o_1) \to o$



7, 3, 2, 1 = **"tracks"**

- **Tracking:** $(3 \cdot \sigma, 5 \cdot \tau, 9 \cdot \sigma) = (3 \cdot \sigma, 5 \cdot \tau) \uplus (9 \cdot \sigma)$

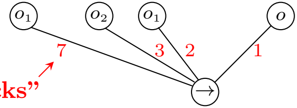
$$\textit{vs.} \ [\sigma, \tau, \sigma] = [\sigma, \tau] + [\sigma]$$

- **Strict Types:**

$$S_k, T \ ::= \ o \in \mathscr{O} \mid (k \cdot S_k)_{k \in K} \to T$$

- **Sequence Types** $\quad (k \cdot S_k)_{k \in K}$
- *Example:* $(7 \cdot o_1, 3 \cdot o_2, 2 \cdot o_1) \to o$

7, 3, 2, 1 = **"tracks"**



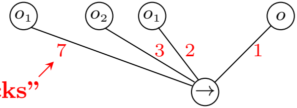
- **Tracking:** $(3 \cdot \sigma, 5 \cdot \tau, 9 \cdot \sigma) = (3 \cdot \sigma, 5 \cdot \tau) \uplus (9 \cdot \sigma)$

vs. $[\sigma, \tau, \sigma] = [\sigma\ , \tau] + [\sigma\ ]$

- **Strict Types:**

$$S_k, T \ ::= \ o \in \mathscr{O} \mid (k \cdot S_k)_{k \in K} \to T$$

- **Sequence Types** $\quad (k \cdot S_k)_{k \in K}$
- *Example:* $(7 \cdot o_1, 3 \cdot o_2, 2 \cdot o_1) \to o$



7, 3, 2, 1 = **"tracks"**

- **Tracking:** $(3 \cdot \sigma, 5 \cdot \tau, 9 \cdot \sigma) = (3 \cdot \sigma, 5 \cdot \tau) \uplus (9 \cdot \sigma)$

vs. $[\sigma, \tau, \sigma] = [\underset{?}{\sigma}, \tau] + [\underset{?}{\sigma}]$

Derivations of S

$$\frac{}{x:\,(k\cdot T)\vdash x:\,T}\texttt{ax} \qquad \frac{C;\,x:(S_k)_{k\in K}\vdash t:T}{C\vdash \lambda x.t:(S_k)_{k\in K}\to T}\texttt{abs}$$

$$\frac{C\vdash t:\,(S_k)_{k\in K}\to T \qquad (D_k\vdash u:\,S_k)_{k\in K}}{C\uplus(\uplus_{k\in K}D_k)\vdash t\,u:\,T}\texttt{app}$$

Derivations of S

$$\frac{}{x : (k \cdot T) \vdash x : T}\,\texttt{ax} \qquad \frac{C;\, x : (S_k)_{k \in K} \vdash t : T}{C \vdash \lambda x.t : (S_k)_{k \in K} \to T}\,\texttt{abs}$$

$$\frac{C \vdash t : (S_k)_{k \in K} \to T \qquad (D_k \vdash u : S_k)_{k \in K}}{C \uplus (\uplus_{k \in K} D_k) \vdash t\,u : T}\,\texttt{app}$$

- System $\texttt{S}$ features **pointers** (called **bipositions**).

Derivations of S

$$\frac{}{x : (k \cdot T) \vdash x : T}\texttt{ax} \qquad \frac{C; x : (S_k)_{k \in K} \vdash t : T}{C \vdash \lambda x.t : (S_k)_{k \in K} \to T}\texttt{abs}$$

$$\frac{C \vdash t : (S_k)_{k \in K} \to T \qquad (D_k \vdash u : S_k)_{k \in K}}{C \uplus (\uplus_{k \in K} D_k) \vdash t\, u : T}\texttt{app}$$

- System $\mathtt{S}$ features **pointers** (called **bipositions**).

Every $\mathtt{S}$-derivation collapses on a $\mathscr{R}$-derivation.

Derivations of S

$$\dfrac{}{x:\,(k\cdot T)\vdash x:\,T}\texttt{ax} \qquad \dfrac{C;\,x:(S_k)_{k\in K}\vdash t:T}{C\vdash \lambda x.t:(S_k)_{k\in K}\to T}\texttt{abs}$$

$$\dfrac{C\vdash t:\,(S_k)_{k\in K}\to T \qquad (D_k\vdash u:\,S_k)_{k\in K}}{C\uplus(\uplus_{k\in K}D_k)\vdash t\,u:\,T}\texttt{app}$$

- System S features **pointers** (called **bipositions**).

$$\boxed{\text{Every S-derivation collapses on a } \mathscr{R}\text{-derivation.}}$$

- Subject reduction is deterministic in S ($\neq \mathscr{R}$).

Theorem (V,LiCS17)

- *A ∞-term t is ∞-WN iff t is* $\mathtt{S}$*-typable in some way.* $\rightsquigarrow$ *Klop's Problem solved*
- *The hereditary head reduction strategy is complete for infinitary weak normalization.*

Infinitary Typing

Theorem (V,LiCS17)

- *A ∞-term t is ∞-WN iff t is $\mathtt{S}$-typable in some way.* $\rightsquigarrow$ *Klop's Problem solved*
- *The hereditary head reduction strategy is complete for infinitary weak normalization.*

Bonus (positive answer to TLCA Problem #20)

System $\mathtt{S}$ also provides a type-theoretic characterization of the **hereditary permutations** (not possible in the inductive case, Tatsuta [LiCS07]).

Infinitary Typing

Theorem (V,LiCS17)

- *A ∞-term t is ∞-WN iff t is* $\mathtt{S}$*-typable in some way.* $\rightsquigarrow$ *Klop's Problem solved*
- *The hereditary head reduction strategy is complete for infinitary weak normalization.*

Bonus (positive answer to TLCA Problem #20)

System $\mathtt{S}$ also provides a type-theoretic characterization of the **hereditary permutations** (not possible in the inductive case, Tatsuta [LiCS07]).

Theorem (V,LiCS18)

- *Every term is typable in systems $\mathscr{R}$ and* $\mathtt{S}$ *(non-trivial).*
- *One can extract from the $\mathscr{R}$-typing the* ***order*** *(arity) of any λ-term.*
- *In the infinitary relational model, no term has an empty denotation.*

The Problem of the Collapse

- Coinductive typing (without validity criterion): allow to type all normalizing terms + some unproductive terms *e.g.*, Ω.

The Problem of the Collapse

- Coinductive typing (without validity criterion): allow to type all normalizing terms + some unproductive terms *e.g.*, Ω.
- Necessity to replace $\mathscr{R}$ (multiset inter.) with $\mathtt{S}$ (sequence inter)

The Problem of the Collapse

- Coinductive typing (without validity criterion): allow to type all normalizing terms + some unproductive terms *e.g.*, Ω.

- Necessity to replace $\mathscr{R}$ (multiset inter.) with $\mathtt{S}$ (sequence inter)

- But do we lose some derivations?

$\boxed{\textbf{Question:}\text{ given } \Pi \text{ a } \mathscr{R}\text{-derivation, is there a } \mathtt{S}\text{-deriv. } P \text{ collapsing on } \Pi\text{?}}$

if true, the infinitary relational model is fully described by system $\mathtt{S}$

The Problem of the Collapse

- Coinductive typing (without validity criterion): allow to type all normalizing terms + some unproductive terms *e.g.*, Ω.
- Necessity to replace $\mathscr{R}$ (multiset inter.) with $\mathtt{S}$ (sequence inter)
- But do we lose some derivations?

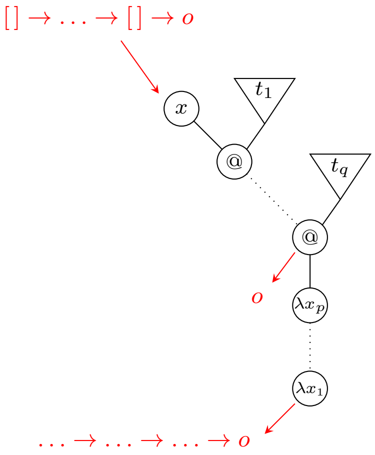
Question: given Π a $\mathscr{R}$-derivation, is there a $\mathtt{S}$-deriv. P collapsing on Π?

if true, the infinitary relational model is fully described by system $\mathtt{S}$

- Easy in the case of normal forms (*i.e.* when Π types a NF), not in other cases.

Difficulties

- In the *productive* cases (HN,WN,SN,∞-WN), in i.t.s., one types the normal forms and uses subject expansion.
  $\boxed{\text{normalizing terms} \subseteq \text{typable terms}}$
- Here, no form of productivity/stabilization.
- We develop a corpus of methods inspired by **first order model theory** (last part of the talk).



Outline

Context

Non-idempotent intersection types

Using type A *once* or *twice* not the same

Rigid vs. non-rigid paradigms

Proof red. deterministic vs. non-deterministic

Question 1

Rigid collapses on non-rigid

Is this collapse surjective?

(A, A, B) and (A, B, A) collapse on $[A, A, B]$

Question 2

In rigid fw., red. paths captured by permutations

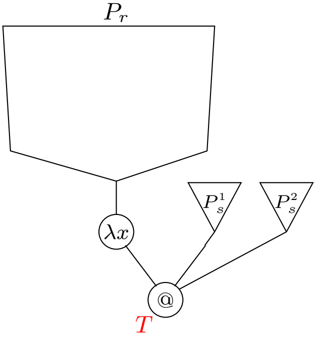
$(A, B, A) \mapsto (A, A, B)$

Is it possible to capture red. paths without perm. ?

All this in a coinductive fw. (no productivity)!

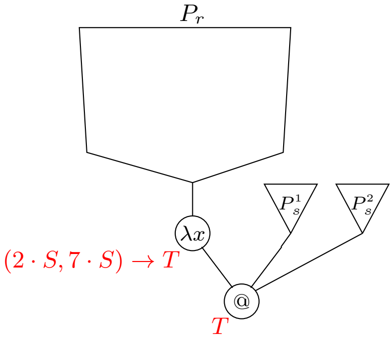
Plan

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



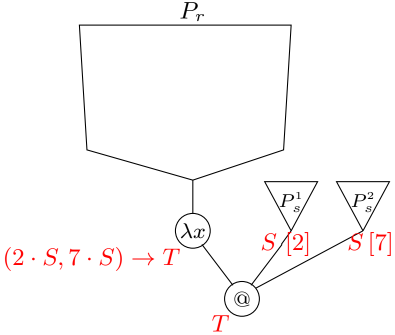
reduces into...

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



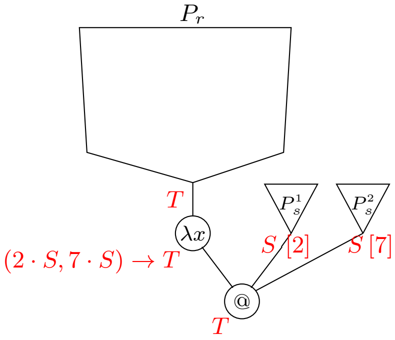
reduces into...

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



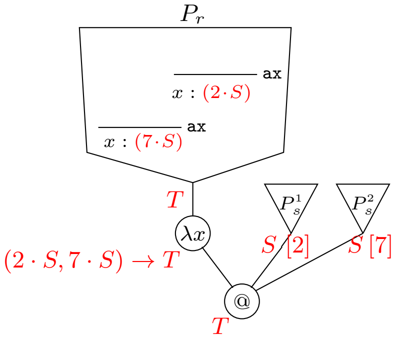
reduces into...

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



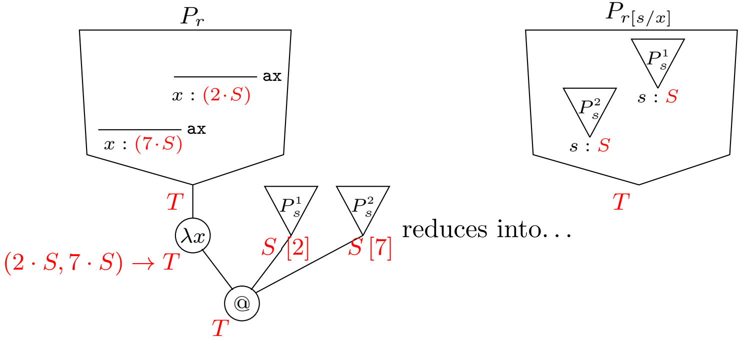
reduces into...

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



reduces into...

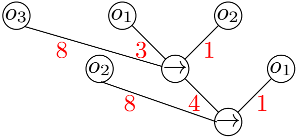
From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$

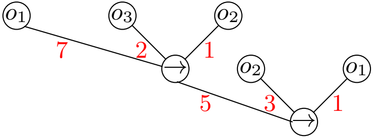


How to encode reduction paths?

- System $\mathtt{S}$: one red. path, poor dynamic behavior.
- System $\mathscr{R}$: rich dynamic behavior, impossible to express red. paths (lack of tracking)

- **Idea 1:** use **iso. of types** (iso of lab. trees)





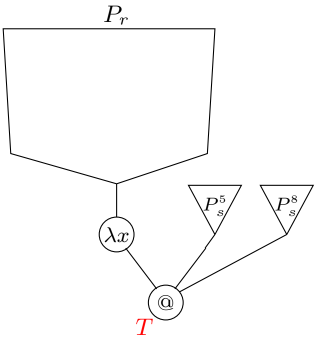
$T_1 = (8\cdot o_2, 4\cdot(8\cdot o_3, 3\cdot o_1) \to o_2) \to o_1$ $\qquad$ $T_2 = (5\cdot(7\cdot o_1, 2\cdot o_3) \to o_2, 3\cdot o_2) \to o_1$

- **Idea 2:** replace $\mathtt{app}$ (syntactic eq.) with $\mathtt{app_h}$ (eq. up to iso)

$$\frac{C \vdash t : (S_k)_{k\in K} \to T \qquad (D_k \vdash u : S'_k)_{k\in K'} \qquad (S_k)_{k\in K} \equiv (S'_k)_{k\in K'}}{C \uplus (\uplus_{k\in K} D_k) \vdash t\, u : T}\,\mathtt{app_h}$$

- **Hybrid** system $\mathtt{S_h}$: every $\mathscr{R}$-deriv. is a $\mathtt{S_h}$-collapse (easy).

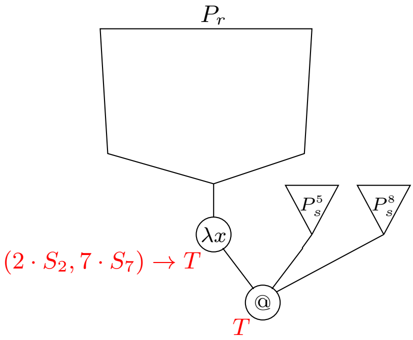
From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



reduces into...

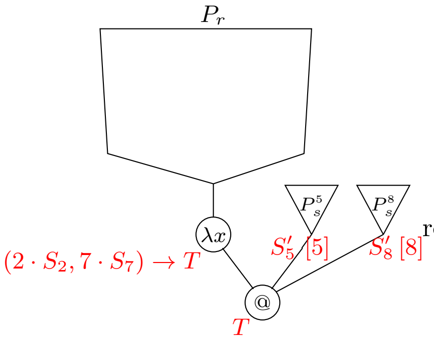
Pseudo-Subject Reduction in $\mathtt{S_h}$

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



reduces into...

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$

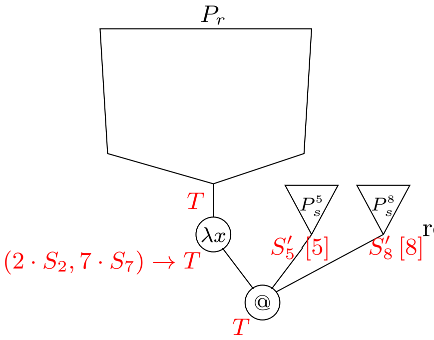


reduces into...

with $(2 \cdot S_2, 7 \cdot S_7) \equiv (5 \cdot S'_5, 8 \cdot S'_8)$

Pseudo-Subject Reduction in $\mathtt{S_h}$

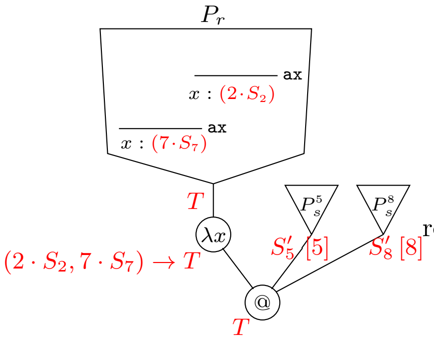
From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



reduces into...

with $(2\cdot S_2, 7\cdot S_7) \equiv (5\cdot S'_5, 8\cdot S'_8)$

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$

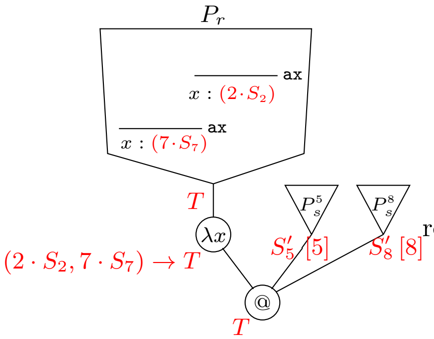


reduces into...

with $(2 \cdot S_2, 7 \cdot S_7) \equiv (5 \cdot S'_5, 8 \cdot S'_8)$

Pseudo-Subject Reduction in $\mathtt{S}_{\mathtt{h}}$

From a typing of $(\lambda x.r)s$ … to a typing of $r[s/x]$

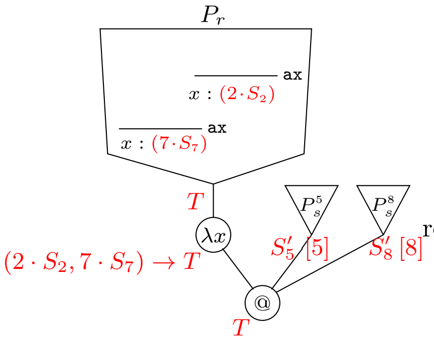


reduces into…

with $(2\cdot S_2, 7\cdot S_7) \equiv (5\cdot S'_5, 8\cdot S'_8)$

$\boxed{\textbf{Assume } S_2 \not\equiv S_7}$

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$

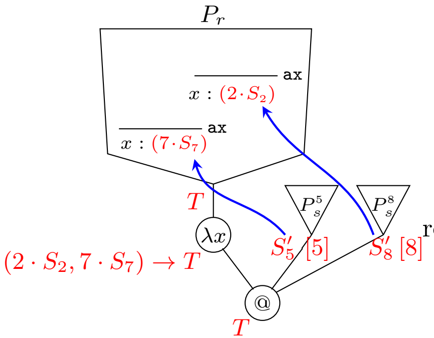


reduces into...

with $(2\cdot S_2, 7\cdot S_7) \equiv (5\cdot S'_5, 8\cdot S'_8)$

$\boxed{\textbf{Assume } S_2 \not\equiv S_7}$ say $S_2 \equiv S'_8$, $S_7 \equiv S'_5$

From a typing of $(\lambda x.r)s$ … to a typing of $r[s/x]$

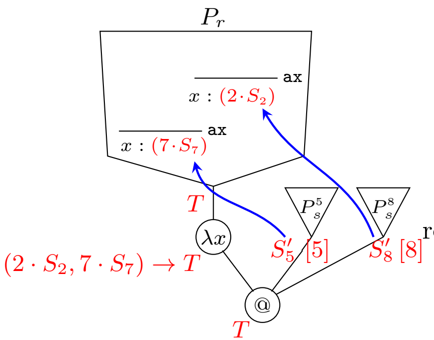


reduces into…

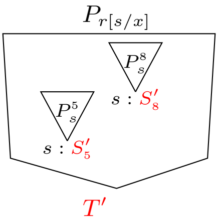
with $(2 \cdot S_2, 7 \cdot S_7) \equiv (5 \cdot S'_5, 8 \cdot S'_8)$

$\boxed{\textbf{Assume } S_2 \not\equiv S_7}$ say $S_2 \equiv S'_8$, $S_7 \equiv S'_5$

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



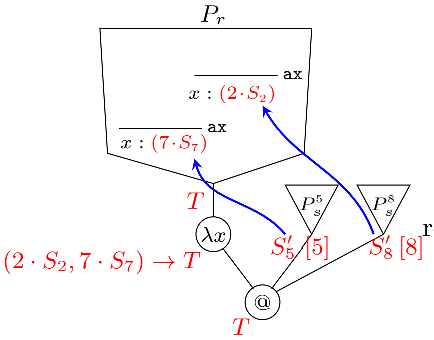
reduces into



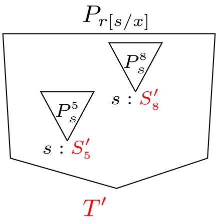
with $(2 \cdot S_2, 7 \cdot S_7) \equiv (5 \cdot S'_5, 8 \cdot S'_8)$

$\boxed{\textbf{Assume } S_2 \not\equiv S_7}$ say $S_2 \equiv S'_8$, $S_7 \equiv S'_5$

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



reduces into



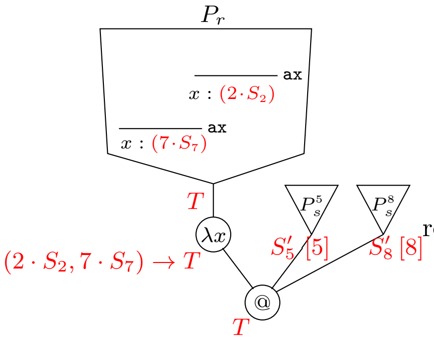
with $T' \equiv T$

with $(2\cdot S_2, 7\cdot S_7) \equiv (5\cdot S'_5, 8\cdot S'_8)$

$\boxed{\textbf{Assume } S_2 \not\equiv S_7}$ say $S_2 \equiv S'_8$, $S_7 \equiv S'_5$

Pseudo-Subject Reduction in $\mathtt{S_h}$

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$

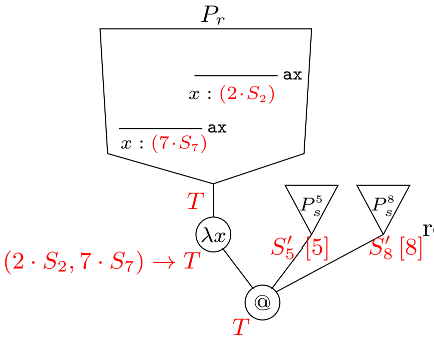


reduces into...

with $(2 \cdot S_2, 7 \cdot S_7) \equiv (5 \cdot S'_5, 8 \cdot S'_8)$

Pseudo-Subject Reduction in $\mathtt{S_h}$

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



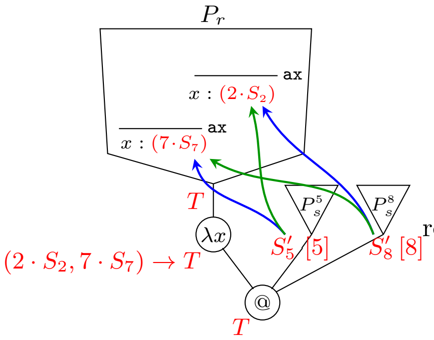
reduces into...

with $(2 \cdot S_2, 7 \cdot S_7) \equiv (5 \cdot S'_5, 8 \cdot S'_8)$

$\boxed{\textbf{Assume } S_2 \equiv S_7}$ s.t. $S_2 \equiv S_7 \equiv S'_5 \equiv S'_8$

Pseudo-Subject Reduction in $\mathtt{S_h}$

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$

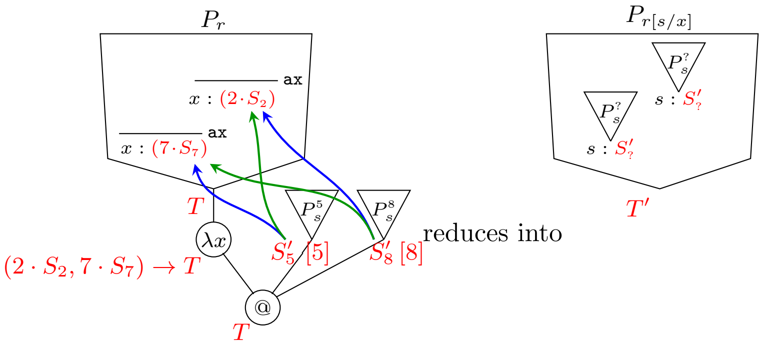


reduces into

with $(2 \cdot S_2, 7 \cdot S_7) \equiv (5 \cdot S'_5, 8 \cdot S'_8)$

$\boxed{\textbf{Assume } S_2 \equiv S_7}$ s.t. $S_2 \equiv S_7 \equiv S'_5 \equiv S'_8$

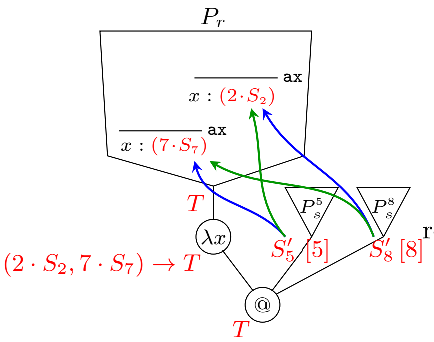
From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



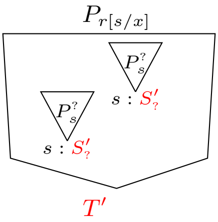
with $(2 \cdot S_2, 7 \cdot S_7) \equiv (5 \cdot S'_5, 8 \cdot S'_8)$

$\boxed{\textbf{Assume } S_2 \equiv S_7}$ s.t. $S_2 \equiv S_7 \equiv S'_5 \equiv S'_8$

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



reduces into



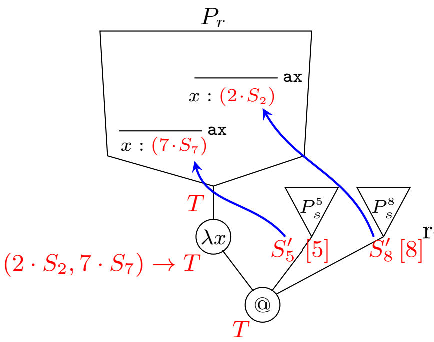
ϕ **interface**
$(2\cdot S_2, 7\cdot S_7) \tilde{\to} (5\cdot S'_5, 8\cdot S'_8)$

with $(2\cdot S_2, 7\cdot S_7) \equiv (5\cdot S'_5, 8\cdot S'_8)$

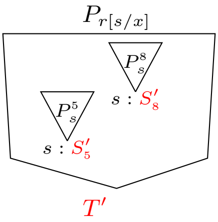
$\boxed{\textbf{Assume } S_2 \equiv S_7}$ s.t. $S_2 \equiv S_7 \equiv S'_5 \equiv S'_8$

Pseudo-Subject Reduction in $\mathtt{S_h}$

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



reduces into



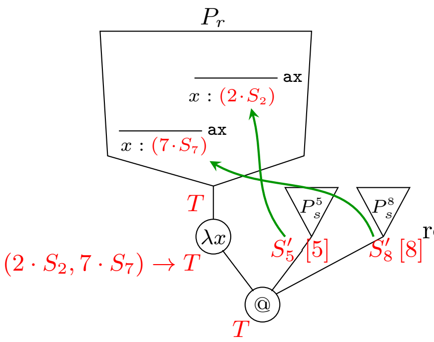
with $(2 \cdot S_2, 7 \cdot S_7) \equiv (5 \cdot S'_5, 8 \cdot S'_8)$

ϕ interface $(2 \cdot S_2, 7 \cdot S_7) \tilde{\to} (5 \cdot S'_5, 8 \cdot S'_8)$

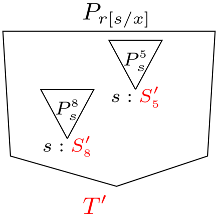
- case $\phi : 2 \mapsto 8,\ 7 \mapsto 5$

$\boxed{\textbf{Assume } S_2 \equiv S_7}$ s.t. $S_2 \equiv S_7 \equiv S'_5 \equiv S'_8$

From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



reduces into



with $(2\cdot S_2, 7\cdot S_7) \equiv (5\cdot S'_5, 8\cdot S'_8)$

ϕ interface $(2\cdot S_2, 7\cdot S_7) \tilde{\to} (5\cdot S'_5, 8\cdot S'_8)$

- case $\phi : 2 \mapsto 5,\ 7 \mapsto 8$

$\boxed{\textbf{Assume } S_2 \equiv S_7}$ s.t. $S_2 \equiv S_7 \equiv S'_5 \equiv S'_8$

Operable Derivations

- hybrid deriv. + interfaces for each $\mathtt{app_h}$-rule = **operable derivation**

Operable Derivations

- hybrid deriv. + interfaces for each $\mathtt{app_h}$-rule = **operable derivation**
- system $\mathtt{S_{op}}$: deterministic with hard-coded red. paths.
  $\mathtt{S}$-derivations: identity interfaces (**trivial** op. deriv.)

Operable Derivations

- hybrid deriv. + interfaces for each $\mathtt{app_h}$-rule = **operable derivation**
- system $\mathtt{S_{op}}$: deterministic with hard-coded red. paths.
  $\mathtt{S}$-derivations: identity interfaces (**trivial** op. deriv.)
- Every $\mathscr{R}$-deriv. Π with a given red. path p can be encoded with a $\mathtt{S}_0$-deriv. P.

$$\textit{multiset} \qquad \Pi \to \Pi_1 \to \Pi_2 \to \dots \to \Pi_n \to \dots$$

Operable Derivations

- hybrid deriv. + interfaces for each $\mathtt{app_h}$-rule = **operable derivation**
- system $\mathtt{S_{op}}$: deterministic with hard-coded red. paths.
  $\mathtt{S}$-derivations: identity interfaces (**trivial** op. deriv.)
- Every $\mathscr{R}$-deriv. Π with a given red. path p can be encoded with a $\mathtt{S}_0$-deriv. P.

$$
\begin{array}{lccccccccccc}
\textit{operable} & P & \to & P_1 & \to & P_2 & \to & \dots & \to & P_n & \to & \dots \\
& \downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \\
\textit{multiset} & \Pi & \to & \Pi_1 & \to & \Pi_2 & \to & \dots & \to & \Pi_n & \to & \dots
\end{array}
$$

Operable Derivations

- hybrid deriv. + interfaces for each $\mathtt{app_h}$-rule = **operable derivation**
- system $\mathtt{S_{op}}$: deterministic with hard-coded red. paths.
  $\mathtt{S}$-derivations: identity interfaces (**trivial** op. deriv.)
- Every $\mathscr{R}$-deriv. Π with a given red. path p can be encoded with a $\mathtt{S}_0$-deriv. P.

$$\begin{array}{lccccccccccc} \textit{operable} & P & \to & P_1 & \to & P_2 & \to & \dots & \to & P_n & \to & \dots \\ & \downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \\ \textit{multiset} & \Pi & \to & \Pi_1 & \to & \Pi_2 & \to & \dots & \to & \Pi_n & \to & \dots \end{array}$$

- Actually, **main theorem**:

$$\begin{array}{lccccccccccc} \textit{trivial} & P & \to & P_1 & \to & P_2 & \to & \dots & \to & P_n & \to & \dots \\ & \downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \\ \textit{multiset} & \Pi & \to & \Pi_1 & \to & \Pi_2 & \to & \dots & \to & \Pi_n & \to & \dots \end{array}$$

Operable Derivations

- hybrid deriv. + interfaces for each $\mathtt{app_h}$-rule = **operable derivation**
- system $\mathtt{S_{op}}$: deterministic with hard-coded red. paths.
  $\mathtt{S}$-derivations: identity interfaces (**trivial** op. deriv.)
- Every $\mathscr{R}$-deriv. Π with a given red. path p can be encoded with a $\mathtt{S}_0$-deriv. P.

$$\begin{array}{lccccccccccc} \textit{operable} & P & \to & P_1 & \to & P_2 & \to & \dots & \to & P_n & \to & \dots \\ & \downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \\ \textit{multiset} & \Pi & \to & \Pi_1 & \to & \Pi_2 & \to & \dots & \to & \Pi_n & \to & \dots \end{array}$$

- Actually, **main theorem**:

$$\begin{array}{lccccccccccc} \textit{trivial} & P & \to & P_1 & \to & P_2 & \to & \dots & \to & P_n & \to & \dots \\ & \downarrow & & \downarrow & & \downarrow & & & & \downarrow & & \\ \textit{multiset} & \Pi & \to & \Pi_1 & \to & \Pi_2 & \to & \dots & \to & \Pi_n & \to & \dots \end{array}$$

- Enough to prove:

$$\boxed{\text{Every operable derivation is isomorphic to a trivial derivation}}$$

iso of op-deriv = nested isos of types commuting with interfaces

Outline

Context

Non-idempotent intersection types

Using type A *once* or *twice* not the same

Rigid vs. non-rigid paradigms

Proof red. deterministic vs. non-deterministic

Question 1

Rigid collapses on non-rigid

Is this collapse surjective?

(A, A, B) and (A, B, A) collapse on $[A, A, B]$

Question 2

In rigid fw., red. paths captured by permutations

$(A, B, A) \mapsto (A, A, B)$

Is it possible to capture red. paths without perm. ?

All this in a coinductive fw. (no productivity)!

Brother threads

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$5 \vdash x : (8{\cdot}o)\to(8{\cdot}o, 9{\cdot}o)\to o'$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$3 \vdash y : o\,[2]$}
\RightLabel{app$_\mathrm{h}$}
\BinaryInfC{$\ldots \vdash x\,y : (8{\cdot}o, 9{\cdot}o)\to o'$}
\RightLabel{abs}
\UnaryInfC{$\ldots \vdash \lambda x.x\,y : (5{\cdot}(8{\cdot}o)\to(8{\cdot}o, 9{\cdot}o)\to o')\to(8{\cdot}o, 9{\cdot}o)\to o'$}
\RightLabel{abs}
\UnaryInfC{$\vdash \lambda yx.x\,y : (7{\cdot}o)\to(5{\cdot}(8{\cdot}o)\to(8{\cdot}o, 9{\cdot}o)\to o')\to(8{\cdot}o, 9{\cdot}o)\to o'$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$4 \vdash z : o\,[3]$}
\RightLabel{app$_\mathrm{h}$}
\BinaryInfC{$\ldots \vdash (\lambda yx.x\,y)z : (5{\cdot}(8{\cdot}o)\to(8{\cdot}o, 9{\cdot}o)\to o')\to(8{\cdot}o, 9{\cdot}o)\to o'$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$2 \vdash a : (\,)\to(3{\cdot}o)\to(2{\cdot}o, 7{\cdot}o)\to o'$}
\RightLabel{app$_\mathrm{h}$}
\UnaryInfC{$\ldots \vdash a\,x : (3{\cdot}o)\to(2{\cdot}o, 7{\cdot}o)\to o'\ [6]$}
\RightLabel{app$_\mathrm{h}$}
\BinaryInfC{$\ldots \vdash ((\lambda yx.xy)z)(a\,x) : (8{\cdot}o, 9{\cdot}o)\to o'$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$4 \vdash b : o\,[3]$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$9 \vdash b : o\,[5]$}
\RightLabel{app$_\mathrm{h}$}
\TrinaryInfC{$\ldots \vdash (((\lambda yx.xy)z)(a\,x))b : o'$}
\end{prooftree}
$$

with *e.g.*, $\phi_1 : \begin{array}{l} 8 \mapsto 2 \\ 9 \mapsto 7 \end{array}$ and $\phi_\varepsilon : \begin{array}{l} 8 \mapsto 5 \\ 9 \mapsto 3 \end{array}$

- One **thread** labelled with 8, one with 9, others with 2, 7, 3 and 5.
- The threads θ_8 and θ_9 are **brothers**.

Brother threads

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{5 \vdash x : (8\cdot o)\to(8\cdot o, 9\cdot o)\to o'}\,\mathsf{ax}
\quad
\dfrac{}{3 \vdash y : o\,[2]}\,\mathsf{ax}
}{\ldots \vdash x\,y : (8\cdot o, 9\cdot o)\to o'}\,\mathsf{app_h}
}{\ldots \vdash \lambda x.x\,y : (5\cdot(8\cdot o)\to(8\cdot o, 9\cdot o)\to o')\to(8\cdot o, 9\cdot o)\to o'}\,\mathsf{abs}
}{\vdash \lambda yx.x\,y : (7\cdot o)\to(5\cdot(8\cdot o)\to(8\cdot o, 9\cdot o)\to o')\to(8\cdot o, 9\cdot o)\to o'}\,\mathsf{abs}
\quad
\dfrac{}{4 \vdash z : o\,[3]}\,\mathsf{ax}
}{\ldots \vdash (\lambda yx.x\,y)z : (5\cdot(8\cdot o)\to(8\cdot o, 9\cdot o)\to o')\to(8\cdot o, 9\cdot o)\to o'}\,\mathsf{app_h}
\quad
\dfrac{
\dfrac{}{2 \vdash a : (\,)\to(3\cdot o)\to(2\cdot o, 7\cdot o)\to o'}\,\mathsf{ax}
}{\ldots \vdash a\,x : (3\cdot o)\to(2\cdot o, 7\cdot o)\to o'\,[6]}\,\mathsf{app_h}
}{\ldots \vdash ((\lambda yx.xy)z)(a\,x) : (8\cdot o, 9\cdot o)\to o'}\,\mathsf{app_h}
\quad
\dfrac{}{4 \vdash b : o\,[3]}\,\mathsf{ax}
\quad
\dfrac{}{9 \vdash b : o\,[5]}\,\mathsf{ax}
}{\ldots \vdash (((\lambda yx.xy)z)(a\,x))b : o'}\,\mathsf{app_h}
$$

with *e.g.*, $\phi_1 : \begin{array}{l} 8 \mapsto 2 \\ 9 \mapsto 7 \end{array}$ and $\phi_\varepsilon : \begin{array}{l} 8 \mapsto 5 \\ 9 \mapsto 3 \end{array}$

- One **thread** labelled with 8, one with 9, others with 2, 7, 3 and 5.
- The threads θ_8 and θ_9 are **brothers**.
- Positive and negative parts in θ_8 and θ_9.
  - Positive: ascend to `ax`.
  - Negative: ascend to `abs`.

Brother threads

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{5 \vdash x : (8{\cdot}o)\to(8{\cdot}o, 9{\cdot}o)\to o'}\,\mathsf{ax}
\quad
\dfrac{}{3 \vdash y : o\,[2]}\,\mathsf{ax}
}{\ldots \vdash x\,y : (8{\cdot}o, 9{\cdot}o)\to o'}\,\mathsf{app_h}
}{\ldots \vdash \lambda x.x\,y : (5{\cdot}(8{\cdot}o)\to(8{\cdot}o, 9{\cdot}o)\to o')\to(8{\cdot}o, 9{\cdot}o)\to o'}\,\mathsf{abs}
}{\vdash \lambda yx.x\,y : (7{\cdot}o)\to(5{\cdot}(8{\cdot}o)\to(8{\cdot}o, 9{\cdot}o)\to o')\to(8{\cdot}o, 9{\cdot}o)\to o'}\,\mathsf{abs}
\quad
\dfrac{}{4 \vdash z : o\,[3]}\,\mathsf{ax}
}{\ldots \vdash (\lambda yx.x\,y)z : (5{\cdot}(8{\cdot}o)\to(8{\cdot}o, 9{\cdot}o)\to o')\to(8{\cdot}o, 9{\cdot}o)\to o'}\,\mathsf{app_h}
\quad
\dfrac{
\dfrac{}{2 \vdash a : (\,)\to(3{\cdot}o)\to(2{\cdot}o, 7{\cdot}o)\to o'}\,\mathsf{ax}
}{\ldots \vdash a\,x : (3{\cdot}o)\to(2{\cdot}o, 7{\cdot}o)\to o'\ [6]}\,\mathsf{app_h}
}{\ldots \vdash ((\lambda yx.xy)z)(a\,x) : (8{\cdot}o, 9{\cdot}o)\to o'}\,\mathsf{app_h}
\quad
\dfrac{}{4 \vdash b : o\,[3]}\,\mathsf{ax}
\quad
\dfrac{}{9 \vdash b : o\,[5]}\,\mathsf{ax}
}{\ldots \vdash (((\lambda yx.xy)z)(a\,x))b : o'}\,\mathsf{app_h}
$$

with *e.g.*, $\phi_1 : \begin{array}{l} 8 \mapsto 2 \\ 9 \mapsto 7 \end{array}$ and $\phi_\varepsilon : \begin{array}{l} 8 \mapsto 5 \\ 9 \mapsto 3 \end{array}$

- One **thread** labelled with 8, one with 9, others with 2, 7, 3 and 5.
- The threads θ_8 and θ_9 are **brothers**.
- Positive and negative parts in θ_8 and θ_9.
  - Positive: ascend to `ax`.
  - Negative: ascend to `abs`.
- **Consumption** in $\mathsf{app_h}$-rules *e.g.*, $\theta_8{}^{\ominus}\tilde{\to}\theta_2$, $\theta_8{}^{\oplus}\tilde{\to}\theta_5$

Brother threads

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$5 \vdash x : (8{\cdot}o) \to (8{\cdot}o, 9{\cdot}o) \to o'$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$3 \vdash y : o\,[2]$}
\RightLabel{$\mathtt{app_h}$}
\BinaryInfC{$\ldots \vdash x\,y : (8{\cdot}o, 9{\cdot}o) \to o'$}
\RightLabel{abs}
\UnaryInfC{$\ldots \vdash \lambda x.x\,y : (5{\cdot}(8{\cdot}o) \to (8{\cdot}o, 9{\cdot}o) \to o') \to (8{\cdot}o, 9{\cdot}o) \to o'$}
\RightLabel{abs}
\UnaryInfC{$\vdash \lambda yx.x\,y : (7{\cdot}o) \to (5{\cdot}(8{\cdot}o) \to (8{\cdot}o, 9{\cdot}o) \to o') \to (8{\cdot}o, 9{\cdot}o) \to o'$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$4 \vdash z : o\,[3]$}
\RightLabel{$\mathtt{app_h}$}
\BinaryInfC{$\ldots \vdash (\lambda yx.x\,y)z : (5{\cdot}(8{\cdot}o) \to (8{\cdot}o, 9{\cdot}o) \to o') \to (8{\cdot}o, 9{\cdot}o) \to o'$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$2 \vdash a : (\,) \to (3{\cdot}o) \to (2{\cdot}o, 7{\cdot}o) \to o'$}
\RightLabel{$\mathtt{app_h}$}
\UnaryInfC{$\ldots \vdash a\,x : (3{\cdot}o) \to (2{\cdot}o, 7{\cdot}o) \to o'\,[6]$}
\RightLabel{$\mathtt{app_h}$}
\BinaryInfC{$\ldots \vdash ((\lambda yx.xy)z)(a\,x) : (8{\cdot}o, 9{\cdot}o) \to o'$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$4 \vdash b : o\,[3]$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$9 \vdash b : o\,[5]$}
\RightLabel{$\mathtt{app_h}$}
\TrinaryInfC{$\ldots \vdash (((\lambda yx.xy)z)(a\,x))b : o'$}
\end{prooftree}
$$

with *e.g.*, $\phi_1 : \begin{array}{l} 8 \mapsto 2 \\ 9 \mapsto 7 \end{array}$ and $\phi_\varepsilon : \begin{array}{l} 8 \mapsto 5 \\ 9 \mapsto 3 \end{array}$

- One **thread** labelled with 8, one with 9, others with 2, 7, 3 and 5.
- The threads θ_8 and θ_9 are **brothers**.
- Positive and negative parts in θ_8 and θ_9.
  - Positive: ascend to `ax`.
  - Negative: ascend to `abs`.
- **Consumption** in $\mathtt{app_h}$-rules *e.g.*, $\theta_8 \overset{\ominus}{\tilde{\to}} \theta_2$, $\theta_8 \overset{\oplus}{\tilde{\to}} \theta_5$
- To have a trivial deriv., one must choose a new value $\mathtt{lab}(\theta)$ for each thread s.t.:

Brother threads

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{5 \vdash x : (8\cdot o)\to(8\cdot o, 9\cdot o)\to o'}\,\mathsf{ax}
\quad
\dfrac{}{3 \vdash y : o\,[2]}\,\mathsf{ax}
}{\ldots \vdash x\,y : (8\cdot o, 9\cdot o)\to o'}\,\mathsf{app_h}
}{\ldots \vdash \lambda x.x\,y : (5\cdot(8\cdot o)\to(8\cdot o, 9\cdot o)\to o')\to(8\cdot o, 9\cdot o)\to o'}\,\mathsf{abs}
}{\vdash \lambda yx.x\,y : (7\cdot o)\to(5\cdot(8\cdot o)\to(8\cdot o, 9\cdot o)\to o')\to(8\cdot o, 9\cdot o)\to o'}\,\mathsf{abs}
\quad
\dfrac{}{4 \vdash z : o\,[3]}\,\mathsf{ax}
}{\ldots \vdash (\lambda yx.x\,y)z : (5\cdot(8\cdot o)\to(8\cdot o, 9\cdot o)\to o')\to(8\cdot o, 9\cdot o)\to o'}\,\mathsf{app_h}
\quad
\dfrac{
\dfrac{}{2 \vdash a : (\,)\to(3\cdot o)\to(2\cdot o, 7\cdot o)\to o'}\,\mathsf{ax}
}{\ldots \vdash a\,x : (3\cdot o)\to(2\cdot o, 7\cdot o)\to o'\;[6]}\,\mathsf{app_h}
}{\ldots \vdash ((\lambda yx.xy)z)(a\,x) : (8\cdot o, 9\cdot o)\to o'}\,\mathsf{app_h}
\quad
\dfrac{}{4 \vdash b : o\,[3]}\,\mathsf{ax}
\quad
\dfrac{}{9 \vdash b : o\,[5]}\,\mathsf{ax}
}{\ldots \vdash (((\lambda yx.xy)z)(a\,x))b : o'}\,\mathsf{app_h}
$$

with *e.g.*, $\phi_1 : \begin{array}{l} 8 \mapsto 2 \\ 9 \mapsto 7 \end{array}$ and $\phi_\varepsilon : \begin{array}{l} 8 \mapsto 5 \\ 9 \mapsto 3 \end{array}$

- One **thread** labelled with 8, one with 9, others with 2, 7, 3 and 5.
- The threads θ_8 and θ_9 are **brothers**.
- Positive and negative parts in θ_8 and θ_9.
  - Positive: ascend to `ax`.
  - Negative: ascend to `abs`.
- **Consumption** in $\mathtt{app_h}$-rules *e.g.*, $\theta_8{}^{\ominus}\tilde{\rightarrow}\theta_2$, $\theta_8{}^{\oplus}\tilde{\rightarrow}\theta_5$
- To have a trivial deriv., one must choose a new value $\mathtt{lab}(\theta)$ for each thread s.t.:
  - $\mathtt{lab}(\theta_8) = \mathtt{lab}(\theta_2)$, $\mathtt{lab}(\theta_9) = \mathtt{lab}(\theta_7)$, $\mathtt{lab}(\theta_8) = \mathtt{lab}(\theta_5)$, $\mathtt{lab}(\theta_9) = \mathtt{lab}(\theta_3)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{5 \vdash x : (8{\cdot}o)\to(8{\cdot}o, 9{\cdot}o)\to o'}\,\mathsf{ax} \quad \dfrac{}{3 \vdash y : o\,[2]}\,\mathsf{ax}}{\ldots \vdash x\,y : (8{\cdot}o, 9{\cdot}o)\to o'}\,\mathsf{app_h}}{\ldots \vdash \lambda x.x\,y : (5{\cdot}(8{\cdot}o)\to(8{\cdot}o, 9{\cdot}o)\to o')\to(8{\cdot}o, 9{\cdot}o)\to o'}\,\mathsf{abs}}{\vdash \lambda yx.x\,y : (7{\cdot}o)\to(5{\cdot}(8{\cdot}o)\to(8{\cdot}o, 9{\cdot}o)\to o')\to(8{\cdot}o, 9{\cdot}o)\to o'}\,\mathsf{abs} \quad \dfrac{}{4 \vdash z : o\,[3]}\,\mathsf{ax}}{\ldots \vdash (\lambda yx.x\,y)z : (5{\cdot}(8{\cdot}o)\to(8{\cdot}o, 9{\cdot}o)\to o')\to(8{\cdot}o, 9{\cdot}o)\to o'}\,\mathsf{app_h}$$

$$\dfrac{\dfrac{}{2 \vdash a : (\,)\to(3{\cdot}o)\to(2{\cdot}o, 7{\cdot}o)\to o'}\,\mathsf{ax}}{\ldots \vdash a\,x : (3{\cdot}o)\to(2{\cdot}o, 7{\cdot}o)\to o'\ [6]}\,\mathsf{app_h}$$

$$\dfrac{\dfrac{\ldots \vdash (\lambda yx.x\,y)z : \ldots \quad \ldots \vdash a\,x : \ldots}{\ldots \vdash ((\lambda yx.xy)z)(a\,x) : (8{\cdot}o, 9{\cdot}o)\to o'}\,\mathsf{app_h} \quad \dfrac{}{4 \vdash b : o\,[3]}\,\mathsf{ax} \quad \dfrac{}{9 \vdash b : o\,[5]}\,\mathsf{ax}}{\ldots \vdash (((\lambda yx.xy)z)(a\,x))b : o'}\,\mathsf{app_h}$$

with *e.g.*, $\phi_1 : \begin{array}{l} 8 \mapsto 2 \\ 9 \mapsto 7 \end{array}$ and $\phi_\varepsilon : \begin{array}{l} 8 \mapsto 5 \\ 9 \mapsto 3 \end{array}$

- One **thread** labelled with 8, one with 9, others with 2, 7, 3 and 5.
- The threads θ_8 and θ_9 are **brothers**.
- Positive and negative parts in θ_8 and θ_9.
  - Positive: ascend to `ax`.
  - Negative: ascend to `abs`.
- **Consumption** in $\mathtt{app_h}$-rules *e.g.*, $\theta_8{}^{\ominus}\tilde{\to}\theta_2$, $\theta_8{}^{\oplus}\tilde{\to}\theta_5$
- To have a trivial deriv., one must choose a new value $\mathtt{lab}(\theta)$ for each thread s.t.:
  - $\mathtt{lab}(\theta_8) = \mathtt{lab}(\theta_2)$, $\mathtt{lab}(\theta_9) = \mathtt{lab}(\theta_7)$, $\mathtt{lab}(\theta_8) = \mathtt{lab}(\theta_5)$, $\mathtt{lab}(\theta_9) = \mathtt{lab}(\theta_3)$
  - No overlap: $\mathtt{lab}(\theta_8) \neq \mathtt{lab}(\theta_9)$, $\mathtt{lab}(\theta_2) \neq \mathtt{lab}(\theta_7)$, $\mathtt{lab}(\theta_3) \neq \mathtt{lab}(\theta_5)$

Milestones of the main proof

- **Prop:** let P be an op. deriv. If the interface of P does not prove an eq. of the form $\mathtt{lab}(\theta_{\mathtt{bro}_1}) = \mathtt{lab}(\theta_{\mathtt{bro}_2})$, then P is isomorphic to a trivial deriv.

Milestones of the main proof

- **Prop:** let P be an op. deriv. If the interface of P does not prove an eq. of the form $\mathtt{lab}(\theta_{\mathtt{bro}_1}) = \mathtt{lab}(\theta_{\mathtt{bro}_2})$, then P is isomorphic to a trivial deriv.

- Ad absurdum, assume that such a proof exist.

Milestones of the main proof

- **Prop:** let P be an op. deriv. If the interface of P does not prove an eq. of the form $\mathtt{lab}(\theta_{\mathtt{bro}_1}) = \mathtt{lab}(\theta_{\mathtt{bro}_2})$, then P is isomorphic to a trivial deriv.

- Ad absurdum, assume that such a proof exist.
  1. Proof of the form $\theta_{\mathtt{bro}_1}(\tilde{\leftarrow} \cup \tilde{\rightarrow})\theta_2(\tilde{\leftarrow} \cup \tilde{\rightarrow}) \ldots \theta_{n-1}(\tilde{\leftarrow} \cup \tilde{\rightarrow})\theta_{\mathtt{bro}_2}$

Milestones of the main proof

- **Prop:** let P be an op. deriv. If the interface of P does not prove an eq. of the form $\mathtt{lab}(\theta_{\mathtt{bro}_1}) = \mathtt{lab}(\theta_{\mathtt{bro}_2})$, then P is isomorphic to a trivial deriv.

- Ad absurdum, assume that such a proof exist.
  1. Proof of the form $\theta_{\mathtt{bro}_1}(\tilde{\leftarrow} \cup \tilde{\rightarrow})\theta_2(\tilde{\leftarrow} \cup \tilde{\rightarrow})\ldots\theta_{n-1}(\tilde{\leftarrow} \cup \tilde{\rightarrow})\theta_{\mathtt{bro}_2}$
  2. Up to a *finite* number of red. steps, then $\theta'_{\mathtt{bro}_1} {}^{\oplus}\tilde{\rightarrow} \theta'_2 {}^{\oplus}\tilde{\rightarrow} \ldots \theta'_{\ell-1} {}^{\oplus}\tilde{\rightarrow} \theta'_{\mathtt{bro}_2}$ with $\ell \leqslant n$

Milestones of the main proof

- **Prop:** let P be an op. deriv. If the interface of P does not prove an eq. of the form $\mathtt{lab}(\theta_{\mathtt{bro}_1}) = \mathtt{lab}(\theta_{\mathtt{bro}_2})$, then P is isomorphic to a trivial deriv.

- Ad absurdum, assume that such a proof exist.
  1. Proof of the form $\theta_{\mathtt{bro}_1}(\tilde{\leftarrow} \cup \tilde{\rightarrow})\theta_2(\tilde{\leftarrow} \cup \tilde{\rightarrow}) \ldots \theta_{n-1}(\tilde{\leftarrow} \cup \tilde{\rightarrow})\theta_{\mathtt{bro}_2}$
  2. Up to a *finite* number of red. steps, then $\theta'_{\mathtt{bro}_1}{}^{\oplus}\tilde{\rightarrow}\theta'_2{}^{\oplus}\tilde{\rightarrow} \ldots \theta'_{\ell-1}{}^{\oplus}\tilde{\rightarrow}\theta'_{\mathtt{bro}_2}$ with $\ell \leqslant n$
  3. *Lem:* if $\theta_a{}^{\oplus}\tilde{\rightarrow}\theta_b$ then $\mathtt{ad}(\theta_a) < \mathtt{ad}(\theta_b)$ (applicative depth)

- **Prop:** let P be an op. deriv. If the interface of P does not prove an eq. of the form $\mathtt{lab}(\theta_{\mathtt{bro}_1}) = \mathtt{lab}(\theta_{\mathtt{bro}_2})$, then P is isomorphic to a trivial deriv.

- Ad absurdum, assume that such a proof exist.
  1. Proof of the form $\theta_{\mathtt{bro}_1}(\tilde{\leftarrow} \cup \tilde{\rightarrow})\theta_2(\tilde{\leftarrow} \cup \tilde{\rightarrow}) \dots \theta_{n-1}(\tilde{\leftarrow} \cup \tilde{\rightarrow})\theta_{\mathtt{bro}_2}$
  2. Up to a *finite* number of red. steps, then $\theta'_{\mathtt{bro}_1} {}^{\oplus}\tilde{\rightarrow} \theta'_2 {}^{\oplus}\tilde{\rightarrow} \dots \theta'_{\ell-1} {}^{\oplus}\tilde{\rightarrow} \theta'_{\mathtt{bro}_2}$ with $\ell \leqslant n$
  3. *Lem:* if $\theta_a {}^{\oplus}\tilde{\rightarrow} \theta_b$ then $\mathtt{ad}(\theta_a) < \mathtt{ad}(\theta_b)$ (applicative depth)
  4. Then $\mathtt{ad}(\theta'_{\mathtt{bro}_1}) < \mathtt{ad}(\theta'_{\mathtt{bro}_2})$.

Milestones of the main proof

- **Prop:** let P be an op. deriv. If the interface of P does not prove an eq. of the form $\mathtt{lab}(\theta_{\mathtt{bro}_1}) = \mathtt{lab}(\theta_{\mathtt{bro}_2})$, then P is isomorphic to a trivial deriv.

- Ad absurdum, assume that such a proof exist.
  1. Proof of the form $\theta_{\mathtt{bro}_1}(\tilde{\leftarrow} \cup \tilde{\rightarrow})\theta_2(\tilde{\leftarrow} \cup \tilde{\rightarrow})\ldots\theta_{n-1}(\tilde{\leftarrow} \cup \tilde{\rightarrow})\theta_{\mathtt{bro}_2}$
  2. Up to a *finite* number of red. steps, then $\theta'_{\mathtt{bro}_1}{}^{\oplus}\tilde{\rightarrow}\theta'_2{}^{\oplus}\tilde{\rightarrow}\ldots\theta'_{\ell-1}{}^{\oplus}\tilde{\rightarrow}\theta'_{\mathtt{bro}_2}$ with $\ell \leqslant n$
  3. *Lem:* if $\theta_a{}^{\oplus}\tilde{\rightarrow}\theta_b$ then $\mathtt{ad}(\theta_a) < \mathtt{ad}(\theta_b)$ (applicative depth)
  4. Then $\mathtt{ad}(\theta'_{\mathtt{bro}_1}) < \mathtt{ad}(\theta'_{\mathtt{bro}_2})$.

  Absurd (for two brother threads).

Milestones of the main proof

- **Prop:** let P be an op. deriv. If the interface of P does not prove an eq. of the form $\mathtt{lab}(\theta_{\mathtt{bro}_1}) = \mathtt{lab}(\theta_{\mathtt{bro}_2})$, then P is isomorphic to a trivial deriv.

- Ad absurdum, assume that such a proof exist.
  1. Proof of the form $\theta_{\mathtt{bro}_1}(\tilde{\leftarrow} \cup \tilde{\rightarrow})\theta_2(\tilde{\leftarrow} \cup \tilde{\rightarrow})\ldots\theta_{n-1}(\tilde{\leftarrow} \cup \tilde{\rightarrow})\theta_{\mathtt{bro}_2}$
  2. Up to a *finite* number of red. steps, then $\theta'_{\mathtt{bro}_1}{}^{\oplus}\tilde{\rightarrow}\theta'_2{}^{\oplus}\tilde{\rightarrow}\ldots\theta'_{\ell-1}{}^{\oplus}\tilde{\rightarrow}\theta'_{\mathtt{bro}_2}$ with $\ell \leqslant n$
  3. *Lem:* if $\theta_a{}^{\oplus}\tilde{\rightarrow}\theta_b$ then $\mathtt{ad}(\theta_a) < \mathtt{ad}(\theta_b)$ (applicative depth)
  4. Then $\mathtt{ad}(\theta'_{\mathtt{bro}_1}) < \mathtt{ad}(\theta'_{\mathtt{bro}_2})$.

  Absurd (for two brother threads).

- By the above prop:

$$\boxed{\text{Every operable deriv. is isomorphic to a } \textit{trivial} \text{ deriv.}}$$

Milestones of the main proof

- **Prop:** let P be an op. deriv. If the interface of P does not prove an eq. of the form $\mathtt{lab}(\theta_{\mathtt{bro}_1}) = \mathtt{lab}(\theta_{\mathtt{bro}_2})$, then P is isomorphic to a trivial deriv.

- Ad absurdum, assume that such a proof exist.
  1. Proof of the form $\theta_{\mathtt{bro}_1}(\tilde{\leftarrow} \cup \tilde{\rightarrow})\theta_2(\tilde{\leftarrow} \cup \tilde{\rightarrow})\ldots\theta_{n-1}(\tilde{\leftarrow} \cup \tilde{\rightarrow})\theta_{\mathtt{bro}_2}$
  2. Up to a *finite* number of red. steps, then $\theta'_{\mathtt{bro}_1}{}^{\oplus}\tilde{\rightarrow}\theta'_2{}^{\oplus}\tilde{\rightarrow}\ldots\theta'_{\ell-1}{}^{\oplus}\tilde{\rightarrow}\theta'_{\mathtt{bro}_2}$ with $\ell \leqslant n$
  3. *Lem:* if $\theta_a{}^{\oplus}\tilde{\rightarrow}\theta_b$ then $\mathtt{ad}(\theta_a) < \mathtt{ad}(\theta_b)$ (applicative depth)
  4. Then $\mathtt{ad}(\theta'_{\mathtt{bro}_1}) < \mathtt{ad}(\theta'_{\mathtt{bro}_2})$.

  Absurd (for two brother threads).

- By the above prop:

$$\boxed{\text{Every operable deriv. is isomorphic to a \textit{trivial} deriv.}}$$

Theorem

- Every $\mathscr{R}$-deriv. Π is the collapse of a $\mathtt{S}$-deriv. P
- Every red. path starting from Π can be encoded in such a P.

Plan

Summary

- Any dynamic behavior in $\mathscr{R}$ (multiset inter.) can be individually represented in $\mathtt{S}$ (sequence inter.)
- Existence of an intermediary system $\mathtt{S}_{\mathtt{op}}$, close to other formalisms (Gardner, Tsukada et al.)
- Every point of the infinitary relational model can studied thtroug a representant in system $\mathtt{S}$.
- Emancipation from productivity.

Want the details?

- Phd dissertation, chapter 13

Thank you for your attention!

Save the date(s):

Types	Braga	21th june	The infinitary relational model
HOR (Floc)	Oxford	7th july	Some aspects of intersection types (invited talk)
LiCS (Floc)	Oxford	9th july	The infinitary relational model