Sequence Types for Hereditary Permutators

Pierre VIAL
Équipe Gallinette
Inria - LS2N

June 25, 2019

TLCA Problem # 20

Characterization of a set of terms
with an intersection type system

TLCA Problem # 20

Characterization of a set of terms with an intersection type system

more precisely, characterizing this set with a unique type

TLCA Problem # 20

Characterization of a set of terms with an intersection type system

more precisely, characterizing this set with a unique type

Hereditary permutators ("invertible" terms)

- Curry-Feys 58
- Dezani 76
- Bergstra-Klop 80

TLCA Problem # 20

Characterization of a set of terms with an intersection type system

more precisely, characterizing this set with a unique type

Hereditary permutators ("invertible" terms)

- Curry-Feys 58
- Dezani 76
- Bergstra-Klop 80

[**Tatsuta 08**] inductive case: not possible

[V. 17] coinductive type system can characterize infinitary semantics

coinductive type grammar

TLCA Problem # 20

Characterization of a set of terms with an intersection type system

more precisely, characterizing this set with a unique type

Hereditary permutators
("invertible" terms)

- Curry-Feys 58
- Dezani 76
- Bergstra-Klop 80

[**Tatsuta 08**] inductive case: not possible

[V. 17] coinductive type system can characterize infinitary semantics

coinductive type grammar

Today: using a coinductive type system to characterize hereditary permutators

TLCA Problem # 20

Characterization of a set of terms with an intersection type system

more precisely, characterizing this set with a unique type

Hereditary permutators
("invertible" terms)

- Curry-Feys 58
- Dezani 76
- Bergstra-Klop 80

[**Tatsuta 08**] inductive case: not possible

[V. 17] coinductive type system can characterize infinitary semantics

coinductive type grammar

Today: using a coinductive type system to characterize hereditary permutators

t is a h.p. iff $\vdash t : \texttt{ptyp}$

Definition (Hereditary Permutators)

t is a **hereditary permutator (h.p.)**

$\Leftrightarrow$ t invertible in Scott's model

$\Leftrightarrow$ t invertible for $\beta\eta$-conversion w.r.t. composition ($\exists u,\ t \circ u =_{\beta\eta} u \circ t =_{\beta\eta} \mathtt{I}$)

$t \circ u := \lambda x.t(u\,x)$

An old problem...

Definition (Hereditary Permutators)

t is a **hereditary permutator (h.p.)**
$\Leftrightarrow$ t invertible in Scott's model
$\Leftrightarrow$ t invertible for $\beta\eta$-conversion w.r.t. composition ($\exists u,\ t \circ u =_{\beta\eta} u \circ t =_{\beta\eta} \mathtt{I}$)

$t \circ u := \lambda x.t(u\,x)$

Characterization with Böhm trees

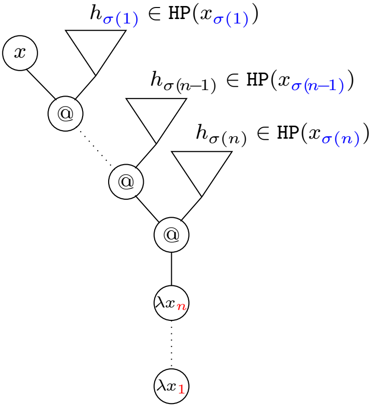
- For all $x \in \mathscr{V}$, the sets $\mathtt{HP}(x)$ of **x-headed Hereditary Permutators (x-HP)** ($x \in \mathscr{V}$) are defined by mutual coinduction:

$$\frac{h_1 \in \mathtt{HP}(x_1)\ \ldots\ h_n \in \mathtt{HP}(x_n) \quad (n \geqslant 0, \sigma \in \mathfrak{S}_n,\ x_i \neq x,\ x_i \text{ pairwise distinct}) \text{ and } h \to_{\mathtt{h}}^* \lambda x_1 \ldots x_n.x\, h_{\sigma(1)} \ldots h_{\sigma(n)}}{h \in \mathtt{HP}(x)}$$

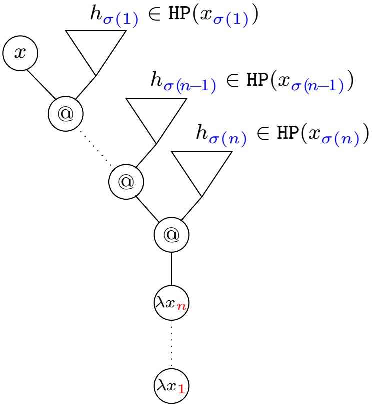
- t is a *(closed) hereditary permutator* iff $t \to_{\mathtt{h}}^* \lambda x.h$ with $h \in \mathtt{HP}(x)$ for some x.

See Barendregt, Chapter 21

An old problem. . .



head normal form of a x-h.p.



head normal form of a x-h.p.

Plan

1. Intersection types and sequences

2. Characterizing hereditary permutators

Intersection types (overview)

- Introduced by **Coppo-Dezani** (78-80) to “interpret more terms”
  - Charac. of Weak Norm. for λI-terms (no erasing β-step).
  - Extended later for λ-terms, head, weak or strong normalizatiion...
  - Filter models
- Model-checking
  - *Ong 06:* monadic second order (MSO) logic is decidable for higher-order recursion schemes (HORS)
  - *Kobayashi-Ong 09:* MSO is decidable for higher-order programs
    + using intersection types to simplify Ong's algorithm.
  - Refined by *Grellois-Melliès 14-15*
- Complexity analysis:
  - Upper bounds for reduction sequences (*Gardner 94, de Carvalho 07*) or exact bounds (*Bernadet-Lengrand 11, Accattoli-Lengrand-Kesner, ICFP'18*).
  - *Terui 06:* upper bounds for terms in a red. sequence
  - *De Benedetti-Ronchi della Roccha 16*: characterization of FPTIME

Plan

Goal: Characterizing the set of h.p. with a *unique* type

Plan

Goal: Characterizing the set of h.p. with a *unique* type

Problem: Impossible with an inductive inter. type system [Tatsuta, LiCS'08]

Plan

Goal: Characterizing the set of h.p. with a *unique* type

Problem: Impossible with an inductive inter. type system [Tatsuta, LiCS'08]

Solution? coinductive type system may characterize sets of terms with an infinitary behavior [V.,17]

$\rightsquigarrow$ *e.g.*, the set of **infinitary WN** terms

(system $\mathtt{S}$, sequential intersection)

Plan

Goal: Characterizing the set of h.p. with a *unique* type

Problem: Impossible with an inductive inter. type system [Tatsuta, LiCS'08]

Solution? coinductive type system may characterize sets of terms with an infinitary behavior [V.,17]

$\rightsquigarrow$ *e.g.*, the set of **infinitary WN** terms

(system $\mathtt{S}$, sequential intersection)

Step 1: characterize the set of h.p. in system $\mathtt{S}$

$\rightsquigarrow$ find a set of types $\mathscr{P}$ s.t.

t typable with $P \in \mathscr{P}$ iff t is a h.p.

Plan

Goal: Characterizing the set of h.p. with a *unique* type

Problem: Impossible with an inductive inter. type system [Tatsuta, LiCS'08]

Solution? coinductive type system may characterize sets of terms with an infinitary behavior [V.,17]

$\rightsquigarrow$ *e.g.*, the set of **infinitary WN** terms

(system $\mathtt{S}$, sequential intersection)

Step 1: characterize the set of h.p. in system $\mathtt{S}$

$\rightsquigarrow$ find a set of types $\mathscr{P}$ s.t.

t typable with $P \in \mathscr{P}$ iff t is a h.p.

Step 2: give the set of h.p. a *unique* type

$\rightsquigarrow$ quotient $\mathscr{P}$ and verify everything is right

Why intersections types? (Coppo, Dezani, 1980)

Characterization in an intersection type system

Usually, equivalences of the form

"the program t is typable iff t is normalizing"

Idea: **several** certificates to a same subprogram (next slide).

Why intersections types? (Coppo, Dezani, 1980)

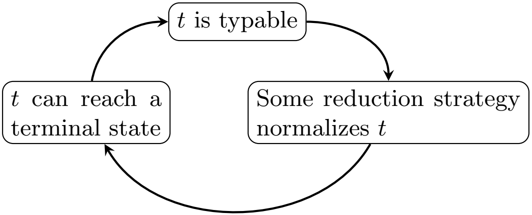
Characterization in an intersection type system

Usually, equivalences of the form

"the program t is typable iff t is normalizing"

Idea: **several** certificates to a same subprogram (next slide).

Proof. Charac. obtained by by the "circular" proof scheme:



Why intersections types? (Coppo, Dezani, 1980)

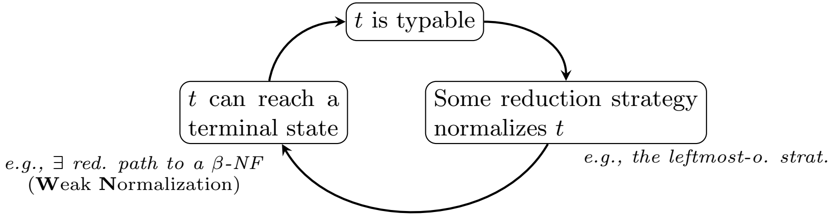
Characterization in an intersection type system

Usually, equivalences of the form

"the program t is typable iff t is normalizing"

Idea: **several** certificates to a same subprogram (next slide).

Proof. Charac. obtained by by the "circular" proof scheme:



- Naively, $A \wedge B$ stands for $A \cap B$:

$$t \textit{ is of type } A \wedge B \textit{ if } t \textit{ can be typed with } A \textit{ as well as } B.$$

$$\frac{I : A \to A \qquad I : (A \to B) \to (A \to B)}{I : (A \to A) \wedge ((A \to B) \to (A \to B))} \wedge-\texttt{intro} \qquad \textit{(with } I = \lambda x.x\textit{)}$$

Intuitions (syntax)

- Naively, $A \wedge B$ stands for $A \cap B$:

  t is of type $A \wedge B$ if t can be typed with A as well as B.

$$\frac{I : A \to A \qquad I : (A \to B) \to (A \to B)}{I : (A \to A) \wedge ((A \to B) \to (A \to B))} \wedge-\texttt{intro} \quad \textit{(with } I = \lambda x.x\textit{)}$$

- Intersection = kind of *finite polymorphism.*

$$(A \to A) \wedge ((A \to B) \to (A \to B)) = \textbf{double} \text{ instance of } \forall X.X \to X$$

(with $X = A$ and $X = A \to B$)

- Naively, $A \wedge B$ stands for $A \cap B$:

t is of type $A \wedge B$ if t can be typed with A as well as B.

$$\frac{I : A \to A \qquad I : (A \to B) \to (A \to B)}{I : (A \to A) \wedge ((A \to B) \to (A \to B))} \wedge-\texttt{intro} \qquad \textit{(with } I = \lambda x.x\textit{)}$$

- Intersection = kind of *finite polymorphism*.

$(A \to A) \wedge ((A \to B) \to (A \to B))$ = **double** instance of $\forall X.X \to X$

(with $X = A$ and $X = A \to B$)

- But *less constrained*:

assigning $x : o \wedge (o \to o') \wedge (o \to o) \to o$ is legal.

*(**not** an instance of a polymorphic type except $\forall X.X :=$ `False`!)*

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

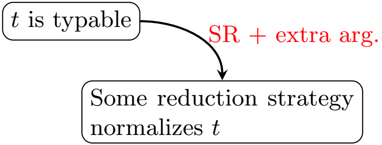
Subject Reduction and Subject Expansion

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

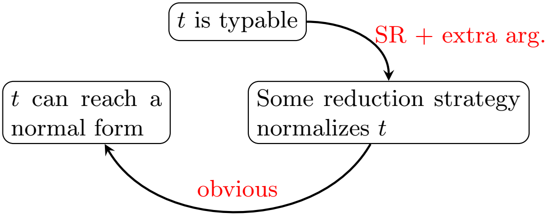


A good intersection type system should enjoy:

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



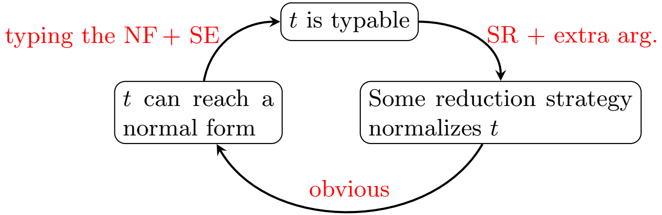
Subject Reduction and Subject Expansion

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



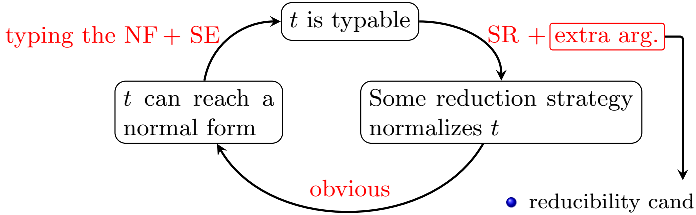
Subject Reduction and Subject Expansion

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



- reducibility cand.
- non-trivial well-founded order.
- non-idem. inter types: trivial
  size of derivation decreases during red.

Intersection types (Coppo-Dezani 80)

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

Intersection types (Coppo-Dezani 80)

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

- **Strict types**:
  no inter. 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* $\wedge$

Intersection types (Coppo-Dezani 80)

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

- **Strict types**:
  no inter. 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* $\wedge$

Assoc.: $(A \wedge B) \wedge C \sim A \wedge (B \wedge C)$

i.e. $\Gamma \vdash t : (A \wedge B) \wedge C$ iff $\Gamma \vdash t : A \wedge (B \wedge C)$

Comm.: $A \wedge B \sim B \wedge A$

Intersection types (Coppo-Dezani 80)

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

- **Strict types**:
  no inter. 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* $\wedge$

Assoc.: $(A \wedge B) \wedge C \sim A \wedge (B \wedge C)$

i.e. $\Gamma \vdash t : (A \wedge B) \wedge C$ iff $\Gamma \vdash t : A \wedge (B \wedge C)$

Comm.: $A \wedge B \sim B \wedge A$

Idempotency? $A \wedge A \sim A$

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

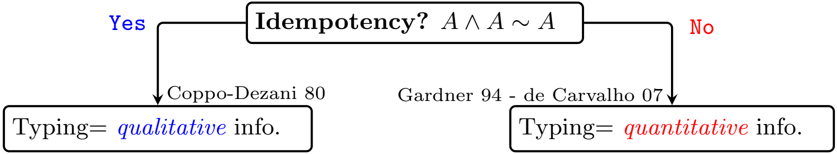
- **Strict types**:
  no inter. on the *right* h.s. of $\to$, *e.g.*, $(A \wedge B) \to A$, not $A \to (B \wedge C)$

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

Assoc.: $(A \wedge B) \wedge C \sim A \wedge (B \wedge C)$

i.e. $\Gamma \vdash t : (A \wedge B) \wedge C$ iff $\Gamma \vdash t : A \wedge (B \wedge C)$

Comm.: $A \wedge B \sim B \wedge A$



Intersection types (Coppo-Dezani 80)

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

- **Strict types**:
  no inter. 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* $\wedge$

Assoc.: $(A \wedge B) \wedge C \sim A \wedge (B \wedge C)$

i.e. $\Gamma \vdash t : (A \wedge B) \wedge C$ iff $\Gamma \vdash t : A \wedge (B \wedge C)$

Comm.: $A \wedge B \sim B \wedge A$

Idempotency? $A \wedge A \sim A$

Yes → Coppo-Dezani 80 → Typing= *qualitative* info.

No → Gardner 94 - de Carvalho 07 → Typing= *quantitative* info.

- Collapsing $A \wedge B \wedge C$ into $[A, B, C]$ (**multiset**) $\rightsquigarrow$ no need for perm rules etc.

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

$$\text{Types:} \qquad \tau,\ \sigma \quad ::= \quad o \quad | \quad [\sigma_i]_{i \in I} \rightarrow \tau$$

- **intersection** = **multiset** of types $[\sigma_i]_{i \in I}$
- only on the left-h.s of $\rightarrow$ (*strictness*)

Types: $\tau,\ \sigma \quad ::= \quad o \quad | \quad [\sigma_i]_{i \in I} \to \tau$

- **intersection** = **multiset** of types $[\sigma_i]_{i \in I}$
- only on the left-h.s of $\to$ (*strictness*)

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

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

System $\mathscr{R}_0$ (Gardner 94-de Carvalho 07)

$$\text{Types:} \qquad \tau,\ \sigma \quad ::= \quad o \quad | \quad [\sigma_i]_{i\in I} \to \tau$$

- **intersection** = **multiset** of types $[\sigma_i]_{i\in I}$
- only on the left-h.s of $\to$ (*strictness*)

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

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

Remark

- **Relevant** system (no weakening, *cf.* `ax`-rule)

System $\mathscr{R}_0$ (Gardner 94-de Carvalho 07)

Types: $\tau,\ \sigma \quad ::= \quad o \quad | \quad [\sigma_i]_{i\in I} \to \tau$

- **intersection** = **multiset** of types $[\sigma_i]_{i\in I}$
- only on the left-h.s of $\to$ (*strictness*)

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

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

Remark

- **Relevant** system (no weakening, *cf.* ax-rule)
- **Non-idempotency ($\sigma \wedge \sigma \neq \sigma$):**
  in app-rule, pointwise multiset sum *e.g.*,

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

System $\mathscr{R}_0$ (Gardner 94-de Carvalho 07)

$$\text{Types:} \qquad \tau,\ \sigma \quad ::= \quad o \quad | \quad [\sigma_i]_{i \in I} \to \tau$$

- **intersection** = **multiset** of types $[\sigma_i]_{i \in I}$
- only on the left-h.s of $\to$ (*strictness*)

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

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

Example

$$\dfrac{\dfrac{}{f : [o] \to o}\ \texttt{ax} \qquad \dfrac{\dfrac{}{f : [o] \to o}\ \texttt{ax} \qquad \dfrac{}{x : o}\ \texttt{ax}}{f\, x : o}\ \texttt{app}}{f(f\, x) : o}\ \texttt{app}$$

System $\mathscr{R}_0$ (Gardner 94-de Carvalho 07)

Types: $\tau, \sigma \quad ::= \quad o \quad | \quad [\sigma_i]_{i\in I} \to \tau$

- **intersection** = **multiset** of types $[\sigma_i]_{i\in I}$
- only on the left-h.s of $\to$ (*strictness*)

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

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

Example

$$\dfrac{\dfrac{}{f : [o] \to o}\ \texttt{ax} \qquad \dfrac{\dfrac{}{f : [o] \to o}\ \texttt{ax} \qquad \dfrac{}{x : o}\ \texttt{ax}}{f\, x : o}\ \texttt{app}}{f : [[o] \to o, [o] \to o], x : [o] \vdash f(f\, x) : o}\ \texttt{app}$$

Types: $\tau,\ \sigma \quad ::= \quad o \quad | \quad [\sigma_i]_{i\in I} \to \tau$

- **intersection** = **multiset** of types $[\sigma_i]_{i\in I}$
- only on the left-h.s of $\to$ (*strictness*)

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

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

System $\mathscr{R}_0$ (Gardner 94-de Carvalho 07)

Types: $\tau,\ \sigma \quad ::= \quad o \quad | \quad [\sigma_i]_{i\in I} \to \tau$

- **intersection** = **multiset** of types $[\sigma_i]_{i\in I}$
- only on the left-h.s of $\to$ (*strictness*)

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

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

Head redexes always typed!

System $\mathscr{R}_0$ (Gardner 94-de Carvalho 07)

Types: $\tau, \sigma \quad ::= \quad o \quad | \quad [\sigma_i]_{i \in I} \to \tau$

- **intersection** = **multiset** of types $[\sigma_i]_{i \in I}$
- only on the left-h.s of $\to$ (*strictness*)

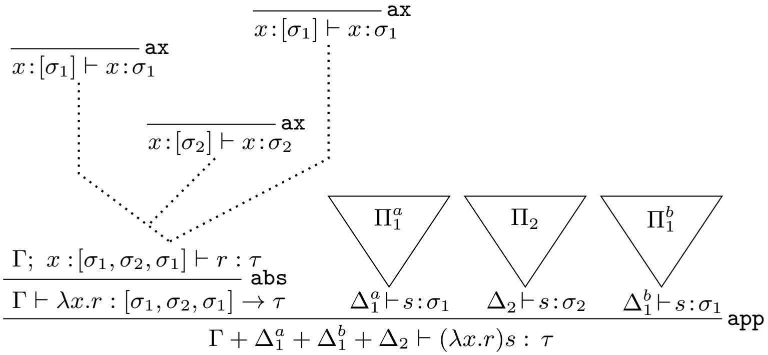
$$\frac{}{x : [\tau] \vdash x : \tau}\ \texttt{ax} \qquad \frac{\Gamma; x : [\sigma_i]_{i \in I} \vdash t : \tau}{\Gamma \vdash \lambda x.t : [\sigma_i]_{i \in I} \to \tau}\ \texttt{abs}$$

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

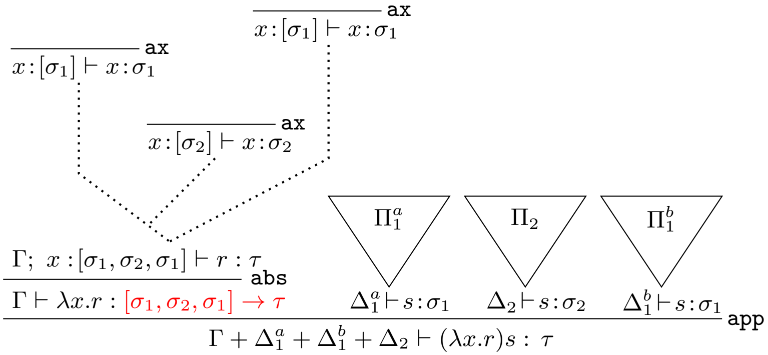
Head redexes always typed!

but an arg. may be typed 0 time

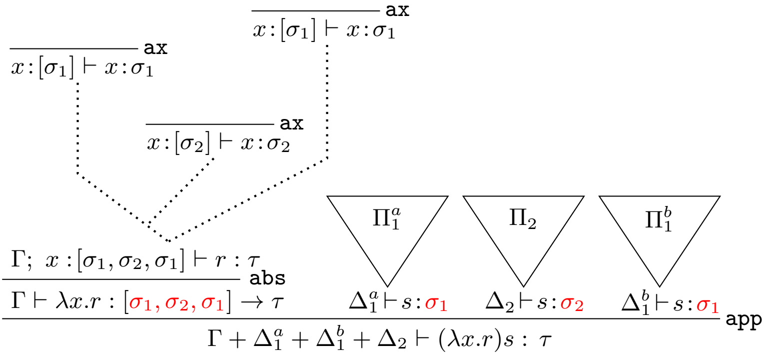
From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



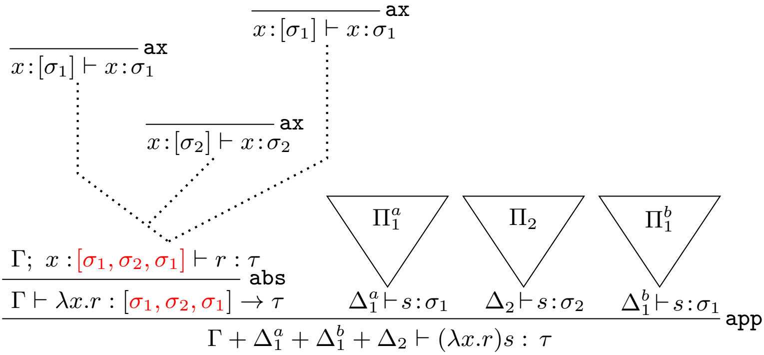
From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



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

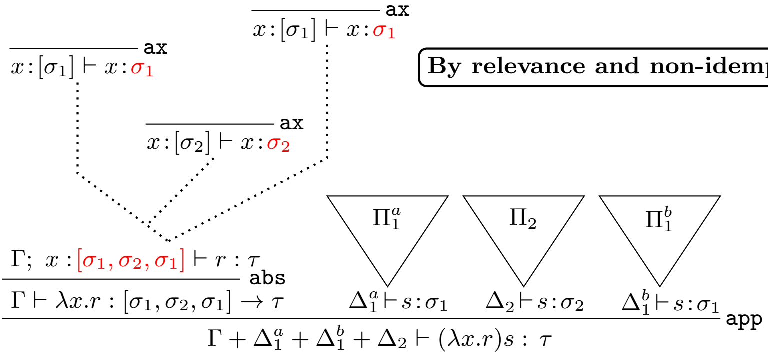


From a typing of $(\lambda x.r)s$ … to a typing of $r[s/x]$



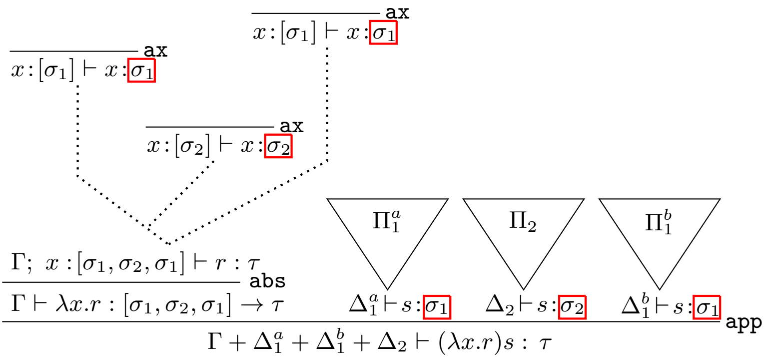
From a typing of $(\lambda x.r)s$ … to a typing of $r[s/x]$

By relevance and non-idempotency!

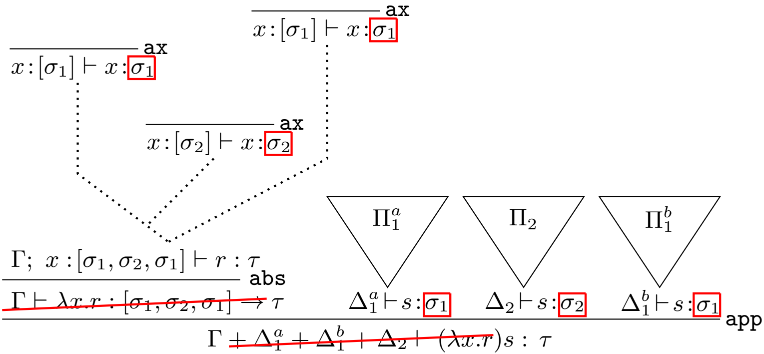


Subject reduction and expansion in $\mathscr{R}_0$

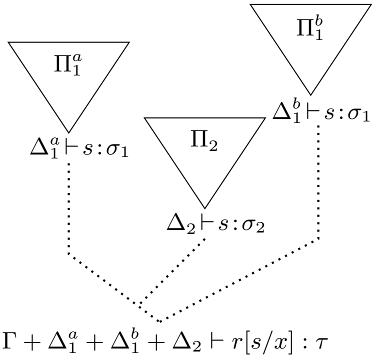
From a typing of $(\lambda x.r)s$ … to a typing of $r[s/x]$



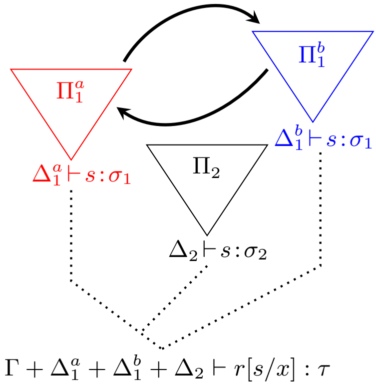
From a typing of $(\lambda x.r)s$ … to a typing of $r[s/x]$



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

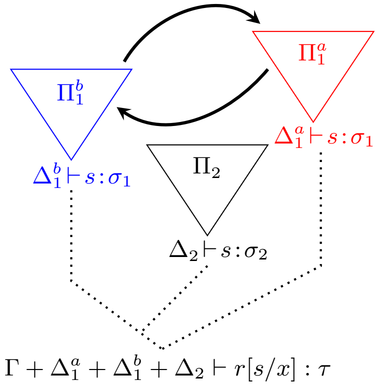


From a typing of $(\lambda x.r)s$ ... to a typing of $r[s/x]$



Non-determinism of SR

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



Non-determinism of SR

System $\mathtt{S}$ and sequence types [V., Lics'17]

System S

- coinductive type grammar
- replace $[\sigma_i]_{i\in I}$ with $(k\cdot\sigma_k)_{k\in K}$

sequence type (new intersection)

System S

- coinductive type grammar
- replace $[\sigma_i]_{i \in I}$ with $(k \cdot \sigma_k)_{k \in K}$

sequence type (new intersection)

$$(3 \cdot \sigma, 5 \cdot \tau, 9 \cdot \sigma) \ \textit{vs.} \ [\sigma, \tau, \sigma]$$

System S

- coinductive type grammar
- replace $[\sigma_i]_{i\in I}$ with $(k \cdot \sigma_k)_{k\in K}$

sequence type (new intersection)

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

System S

- coinductive type grammar
- replace $[\sigma_i]_{i\in I}$ with $(k\cdot\sigma_k)_{k\in K}$

↓ **sequence type** (new intersection)

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

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

System S

- coinductive type grammar
- replace $[\sigma_i]_{i \in I}$ with $(k \cdot \sigma_k)_{k \in K}$

sequence type (new intersection)

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

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

Why system S?

System S

- **coinductive** type grammar
- replace $[\sigma_i]_{i \in I}$ with $(k \cdot \sigma_k)_{k \in K}$

→ **sequence type** (new intersection)

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

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

Why system S?

- Coinduction necessary to *fully* type infinite NF

System S

- coinductive type grammar
- replace $[\sigma_i]_{i\in I}$ with $(k\cdot\sigma_k)_{k\in K}$

→ **sequence type** (new intersection)

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

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

Why system S?

- Coinduction necessary to *fully* type infinite NF
- Coinductive type grammar $\rightsquigarrow$ Ω is typable (unsoundness)

System S

- coinductive type grammar
- replace $[\sigma_i]_{i \in I}$ with $(k \cdot \sigma_k)_{k \in K}$

$\downarrow$ **sequence type** (new intersection)

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

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

Why system S?

- Coinduction necessary to *fully* type infinite NF
- Coinductive type grammar $\rightsquigarrow$ Ω is typable (unsoundness)
- Tracking necessary to recover soundness
  $\rightsquigarrow$ approximability (= validity criterion, next slide)

System S allows characterizing infinitary weak normalization

Approximability (intuitions)

- Order $\leqslant$ on the set derivations based on truncation

$$\dfrac{\overline{x : [[\ ] \to o] \vdash x : [\ ] \to o}}{x : [[o] \to o] \qquad \vdash x\, y : o}$$

Approximability (intuitions)

- Order $\leqslant$ on the set derivations based on truncation

$$\dfrac{\dfrac{}{x : [[o] \to o] \vdash x : [o] \to o} \qquad \dfrac{}{y : [o] \vdash y : o}}{x : [[o] \to o]; y : [o] \vdash x\, y : o}$$

Approximability (intuitions)

- Order $\leqslant$ on the set derivations based on truncation (black $\leqslant$ black+red)

$$\dfrac{\dfrac{}{x : [[o] \to o] \vdash x : [o] \to o} \qquad \dfrac{}{y : [o] \vdash y : o}}{x : [[o] \to o]; y : [o] \vdash x\, y : o}$$

Approximability (intuitions)

- Order $\leqslant$ on the set derivations based on truncation (black $\leqslant$ black+red)

$$\dfrac{\dfrac{}{x : [[o] \to o] \vdash x : [o] \to o} \qquad \dfrac{}{y : [o] \vdash y : o}}{x : [[o] \to o]; y : [o] \vdash x\, y : o}$$

$P_1 \leqslant P_2 \rightsquigarrow P_1$ **approximates** P_2

- Order $\leqslant$ on the set derivations based on truncation (black $\leqslant$ black+red)

$$\dfrac{\dfrac{}{x : [[o] \to o] \vdash x : [o] \to o} \qquad \dfrac{}{y : [o] \vdash y : o}}{x : [[o] \to o]; y : [o] \vdash x\, y : o}$$

$P_1 \leqslant P_2 \rightsquigarrow P_1$ **approximates** P_2

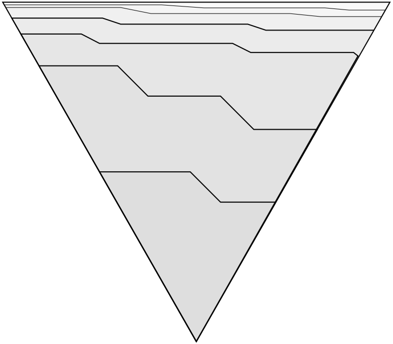
- **Approximability (def.):** a S-derivation P is approximable if it is the supremum of its finite approximations

- Order $\leqslant$ on the set derivations based on truncation (black $\leqslant$ black+red)

$$\dfrac{\overline{x : [[o] \to o] \vdash x : [o] \to o} \qquad \overline{y : [o] \vdash y : o}}{x : [[o] \to o]; y : [o] \vdash x\, y : o}$$

$P_1 \leqslant P_2 \rightsquigarrow P_1$ **approximates** P_2

- **Approximability (def.):** a S-derivation P is approximable if it is the supremum of its finite approximations

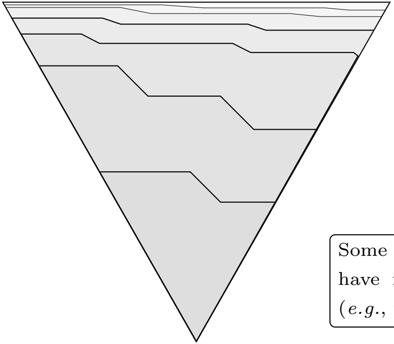


- Order $\leqslant$ on the set derivations based on truncation (black $\leqslant$ black+red)

$$\dfrac{\overline{x : [[o] \to o] \vdash x : [o] \to o} \qquad \overline{y : [o] \vdash y : o}}{x : [[o] \to o]; y : [o] \vdash x\, y : o}$$

$P_1 \leqslant P_2 \rightsquigarrow P_1$ **approximates** P_2

- **Approximability (def.):** a S-derivation P is approximable if it is the supremum of its finite approximations



Some derivations do not have finite approximations (*e.g.*, typings of Ω)

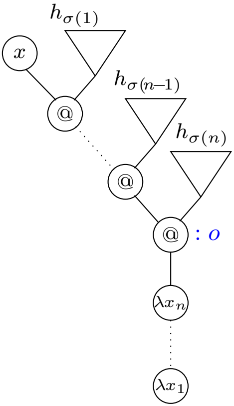
Properties of system S

Plan

Typing normal h.p. with permutator pairs

Goal:
Finding a set of pairs of types (S, T) s.t.
$x : S \vdash h : T$ when h is x-h.p.

Tracks are ignored above! (ok for NF)



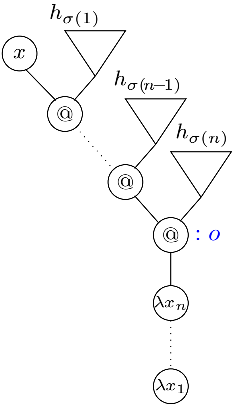
h

Goal:
Finding a set of pairs of types (S, T) s.t. $x : S \vdash h : T$ when h is x-h.p.

Tracks are ignored above! (ok for NF)

- $h = \lambda x_1 \dots x_n . x\, h_{\sigma(1)} \dots h_{\sigma(n)}$

  $h_1, \dots, h_n$ headed by $x_1, \dots, x_n$



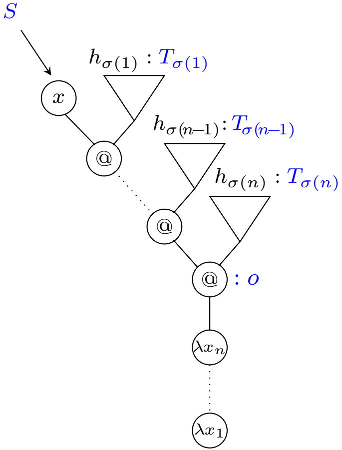
h

Goal:
Finding a set of pairs of types (S, T) s.t. $x : S \vdash h : T$ when h is x-h.p.

Tracks are ignored above! (ok for NF)

- $h = \lambda x_1 \dots x_n.x\, h_{\sigma(1)} \dots h_{\sigma(n)}$

  $h_1, \dots, h_n$ headed by $x_1, \dots, x_n$
- we want $x : S \vdash h : T$ and
  $x_1 : S_1 \vdash h_1 : T_1, \dots, x_n : S_n \vdash h_n\ T_n$



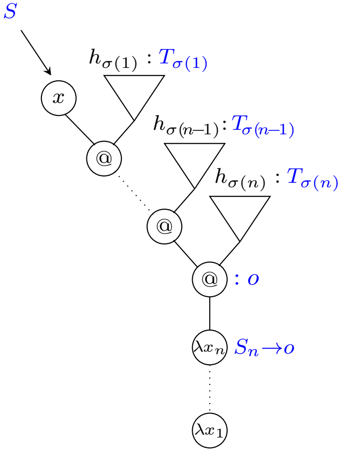
$h : T$

Goal:
Finding a set of pairs of types (S, T) s.t.
$x : S \vdash h : T$ when h is x-h.p.

Tracks are ignored above! (ok for NF)

- $h = \lambda x_1 \ldots x_n.x\, h_{\sigma(1)} \ldots h_{\sigma(n)}$

  $h_1, \ldots, h_n$ headed by $x_1, \ldots, x_n$
- we want $x : S \vdash h : T$ and
  $x_1 : S_1 \vdash h_1 : T_1, \ldots, x_n : S_n \vdash h_n\ T_n$



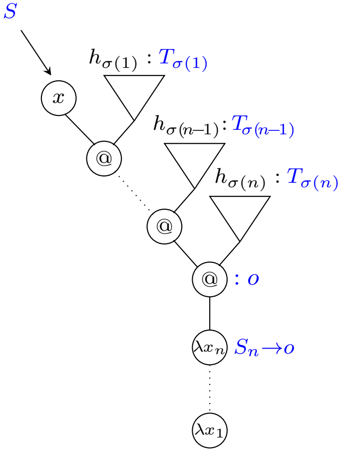
$h : T$

Goal:
Finding a set of pairs of types (S, T) s.t.
$x : S \vdash h : T$ when h is x-h.p.

Tracks are ignored above! (ok for NF)

- $h = \lambda x_1 \dots x_n . x\, h_{\sigma(1)} \dots h_{\sigma(n)}$

  $h_1, \dots, h_n$ headed by $x_1, \dots, x_n$
- we want $x : S \vdash h : T$ and
  $x_1 : S_1 \vdash h_1 : T_1, \dots, x_n : S_n \vdash h_n\ T_n$



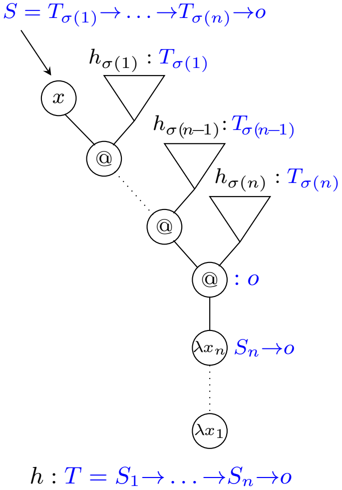
$h : T = S_1 \to \ldots \to S_n \to o$

Goal:
Finding a set of pairs of types (S, T) s.t. $x : S \vdash h : T$ when h is x-h.p.

Tracks are ignored above! (ok for NF)

- $h = \lambda x_1 \ldots x_n . x\, h_{\sigma(1)} \ldots h_{\sigma(n)}$

  $h_1, \ldots, h_n$ headed by $x_1, \ldots, x_n$
- we want $x : S \vdash h : T$ and
  $x_1 : S_1 \vdash h_1 : T_1, \ldots, x_n : S_n \vdash h_n\ T_n$

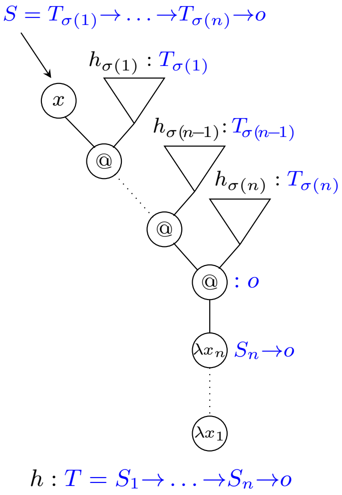


Goal:
Finding a set of pairs of types (S, T) s.t.
$x : S \vdash h : T$ when h is x-h.p.

Tracks are ignored above! (ok for NF)

- $h = \lambda x_1 \ldots x_n . x \, h_{\sigma(1)} \ldots h_{\sigma(n)}$

  $h_1, \ldots, h_n$ headed by $x_1, \ldots, x_n$
- we want $x : S \vdash h : T$ and
  $x_1 : S_1 \vdash h_1 : T_1, \ldots, x_n : S_n \vdash h_n \, T_n$



Goal:
Finding a set of pairs of types (S, T) s.t.
$x : S \vdash h : T$ when h is x-h.p.

Tracks are ignored above! (ok for NF)

- $h = \lambda x_1 \ldots x_n.x\, h_{\sigma(1)} \ldots h_{\sigma(n)}$

  $h_1, \ldots, h_n$ headed by $x_1, \ldots, x_n$
- we want $x : S \vdash h : T$ and
  $x_1 : S_1 \vdash h_1 : T_1, \ldots, x_n : S_n \vdash h_n\ T_n$

We obtain:
$S = T_{\sigma(1)} \to \ldots \to T_{\sigma(n)} \to o$
and $T = S_1 \to \ldots \to S_n \to o$

We obtain:
$S = T_{\sigma(1)} \rightarrow \ldots \rightarrow T_{\sigma(n)} \rightarrow o$
and $T = S_1 \rightarrow \ldots \rightarrow S_n \rightarrow o$

Permutator pairs

- When o ranges over $\mathscr{O}$ (the set of type atoms), the set $\texttt{PP}(o)$ of **o-permutator pairs** (S, T), where S and T are $\texttt{S}$-types, is defined by mutual coinduction:

$$\frac{(S_1, T_1) \in \texttt{PP}(o_1), \ldots, \ (S_n, T_n) \in \texttt{PP}(o_n) \qquad \sigma \in \mathfrak{S}_n}{((2 \cdot T_{\sigma(1)}) \to \ldots \to (2 \cdot T_{\sigma(n)}) \to o, (2 \cdot S_1) \to \ldots (2 \cdot S_n) \to o) \in \texttt{PP}(o)}$$

We obtain:
$S = T_{\sigma(1)} \to \ldots \to T_{\sigma(n)} \to o$
and $T = S_1 \to \ldots \to S_n \to o$

Permutator pairs

- When o ranges over $\mathscr{O}$ (the set of type atoms), the set $\texttt{PP}(o)$ of **o-permutator pairs** (S, T), where S and T are $\texttt{S}$-types, is defined by mutual coinduction:

$$\frac{(S_1, T_1) \in \texttt{PP}(o_1), \ldots, \ (S_n, T_n) \in \texttt{PP}(o_n) \qquad \sigma \in \mathfrak{S}_n}{((2 \cdot T_{\sigma(1)}) \to \ldots \to (2 \cdot T_{\sigma(n)}) \to o, (2 \cdot S_1) \to \ldots (2 \cdot S_n) \to o) \in \texttt{PP}(o)}$$

Permutator pairs

- When o ranges over $\mathscr{O}$ (the set of type atoms), the set $\mathtt{PP}(o)$ of **o-permutator pairs** (S, T), where S and T are $\mathtt{S}$-types, is defined by mutual coinduction:

$$\frac{(S_1, T_1) \in \mathtt{PP}(o_1), \ldots, \; (S_n, T_n) \in \mathtt{PP}(o_n) \qquad \sigma \in \mathfrak{S}_n}{((2 \cdot T_{\sigma(1)}) \to \ldots \to (2 \cdot T_{\sigma(n)}) \to o, (2 \cdot S_1) \to \ldots (2 \cdot S_n) \to o) \in \mathtt{PP}(o)}$$

- A pair $(S, T) \in \mathtt{PP}(o)$ is said to be **proper**, if, for all $o' \in \mathscr{O}$, o' occurs at most once in S and in T. The set of proper o-permutator pairs is denoted $\mathtt{PPP}(o)$.

Permutator pairs

- When o ranges over $\mathscr{O}$ (the set of type atoms), the set $\mathtt{PP}(o)$ of **o-permutator pairs** (S, T), where S and T are $\mathtt{S}$-types, is defined by mutual coinduction:

$$\frac{(S_1, T_1) \in \mathtt{PP}(o_1), \ldots, \ (S_n, T_n) \in \mathtt{PP}(o_n) \qquad \sigma \in \mathfrak{S}_n}{((2 \cdot T_{\sigma(1)}) \to \ldots \to (2 \cdot T_{\sigma(n)}) \to o, (2 \cdot S_1) \to \ldots (2 \cdot S_n) \to o) \in \mathtt{PP}(o)}$$

- A pair $(S, T) \in \mathtt{PP}(o)$ is said to be **proper**, if, for all $o' \in \mathscr{O}$, o' occurs at most once in S and in T. The set of proper o-permutator pairs is denoted $\mathtt{PPP}(o)$.

Characterization of h.p. in system S

- t is a x-h.p. iff $x : (2 \cdot S) \vdash t : T$ for some $(S, T) \in \mathtt{PPP}$
- t is a h.p. iff $\vdash t : (2 \cdot S) \to T$ for some $(S, T) \in \mathtt{PPP}$.

- For ∞-NF: $\Rightarrow$ (animation) $\Leftarrow$: use properness
- non-NF: use ∞-subj. reduction and expansion

Characterization in system S

- t is a x-h.p. iff $x : (2{\cdot}S) \vdash t : T$ (with (S, T) perm. pair)
- t is a h.p. iff $\vdash t : (2{\cdot}S) \rightarrow T$

Proof.

- NF case: previous slide
- Use infinitary subject reduction and expansion

Characterization in system S

- t is a x-h.p. iff $x : (2\cdot S) \vdash t : T$ (with (S,T) perm. pair)
- t is a h.p. iff $\vdash t : (2\cdot S) \to T$

Proof.

- NF case: previous slide
- Use infinitary subject reduction and expansion

Question: how to give a unique type to all h.p.?

Characterization in system $\mathtt{S}$

- t is a x-h.p. iff $x : (2 \cdot S) \vdash t : T$ (with (S, T) perm. pair)
- t is a h.p. iff $\vdash t : (2 \cdot S) \rightarrow T$

Proof.

- NF case: previous slide
- Use infinitary subject reduction and expansion

Question: how to give a unique type to all h.p.?

Idea: "collapse" the $(2 \cdot S) \rightarrow T$

Characterization in system S

- t is a x-h.p. iff $x : (2{\cdot}S) \vdash t : T$ (with (S, T) perm. pair)
- t is a h.p. iff $\vdash t : (2{\cdot}S) \to T$

Proof.

- NF case: previous slide
- Use infinitary subject reduction and expansion

Question: how to give a unique type to all h.p.?

Idea: "collapse" the $(2{\cdot}S) \to T$

Does this preserve type soundness/completeness?

Permutator schemes of degree d

Let $d \in \mathbb{N}$. A x-**permutator scheme** of degree d is a term t whose Böhm tree is equal to that of a x-h.p. for applicative depth $< d$.

Applicative depth: number of nestings inside arguments.

- Any term t is a 0-p.s.
- $h = \lambda x\, x_1 x_2.(x\,(\lambda x_{1,1} x_{1,2}.x_2\, t_1\, t_2))(\lambda x_{2,1}.x_1\, t_3)$ is a 2-p.s. ($t_{1,2,3}$:terms)

Permutator schemes of degree d

Let $d \in \mathbb{N}$. A x-**permutator scheme** of degree d is a term t whose Böhm tree is equal to that of a x-h.p. for applicative depth $< d$.

Applicative depth: number of nestings inside arguments.

- Any term t is a 0-p.s.
- $h = \lambda x\, x_1 x_2.(x\,(\lambda x_{1,1} x_{1,2}.x_2\, t_1\, t_2))(\lambda x_{2,1}.x_1\, t_3)$ is a 2-p.s. ($t_{1,2,3}$:terms)

Proper permutator pairs of degree d

$\mathtt{PPP}_d$: truncations $((S)^{\leqslant d}, (T)^{\leqslant d})$ of a proper permutator pair at domain depth $< d$ (finite types!)

Domain depth: number of nestings inside left-hand sides of arrows

Truncation

Permutator schemes of degree d

Let $d \in \mathbb{N}$. A x-**permutator scheme** of degree d is a term t whose Böhm tree is equal to that of a x-h.p. for applicative depth $< d$.

Applicative depth: number of nestings inside arguments.

- Any term t is a 0-p.s.
- $h = \lambda x\, x_1 x_2.(x\,(\lambda x_{1,1} x_{1,2}.x_2\, t_1\, t_2))(\lambda x_{2,1}.x_1\, t_3)$ is a 2-p.s. ($t_{1,2,3}$:terms)

Proper permutator pairs of degree d

$\mathtt{PPP}_d$: truncations $((S)^{\leqslant d}, (T)^{\leqslant d})$ of a proper permutator pair at domain depth $< d$ (finite types!)

Domain depth: number of nestings inside left-hand sides of arrows

Characterization in system $\mathtt{S}$

t is a d-p.s. headed by x iff $x : (2{\cdot}S) \vdash t : T$ (with $(S, T) \in \mathtt{PPP}_d$)

Truncation

Permutator schemes of degree d

Let $d \in \mathbb{N}$. A x-**permutator scheme** of degree d is a term t whose Böhm tree is equal to that of a x-h.p. for applicative depth $< d$.

Applicative depth: number of nestings inside arguments.

- Any term t is a 0-p.s.
- $h = \lambda x\, x_1 x_2.(x\,(\lambda x_{1,1} x_{1,2}.x_2\, t_1\, t_2))(\lambda x_{2,1}.x_1\, t_3)$ is a 2-p.s. ($t_{1,2,3}$:terms)

Proper permutator pairs of degree d

PPP_d: truncations $((S)^{\leqslant d}, (T)^{\leqslant d})$ of a proper permutator pair at domain depth $< d$ (finite types!)

Domain depth: number of nestings inside left-hand sides of arrows

Characterization in system $\mathtt{S}$

t is a d-p.s. headed by x iff $x : (2 \cdot S) \vdash t : T$ (with $(S, T) \in \mathsf{PPP}_d$)

Compatible truncation

$x : (2 \cdot S) \vdash t : T$ iff, for all $d \in \mathbb{N}$, $x : (2 \cdot (S)^{\leqslant d}) \vdash t : (T)^{\leqslant d}$

System $\mathtt{S_{hp}}$ = System $\mathtt{S}$ +

"Infinitary rule":

$$\frac{x : (2 \cdot S) \vdash t : T \qquad (S,T) \in \mathtt{PPP}}{\vdash \lambda x.t : \mathtt{ptyp}}\,\mathtt{hp}$$

"Finitary" rule (level d for all d):

$$\frac{x : (2 \cdot S) \vdash t : T \qquad (S,T) \in \mathtt{PPP}_d}{\vdash \lambda x.t : \mathtt{ptyp}_d}\,\mathtt{hp}_d$$

System $\mathtt{S_{hp}}$ = System $\mathtt{S}$ +

"Infinitary rule":

$$\frac{x : (2 \cdot S) \vdash t : T \qquad (S,T) \in \mathtt{PPP}}{\vdash \lambda x.t : \mathtt{ptyp}}\,\mathtt{hp}$$

"Finitary" rule (level d for all d):

$$\frac{x : (2 \cdot S) \vdash t : T \qquad (S,T) \in \mathtt{PPP}_d}{\vdash \lambda x.t : \mathtt{ptyp}_d}\,\mathtt{hp}_d$$

Approximability extended with the rule
$\mathtt{ptyp}_d \leqslant \mathtt{ptyp}$ for all d

System $\mathtt{S}_{\mathtt{hp}}$

System $\mathtt{S}_{\mathtt{hp}}$ = System $\mathtt{S}$ +

"Infinitary rule":

$$\frac{x : (2 \cdot S) \vdash t : T \qquad (S, T) \in \mathtt{PPP}}{\vdash \lambda x.t : \mathtt{ptyp}} \, \mathtt{hp}$$

"Finitary" rule (level d for all d):

$$\frac{x : (2 \cdot S) \vdash t : T \qquad (S, T) \in \mathtt{PPP}_d}{\vdash \lambda x.t : \mathtt{ptyp}_d} \, \mathtt{hp}_d$$

Approximability extended with the rule
$\mathtt{ptyp}_d \leqslant \mathtt{ptyp}$ for all d

Lemma (inversion for normal forms)

Let t be a (finite or not) normal form. Then $\vdash t : \mathtt{ptyp}$ iff t is a hereditary permutator.

System S_{hp} = System $\mathtt{S}$ +

"Infinitary rule":

$$\frac{x : (2 \cdot S) \vdash t : T \qquad (S, T) \in \mathtt{PPP}}{\vdash \lambda x.t : \mathtt{ptyp}}\ \mathtt{hp}$$

"Finitary" rule (level d for all d):

$$\frac{x : (2 \cdot S) \vdash t : T \qquad (S, T) \in \mathtt{PPP}_d}{\vdash \lambda x.t : \mathtt{ptyp}_d}\ \mathtt{hp}_d$$

Approximability extended with the rule
$\mathtt{ptyp}_d \leqslant \mathtt{ptyp}$ for all d

Lemma (inversion for normal forms)

Let t be a (finite or not) normal form. Then $\vdash t : \mathtt{ptyp}$ iff t is a hereditary permutator.

Lemma (compatible truncation in S_{hp})

Let t be a (finite or not) normal form. Then $\vdash t : \mathtt{ptyp}$ iff $t \vdash t : \mathtt{ptyp}_d$.

Finitary soundness

If P proves $C \vdash t : T$ in $\mathtt{S}_{\mathtt{hp}}$ and P is finite, then t is HN.

Finitary soundness

If P proves $C \vdash t : T$ in $\mathtt{S}_{\mathtt{hp}}$ and P is finite, then t is HN.

Infinitary subject reduction in $\mathtt{S}_{\mathtt{hp}}$

If $t \rightarrow^{\infty} t'$ and $C \vdash t : T$ then $C \vdash t' : T$.

$t \rightarrow^{\infty} t'$: ∞-productive reduction path, possibly computing the Böhm tree of t

Dynamic properties and final characterization

Finitary soundness

If P proves $C \vdash t : T$ in $\mathtt{S}_{\mathtt{hp}}$ and P is finite, then t is HN.

Infinitary subject reduction in $\mathtt{S}_{\mathtt{hp}}$

If $t \to^{\infty} t'$ and $C \vdash t : T$ then $C \vdash t' : T$.

$t \to^{\infty} t'$: ∞-productive reduction path, possibly computing the Böhm tree of t

Infinitary subject expansion in $\mathtt{S}_{\mathtt{hp}}$

If $t \to^{\infty} t'$ and $C \vdash t' : T$, then $C \vdash t : T$.

Approximability is crucial for ∞-subj. exp.!

Dynamic properties and final characterization

Finitary soundness

If P proves $C \vdash t : T$ in $\mathtt{S}_{\mathrm{hp}}$ and P is finite, then t is HN.

Infinitary subject reduction in $\mathtt{S}_{\mathrm{hp}}$

If $t \to^{\infty} t'$ and $C \vdash t : T$ then $C \vdash t' : T$.

$t \to^{\infty} t'$: ∞-productive reduction path, possibly computing the Böhm tree of t

Infinitary subject expansion in $\mathtt{S}_{\mathrm{hp}}$

If $t \to^{\infty} t'$ and $C \vdash t' : T$, then $C \vdash t : T$.

Approximability is crucial for ∞-subj. exp.!

From ∞-s.r. and s.e. + inversion for NF + truncation:

A unique type for hereditary permutators

For all terms t, t is a h.p. iff $\vdash t : \mathtt{ptyp}$ in system $\mathtt{S}_{\mathrm{hp}}$.

Summary and future work

Contribution: characterizing hereditary permutators with a unique type

- Not possible in the inductive case (not r.e.)
- System $\mathtt{S}$: coinductive variant of **non-idempotent intersection** types:
  - elements of multisets annotated with **tracks**
  - allows recovering soundness w. **approximability** (=*validity criterion*).
- In system $\mathtt{S}$, h.p. characterized with **proper permutator pairs**.

- System $\mathtt{S}_{\mathtt{hp}}$: **"collapsing"** proper permutator pairs using constant $\mathtt{ptyp}$.
  $\mathtt{ptyp}$ approximated by $\mathtt{ptyp}_d \rightsquigarrow$ approximability extends to $\mathtt{S}_{\mathtt{hp}}$.
- Truncation lemma for t NF:

$$\vdash t : \mathtt{ptyp} \text{ approximated by } \vdash t : \mathtt{ptyp}_d \text{ (for all } d \in \mathbb{N})$$

- $\mathtt{S}_{\mathtt{hp}}$: infinitary subject reduction and expansion
  $\rightsquigarrow$ retrieving methods of finitary intersection type systems
- Soundness and completeness w.r.t. h.p.: t is a h.p. iff $\vdash t : \mathtt{ptyp}$ in $\mathtt{S}_{\mathtt{hp}}$

Future work: Other sets of terms with infinitary behaviors

inc. normalization in other ∞-calculi