The Expressive Power of Coinductive Rigid Types with non-Idempotent Intersection

Pierre VIAL
IRIF, Paris 7

HOR 2016

June 25, 2016

PLAN

INTERSECTION TYPES

- Simple type systems (STS): Typable $\Rightarrow$ Normalizable.

INTERSECTION TYPES

- Simple type systems (STS): Typable $\Rightarrow$ Normalizable.
- Intersection type systems (ITS): Typable $\Leftrightarrow$ Normalizable.

INTERSECTION TYPES

- Simple type systems (STS): Typable $\Rightarrow$ Normalizable.
- Intersection type systems (ITS): Typable $\Leftrightarrow$ Normalizable.
- STS: a variable x can be assigned only one type (that can be used several times).

INTERSECTION TYPES

- Simple type systems (STS): Typable $\Rightarrow$ Normalizable.
- Intersection type systems (ITS): Typable $\Leftrightarrow$ Normalizable.
- STS: a variable x can be assigned only one type (that can be used several times).
- ITS: a variable can be typed several times, with different types. $x : A \wedge B \wedge B \wedge C$.

INTERSECTION TYPES

- Simple type systems (STS): Typable $\Rightarrow$ Normalizable.
- Intersection type systems (ITS): Typable $\Leftrightarrow$ Normalizable.
- STS: a variable x can be assigned only one type (that can be used several times).
- ITS: a variable can be typed several times, with different types. $x : A \wedge B \wedge B \wedge C$.
- *Example:* usually, xx cannot be typed in STS, but xx can be typed in ITS: if x is assigned $A \wedge (A \to B)$, then $xx : B$ is derivable.

WHAT KIND OF INTERSECTION?

Intersection $\wedge$.

Associativity assumed. Commutativity ($A \wedge B = B \wedge A$) ? Idempotency ($A \wedge A = A$) ?

WHAT KIND OF INTERSECTION?

Intersection $\wedge$.

Associativity assumed. Commutativity ($A \wedge B = B \wedge A$) ? Idempotency ($A \wedge A = A$) ?

- Idempotent, commutative: $A \wedge B \wedge A = A \wedge A \wedge B = A \wedge B$.
  Paradigm: sets, $\{A, B, A\} = \{A, A, B\} = \{A, B\}$

WHAT KIND OF INTERSECTION?

Intersection $\wedge$.

Associativity assumed. Commutativity ($A \wedge B = B \wedge A$) ? Idempotency ($A \wedge A = A$) ?

- Idempotent, commutative: $A \wedge B \wedge A = A \wedge A \wedge B = A \wedge B$.
  Paradigm: sets, $\{A, B, A\} = \{A, A, B\} = \{A, B\}$
- Non-Idempotent, commutative: $A \wedge B \wedge A = A \wedge A \wedge B \neq A \wedge B$.
  Paradigm: multisets, $[A, B, A] = [A, A, B] \neq [A, B]$.

WHAT KIND OF INTERSECTION?

Intersection $\wedge$.

Associativity assumed. Commutativity ($A \wedge B = B \wedge A$) ? Idempotency ($A \wedge A = A$) ?

- Idempotent, commutative: $A \wedge B \wedge A = A \wedge A \wedge B = A \wedge B$.
  Paradigm: sets, $\{A, B, A\} = \{A, A, B\} = \{A, B\}$
- Non-Idempotent, commutative: $A \wedge B \wedge A = A \wedge A \wedge B \neq A \wedge B$.
  Paradigm: multisets, $[A, B, A] = [A, A, B] \neq [A, B]$.
- Non-Idempotent, non-commutative: $A \wedge B \wedge A \neq A \wedge A \wedge B$.
  Paradigm: lists, $(A, B, A) \neq (A, A, B) \neq (A, B)$ (this does not work).

What Kind of Intersection?

Intersection $\wedge$.

Associativity assumed. Commutativity ($A \wedge B = B \wedge A$) ? Idempotency ($A \wedge A = A$) ?

- Idempotent, commutative: $A \wedge B \wedge A = A \wedge A \wedge B = A \wedge B$.
  Paradigm: sets, $\{A, B, A\} = \{A, A, B\} = \{A, B\}$
- Non-Idempotent, commutative: $A \wedge B \wedge A = A \wedge A \wedge B \neq A \wedge B$.
  Paradigm: multisets, $[A, B, A] = [A, A, B] \neq [A, B]$.
- Non-Idempotent, non-commutative: $A \wedge B \wedge A \neq A \wedge A \wedge B$.
  Paradigm: lists, $(A, B, A) \neq (A, A, B) \neq (A, B)$ (this does not work).
  In-between possibility: **rigidity** (paradigm: sequences)

CONTENTS OF THIS TALK

- We consider here two coinductive ITS, namely $\mathscr{M}$ and $\mathtt{S}$
  - In $\mathscr{M}$ (adapted from Gardner[94]/de Carvalho[07]), intersection is represented by means of multisets.
  - In $\mathtt{S}$, intersection is represented by means of sequences.

CONTENTS OF THIS TALK

- We consider here two coinductive ITS, namely $\mathscr{M}$ and $\mathtt{S}$
  - In $\mathscr{M}$ (adapted from Gardner[94]/de Carvalho[07]), intersection is represented by means of multisets.
  - In $\mathtt{S}$, intersection is represented by means of sequences.

- Forget about the order (inside a sequence):
  - A $\mathtt{S}$-type T **collapses** into a $\mathscr{M}$-type $\overline{T} = \tau$
  - A $\mathtt{S}$-context C collapses into a $\mathscr{M}$-context $\overline{C} = \Gamma$.
  - Likewise, a $\mathtt{S}$-judgment collapses into a $\mathscr{M}$-judgment.
  - Collapsing the judgments, a $\mathtt{S}$-derivation P collapses into a $\mathscr{M}$-derivation $\overline{P} = \Pi$.

CONTENTS OF THIS TALK

- We consider here two coinductive ITS, namely $\mathscr{M}$ and $\mathtt{S}$
  - In $\mathscr{M}$ (adapted from Gardner[94]/de Carvalho[07]), intersection is represented by means of multisets.
  - In $\mathtt{S}$, intersection is represented by means of sequences.
- Forget about the order (inside a sequence):
  - A $\mathtt{S}$-type T **collapses** into a $\mathscr{M}$-type $\overline{T} = \tau$
  - A $\mathtt{S}$-context C collapses into a $\mathscr{M}$-context $\overline{C} = \Gamma$.
  - Likewise, a $\mathtt{S}$-judgment collapses into a $\mathscr{M}$-judgment.
  - Collapsing the judgments, a $\mathtt{S}$-derivation P collapses into a $\mathscr{M}$-derivation $\overline{P} = \Pi$.
- **Question 1 (full collapse?):** for all $\mathscr{M}$-derivation Π, is there a $\mathtt{S}$-derivation P that collapses into Π ? (easy for types and contexts)

PLAN

Sequences

- If X is a set, let $\mathtt{S}(\mathtt{X})$ be the set of families of elements of X indexed by integers $\geqslant 2$.

Sequences

- If X is a set, let $\mathtt{S(X)}$ be the set of families of elements of X indexed by integers $\geqslant 2$.
- Let $\vec{x} = (x_k)_{k \in K} \in \mathtt{S(X)}$ and $k \in K$.
  Integers are seen as **tracks** and we say that x_k is placed on **track** k inside $\vec{x}$.

SEQUENCES

- If X is a set, let $\mathtt{S(X)}$ be the set of families of elements of X indexed by integers $\geqslant 2$.

- Let $\vec{x} = (x_k)_{k \in K} \in \mathtt{S(X)}$ and $k \in K$.
  Integers are seen as **tracks** and we say that x_k is placed on **track** k inside $\vec{x}$.

- The sequence $(x_k)_{k \in \{3,\ 5, 9\}}$ with $x_3 = x,\ x_5 = y,\ x_9 = z$ will be written:
  $$(3 \cdot x, 5 \cdot y, 9 \cdot z)$$

SEQUENCES

- If X is a set, let $\mathtt{S(X)}$ be the set of families of elements of X indexed by integers $\geqslant 2$.
- Let $\vec{x} = (x_k)_{k \in K} \in \mathtt{S(X)}$ and $k \in K$.
  Integers are seen as **tracks** and we say that x_k is placed on **track** k inside $\vec{x}$.
- The sequence $(x_k)_{k \in \{3,\ 5,9\}}$ with $x_3 = x,\ x_5 = y,\ x_9 = z$ will be written:
  $$(3 \cdot x, 5 \cdot y, 9 \cdot z)$$
- We cannot always perform the union of sequences.

COLLAPSING SEQUENCES INTO MULTISETS

- $\mathscr{M}(X)$ be the set of multisets of elements of X.

Collapsing Sequences into Multisets

- $\mathscr{M}(X)$ be the set of multisets of elements of X.
- **Observation:** $\mathscr{M}(X)$ is the quotient set $\mathtt{S(X)}/\overset{1}{\equiv}$ where $(x_k)_{k\in K} \overset{1}{\equiv} (x'_k)_{k\in K'}$ if there is a bijection $\rho :\ K \to K'$ a bijection such that: $\forall k \in K,\ x_k = x'_{\rho(k)}$.

COLLAPSING SEQUENCES INTO MULTISETS

- $\mathscr{M}(X)$ be the set of multisets of elements of X.
- **Observation:** $\mathscr{M}(X)$ is the quotient set $\mathtt{S(X)}/\overset{1}{\equiv}$ where $(x_k)_{k\in K} \overset{1}{\equiv} (x'_k)_{k\in K'}$ if there is a bijection $\rho : K \to K'$ a bijection such that: $\forall k \in K,\ x_k = x'_{\rho(k)}$.
- *Example:* $[x, y, x]$ is the **collapse** of $(2 \cdot x, 3 \cdot y, 5 \cdot x)$.

COLLAPSING SEQUENCES INTO MULTISETS

- $\mathscr{M}(X)$ be the set of multisets of elements of X.
- **Observation:** $\mathscr{M}(X)$ is the quotient set $\mathtt{S(X)}/\overset{1}{\equiv}$ where $(x_k)_{k\in K} \overset{1}{\equiv} (x'_k)_{k\in K'}$ if there is a bijection $\rho : K \to K'$ a bijection such that: $\forall k \in K,\ x_k = x'_{\rho(k)}$.
- *Example:* $[x, y, x]$ is the **collapse** of $(2 \cdot x, 3 \cdot y, 5 \cdot x)$.
- Equalities: $[x, y, x] = [x, x, y]$ but $(2 \cdot x, 3 \cdot y, 5 \cdot x) \neq (2 \cdot x, 3 \cdot x, 5 \cdot y)$ Equality is said to be **tight** for sequences (synctactic equality) and **loose** for multisets.

PLAN

STRICT INTERSECTION TYPES

- Let $\mathscr{X}$ be a countable set of type variables (metavariable α).
- Simple Type System.
  - In a context $\Gamma, x : \sigma$. A judgment is of the form $\Gamma \vdash t : \tau$.
  - $\tau, \sigma \ ::= \ \alpha \mid \sigma \to \tau$

STRICT INTERSECTION TYPES

- Let $\mathscr{X}$ be a countable set of type variables (metavariable α).
- Simple Type System.
  - In a context $\Gamma, x : \sigma$. A judgment is of the form $\Gamma \vdash t : \tau$.
  - $\tau, \sigma \ ::= \ \alpha \mid \sigma \to \tau$
- Intersection Type System:
  - In a context $\Gamma, x : \bigwedge_{i \in I} \sigma_i$. A judgment is of the form $\Gamma \vdash t : \tau$.
  - $\tau, \sigma_i \ ::= \ \alpha \mid (\bigwedge_{i \in I} \sigma_i) \to \tau$

Strict Intersection Types

- Let $\mathscr{X}$ be a countable set of type variables (metavariable α).
- Simple Type System.
  - In a context $\Gamma, x : \sigma$. A judgment is of the form $\Gamma \vdash t : \tau$.
  - $\tau, \sigma \ ::= \ \alpha \mid \sigma \to \tau$
- Intersection Type System:
  - In a context $\Gamma, x : \bigwedge_{i \in I} \sigma_i$. A judgment is of the form $\Gamma \vdash t : \tau$.
  - $\tau, \sigma_i \ ::= \ \alpha \mid (\bigwedge_{i \in I} \sigma_i) \to \tau$
- The application typing rule generally relies upon equality of intersection types (see next slide).

TYPING APPLICATION

- Symple types:

$$\frac{\Gamma \vdash t : \sigma \to \tau \qquad \Delta \vdash u : \sigma'}{\Gamma, \Delta \vdash t\, u : \tau}\ \mathtt{app}$$

Constraint: $\sigma = \sigma'$ (Γ and Δ do not type the same variables)

Typing Application

- Intersection types:

$$\dfrac{\Gamma \vdash t : \bigwedge_{i\in I} \sigma_i \to \tau \qquad (\Delta_i \vdash u : \sigma'_i)^{i \in I'}}{\Gamma \cup \bigcup_{i\in I} \Delta_i \vdash t\,u : \tau}\ \texttt{app}$$

TYPING APPLICATION

- Intersection types:

$$\frac{\Gamma \vdash t : \bigwedge_{i \in I} \sigma_i \to \tau \qquad (\Delta_i \vdash u : \sigma'_i)^{i \in I'}}{\Gamma \cup \bigcup_{i \in I} \Delta_i \vdash t\, u : \tau}\ \texttt{app}$$

Constraint: $\bigwedge_{i \in I} \sigma_i = \bigwedge_{i \in I'} \sigma'_i$

Typing Application

- Intersection types:

$$\dfrac{\Gamma \vdash t : \bigwedge_{i\in I} \sigma_i \to \tau \qquad (\Delta_i \vdash u : \sigma'_i)^{i\in I'}}{\Gamma \cup \bigcup_{i\in I} \Delta_i \vdash t\,u : \tau}\ \texttt{app}$$

Constraint: $\bigwedge_{i\in I} \sigma_i = \bigwedge_{i\in I'} \sigma'_i$ **Motto**: $\mathscr{L} = \mathscr{R}$

TYPING APPLICATION

- Intersection types:

$$\frac{\Gamma \vdash t : \bigwedge_{i\in I} \sigma_i \to \tau \qquad (\Delta_i \vdash u : \sigma'_i)^{i\in I'}}{\Gamma \cup \bigcup_{i\in I} \Delta_i \vdash t\,u : \tau}\ \texttt{app}$$

Constraint: $\bigwedge_{i\in I} \sigma_i = \bigwedge_{i\in I'} \sigma'_i$ **Motto**: $\mathscr{L} = \mathscr{R}$

- Idem Commutative ITS: i=set, $\wedge = \cup$: $\mathscr{L} = \mathscr{R}$ is $\{\sigma_i\}_{i\in I} = \{\sigma'_i\}_{i\in I'}$

TYPING APPLICATION

- Intersection types:

$$\frac{\Gamma \vdash t : \bigwedge_{i \in I} \sigma_i \to \tau \qquad (\Delta_i \vdash u : \sigma'_i)^{i \in I'}}{\Gamma \cup \bigcup_{i \in I} \Delta_i \vdash t\, u : \tau}\ \texttt{app}$$

Constraint: $\bigwedge_{i \in I} \sigma_i = \bigwedge_{i \in I'} \sigma'_i$ **Motto**: $\mathscr{L} = \mathscr{R}$

- Idem Commutative ITS: i=set, $\wedge = \cup$: $\mathscr{L} = \mathscr{R}$ is $\{\sigma_i\}_{i \in I} = \{\sigma'_i\}_{i \in I'}$
- Non-Idem Commutative ITS: i=multiset, $\wedge = +$: $\mathscr{L} = \mathscr{R}$ is $[\sigma_i]_{i \in I} = [\sigma'_i]_{i \in I'}$

TYPING APPLICATION

- Intersection types:

$$\frac{\Gamma \vdash t : \bigwedge_{i\in I} \sigma_i \to \tau \qquad (\Delta_i \vdash u : \sigma'_i)^{i\in I'}}{\Gamma \cup \bigcup_{i\in I} \Delta_i \vdash t\,u : \tau}\ \texttt{app}$$

Constraint: $\bigwedge_{i\in I} \sigma_i = \bigwedge_{i\in I'} \sigma'_i$ **Motto**: $\mathscr{L} = \mathscr{R}$

- Idem Commutative ITS: i=set, $\wedge = \cup$: $\mathscr{L} = \mathscr{R}$ is $\{\sigma_i\}_{i\in I} = \{\sigma'_i\}_{i\in I'}$
- Non-Idem Commutative ITS: i=multiset, $\wedge = +$: $\mathscr{L} = \mathscr{R}$ is $[\sigma_i]_{i\in I} = [\sigma'_i]_{i\in I'}$
- Rigid ITS: i=sequence, $\wedge = \cup$ (disjoint): L = R is $(S_k)_{k\in K} = (S'_k)_{k\in K'}$
  ($C(x), D_k(x)$ disjoint for all x)

TYPING RULES OF $\mathscr{M}_0$ (GARDNER / DE CARVALHO)

Contexts (Γ, Δ): collection of $x : [\tau_i]_{i \in I}$.

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

$$\frac{\Gamma \vdash t : [\sigma_i]_{i\in I} \to \tau \qquad (\Delta_i \vdash u : \sigma'_i)^{i\in I'}}{\Gamma + \sum_{i\in I} \Delta_i \vdash t(u) : \tau}\ \mathrm{app}$$

Constraint in `app`: multiset equality $[\sigma_i]_{i\in I} = [\sigma'_i]_{i\in I'}$ must hold.

Typing Rules of $\mathscr{M}_0$ (Gardner/de Carvalho)

Contexts (Γ, Δ): collection of $x : [\tau_i]_{i \in I}$.

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

$$\frac{\Gamma \vdash t : [\sigma_i]_{i\in I} \to \tau \qquad (\Delta_i \vdash u : \sigma'_i)^{i \in I'}}{\Gamma + \sum_{i\in I} \Delta_i \vdash t(u) : \tau}\ \text{app}$$

Constraint in `app`: multiset equality $[\sigma_i]_{i\in I} = [\sigma'_i]_{i\in I'}$ must hold.

Remark

- Multiset sum: $[\alpha, \beta, \alpha] + [\alpha, \beta, \gamma, \gamma] = [\alpha, \alpha, \alpha, \beta, \beta, \gamma, \gamma]$
- No implicit contraction: accumulation of typing information .

EXAMPLE

Typing $\Delta = \lambda x.xx$ (with application arity = 3):

$$\dfrac{\dfrac{\dfrac{}{x : [[\alpha, \beta, \alpha] \to \alpha] \vdash x : [\alpha, \beta, \alpha] \to \alpha}\ \text{ax} \quad \dfrac{}{x : [\alpha] \vdash x : \alpha}\ \text{ax} \quad \dfrac{}{x : [\beta] \vdash x : \beta}\ \text{ax} \quad \dfrac{}{x : [\alpha] \vdash x : \alpha}\ \text{ax}}{x : [\alpha, \beta, \alpha, [\alpha, \beta, \alpha] \to \alpha] \vdash xx : \alpha}\ \text{ap}}{\vdash \lambda x.xx : [\alpha, \beta, \alpha, [\alpha, \beta, \alpha] \to \alpha] \to \alpha}\ \text{abs}$$

SUBJECT REDUCTION PROPERTY FOR $\mathscr{M}_0$

If $\Pi \triangleright \Gamma \vdash t : \tau$ and $t \to t'$, then $\exists \Pi' \triangleright \Gamma \vdash t' : \tau$

Subject Reduction Property for $\mathscr{M}_0$

If $\Pi \rhd \Gamma \vdash t : \tau$ and $t \to t'$, then $\exists \Pi' \rhd \Gamma \vdash t' : \tau$

$$(\lambda x.r)s \to r[s/x]$$

$$\dfrac{\dfrac{\begin{array}{c}\Pi_r\\ \vdots\\ \Gamma, x : [\sigma_i]_{i\in I} \vdash r \quad : \tau\end{array}}{\Gamma \vdash \lambda x.r : [\sigma_i]_{i\in I} \to \tau}\,\texttt{abs} \qquad \left(\begin{array}{c}\Pi_i\\ \vdots\\ \Delta_i \vdash s : \sigma_i\end{array}\right)^{i\in I}}{\Gamma + \sum_{i\in I}\Delta_i \vdash (\lambda x.r)s : \tau}\,\texttt{app}$$

SUBJECT REDUCTION PROPERTY FOR $\mathscr{M}_0$

If $\Pi \rhd \Gamma \vdash t : \tau$ and $t \to t'$, then $\exists \Pi' \rhd \Gamma \vdash t' : \tau$

$$(\lambda x.r)s \to r[s/x]$$

Axiom leaves
typing x inside Π_r

$$\begin{array}{c}\Pi_r \\ \vdots \quad \left(\dfrac{}{x : [\sigma_i] \vdash x : \sigma_i}\,\texttt{ax}\right)_{i\in I} \\ \vdots\end{array}$$

$$\dfrac{\dfrac{\Gamma, x : [\sigma_i]_{i\in I} \vdash r : \tau}{\Gamma \vdash \lambda x.r : [\sigma_i]_{i\in I} \to \tau}\,\texttt{abs} \qquad \left(\begin{array}{c}\Pi_i \\ \vdots \\ \Delta_i \vdash s : \sigma_i\end{array}\right)^{i \in I}}{\Gamma + \sum_{i\in I} \Delta_i \vdash (\lambda x.r)s : \tau}\,\texttt{app}$$

Subject Reduction Property for $\mathscr{M}_0$

If $\Pi \triangleright \Gamma \vdash t : \tau$ and $t \to t'$, then $\exists \Pi' \triangleright \Gamma \vdash t' : \tau$

$$(\lambda x.r)s \to r[s/x]$$

$$\dfrac{\dfrac{\begin{array}{c}\Pi_r \\ \left(\dfrac{}{x : [\sigma_i] \vdash x : \boxed{\sigma_i}}\,\mathtt{ax} \right)_{i \in I} \\ \vdots \\ \Gamma, x : [\sigma_i]_{i \in I} \vdash r \quad : \tau\end{array}}{\Gamma \vdash \lambda x.r : [\sigma_i]_{i \in I} \to \tau}\,\mathtt{abs} \qquad \left(\begin{array}{c}\Pi_i \\ \vdots \\ \Delta_i \vdash s : \boxed{\sigma_i}\end{array} \right)^{i \in I}}{\Gamma + \sum_{i \in I} \Delta_i \vdash (\lambda x.r)s : \tau}\,\mathtt{app}$$

Subject Reduction Property for $\mathscr{M}_0$

If $\Pi \triangleright \Gamma \vdash t : \tau$ and $t \to t'$, then $\exists \Pi' \triangleright \Gamma \vdash t' : \tau$

$$(\lambda x.r)s \to r[s/x]$$

“association”

$$\dfrac{\dfrac{\begin{array}{c}\Pi_r \\ \left(\dfrac{}{x : [\sigma_i] \vdash x : \boxed{\sigma_i}}\,\texttt{ax}\right)_{i\in I} \\ \vdots \\ \Gamma, x : [\sigma_i]_{i\in I} \vdash r : \tau\end{array}}{\Gamma \vdash \lambda x.r : [\sigma_i]_{i\in I} \to \tau}\,\texttt{abs} \qquad \left(\begin{array}{c}\Pi_i \\ \vdots \\ \Delta_i \vdash s : \boxed{\sigma_i}\end{array}\right)^{i\in I}}{\Gamma + \sum_{i\in I} \Delta_i \vdash (\lambda x.r)s : \tau}\,\texttt{app}$$

Subject Reduction Property for $\mathscr{M}_0$

If $\Pi \rhd \Gamma \vdash t : \tau$ and $t \to t'$, then $\exists \Pi' \rhd \Gamma \vdash t' : \tau$

$$(\lambda x.r)s \to r[s/x]$$

$$\dfrac{\dfrac{\begin{array}{c}\Pi_r \\ \left(\dfrac{}{\cancel{x : [\sigma_i] \vdash x :}\ \boxed{\sigma_i}}\,\texttt{ax}\right)_{i \in I} \\ \vdots \\ \Gamma, x : [\sigma_i]_{i \in I} \vdash r : \tau\end{array}}{\cancel{\Gamma \vdash \lambda x.r : [\sigma_i]_{i \in I} \to \tau}}\,\texttt{abs} \qquad \left(\begin{array}{c}\Pi_i \\ \vdots \\ \Delta_i \vdash s : \boxed{\sigma_i}\end{array}\right)^{i \in I}}{\cancel{\Gamma + \sum_{i \in I} \Delta_i \vdash (\lambda x.r)s : \tau}}\,\texttt{app}$$

“association”

SUBJECT REDUCTION PROPERTY FOR $\mathscr{M}_0$

If $\Pi \triangleright \Gamma \vdash t : \tau$ and $t \to t'$, then $\exists \Pi' \triangleright \Gamma \vdash t' : \tau$

$$(\lambda x.r)s \to r[s/x]$$

$$\begin{array}{c} \Pi_r \\ \vdots \\ \Gamma + \sum_{i \in I} \Delta_i \vdash r\,[s/x] : \tau \end{array} \quad \left(\begin{array}{c} \Pi_i \\ \vdots \\ \Delta_i \vdash s : \sigma_i \end{array} \right)^{i \in I}$$

SUBJECT REDUCTION PROPERTY FOR $\mathscr{M}_0$

If $\Pi \rhd \Gamma \vdash t : \tau$ and $t \to t'$, then $\exists \Pi' \rhd \Gamma \vdash t' : \tau$

$$(\lambda x.r)s \to r[s/x]$$

$$\begin{array}{c} \Pi_r \quad \left(\begin{array}{c} \Pi_i \\ \vdots \\ \Delta_i \vdash s : \sigma_i \end{array}\right)^{i \in I} \\ \vdots \\ \Gamma + \sum_{i\in I} \Delta_i \vdash r\,[s/x] : \tau \end{array}$$

Vocabulary:
We say each **association** (between x-axiom leaves and arg-derivations) or **reduction choice**, yields a **reduced derivation** Π' typing $r[s/x]$.

SUBJECT REDUCTION PROPERTY FOR $\mathscr{M}_0$

If $\Pi \triangleright \Gamma \vdash t : \tau$ and $t \to t'$, then $\exists \Pi' \triangleright \Gamma \vdash t' : \tau$

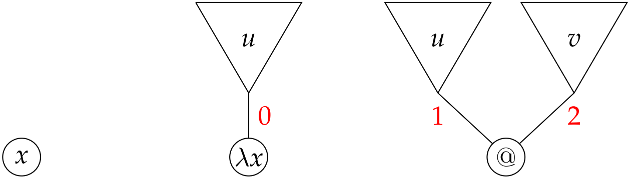
$$(\lambda x.r)s \to r[s/x]$$

$$\begin{array}{c} \Pi_r \quad \left(\begin{array}{c} \Pi_i \\ \vdots \\ \Delta_i \vdash s : \sigma_i \end{array} \right)^{i \in I} \\ \vdots \\ \Gamma + \sum_{i \in I} \Delta_i \vdash r\,[s/x] : \tau \end{array}$$

Observation:
If a type σ occurs several times in $[\sigma_i]_{i \in I}$, there can be several associations, each one yielding a possibly different reduced derivation Π'.

PLAN

∞-TERMS

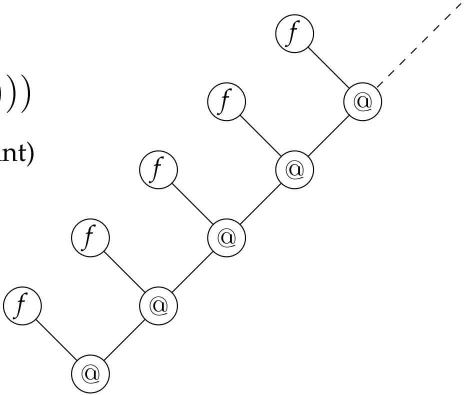


Variable x **Abstraction** $\lambda x.u$ **Application** $u\,v$

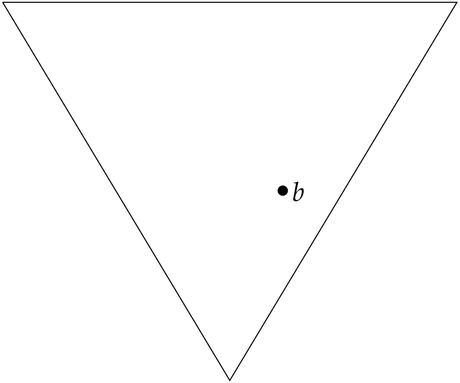
001-TERMS

$$f^\omega := f(f(f(\ldots)))$$

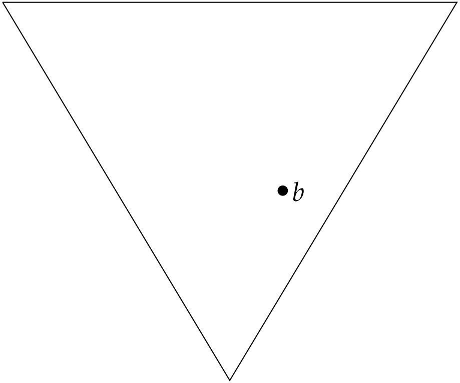
i.e. $f^\omega = f(f^\omega)$ (fixpoint)



001-TERMS

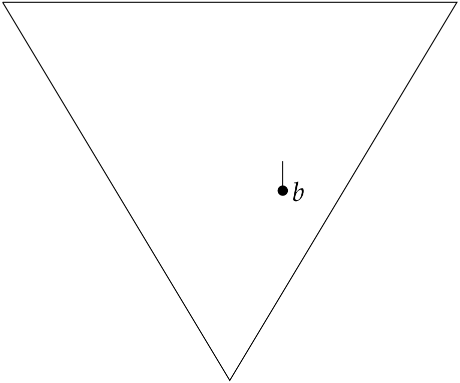


001-TERMS



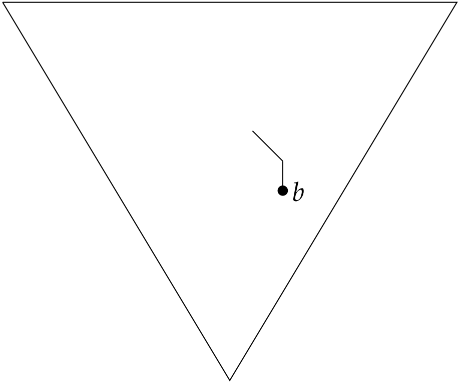
- Start from $b \in \mathrm{supp}(t)$

001-terms



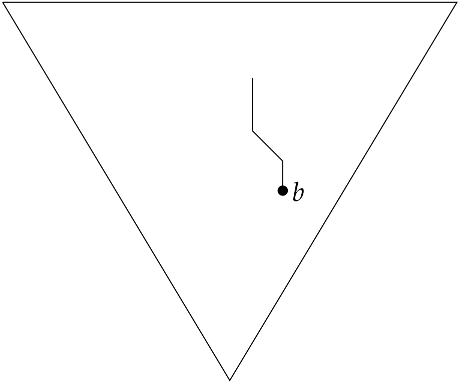
- Start from $b \in \mathrm{supp}(t)$
- Move $\uparrow$ or $\nwarrow$

001-TERMS



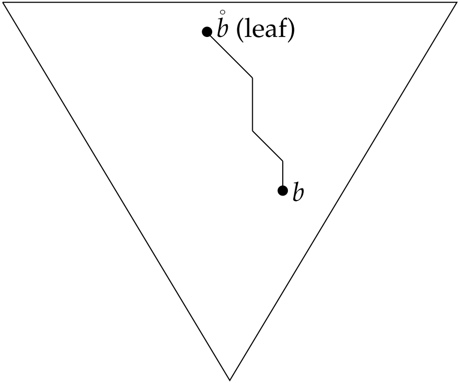
- Start from $b \in \mathrm{supp}(t)$
- Move $\uparrow$ or $\nwarrow$

001-TERMS



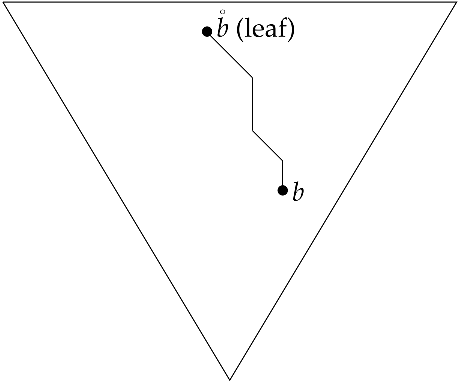
- Start from $b \in \mathrm{supp}(t)$
- Move $\uparrow$ or $\nwarrow$

001-TERMS



- Start from $b \in \mathrm{supp}(t)$
- Move $\uparrow$ or $\nwarrow$
- A leaf $\mathring{b}$ must be reached

001-TERMS

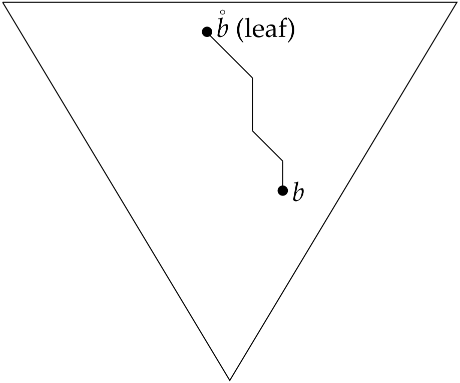


- Start from $b \in \mathrm{supp}(t)$
- Move $\uparrow$ or $\nwarrow$
- A leaf $\mathring{b}$ must be reached
- $\searrow$-induction on 001-terms

001-TERMS

Λ^{001}: the set of ∞-terms t s.t.:

$$b \text{ is an infinite branch of } t \Rightarrow \texttt{ad}(b) = \infty.$$



- Start from $b \in \text{supp}(t)$
- Move $\uparrow$ or $\nwarrow$
- A leaf $\mathring{b}$ must be reached
- $\searrow$-induction on 001-terms

RIGID TYPES (SYSTEM S)

- The set Types is defined by the coinductive grammar

$$S_k,\ T\ ::=\ \alpha \mid (S_k)_{k\in K} \rightarrow T$$

RIGID TYPES (SYSTEM S)

- The set Types is defined by the coinductive grammar

$$S_k,\ T \ ::= \ \alpha \mid (S_k)_{k \in K} \to T$$

- $(S_k)_{k \in K}$: **sequence type** (with $K \subseteq \mathbb{N} - \{0, 1\}$).

RIGID TYPES (SYSTEM S)

- The set Types is defined by the coinductive grammar

$$S_k,\ T\ ::=\ \alpha \mid (S_k)_{k\in K} \to T$$

- $(S_k)_{k\in K}$: **sequence type** (with $K \subseteq \mathbb{N} - \{0,1\}$).

- The relation $\equiv$ (between types or seq. types) is defined coinductively:
  - $\alpha \equiv \alpha$.
  - $(S_k)_{k\in K} \to T \equiv (S'_k)_{k\in K'} \to T'$ if $(S_k)_{k\in K} \equiv (S'_k)_{k'\in K'}$ and $T \equiv T'$.

RIGID TYPES (SYSTEM S)

- The set Types is defined by the coinductive grammar

$$S_k,\ T \ ::= \ \alpha \mid (S_k)_{k \in K} \to T$$

- $(S_k)_{k\in K}$: **sequence type** (with $K \subseteq \mathbb{N} - \{0,1\}$).

- The relation $\equiv$ (between types or seq. types) is defined coinductively:
  - $\alpha \equiv \alpha$.
  - $(S_k)_{k\in K} \to T \equiv (S'_k)_{k\in K'} \to T'$ if $(S_k)_{k\in K} \equiv (S'_k)_{k' \in K'}$ and $T \equiv T'$.
  - $(S_k)_{k\in K} \equiv (S'_k)_{k\in K'}$ if there is a bijection $\rho : K \to K'$ s.t. $\forall k \in K,\ S_k \equiv S'_{\sigma(k)}$ (such a ρ is called a **root isomorphism**).

RIGID TYPES (SYSTEM S)

- The set Types is defined by the coinductive grammar

$$S_k,\ T \ ::=\ \alpha \mid (S_k)_{k\in K} \to T$$

- $(S_k)_{k\in K}$: **sequence type** (with $K \subseteq \mathbb{N} - \{0,1\}$).

- The relation $\equiv$ (between types or seq. types) is defined coinductively:
  - $\alpha \equiv \alpha$.
  - $(S_k)_{k\in K} \to T \equiv (S'_k)_{k\in K'} \to T'$ if $(S_k)_{k\in K} \equiv (S'_k)_{k'\in K'}$ and $T \equiv T'$.
  - $(S_k)_{k\in K} \equiv (S'_k)_{k\in K'}$ if there is a bijection $\rho : K \to K'$ s.t. $\forall k \in K,\ S_k \equiv S'_{\sigma(k)}$ (such a ρ is called a **root isomorphism**).

- Notion of full **type** (resp. **sequence type**) **isomorphism** when $T \equiv T'$ (resp. $(S_k)_{k\in K} \equiv (S'_k)_{k\in K'}$).

S-DERIVATIONS

The set Deriv of rigid derivations is *coinductively* generated by:

$$\frac{}{x : (T)_k \vdash x : T}\ \mathtt{ax} \qquad \frac{C \vdash t : T\ (\text{tr. } 0)}{C - x \vdash \lambda x.t : C(x) \to T}\ \mathtt{abs}$$

$$\frac{C \vdash t : (S_k)_{k \in K} \to T\ (\text{tr. } 1) \qquad \left(D_k \vdash u : S'_k\ \ (\text{tr. } k)\right)^{k \in K'}}{C \cup \bigcup_{k \in K} D_k \vdash t(u) : T}\ \mathtt{app}$$

S-DERIVATIONS

The set Deriv of rigid derivations is *coinductively* generated by:

$$\frac{}{x : (T)_k \vdash x : T}\ \texttt{ax} \qquad \frac{C \vdash t : T\ \text{(tr. 0)}}{C - x \vdash \lambda x.t : C(x) \to T}\ \texttt{abs}$$

$$\frac{C \vdash t : (S_k)_{k \in K} \to T\ \text{(tr. 1)} \qquad \left(D_k \vdash u : S'_k\ \text{(tr. } k\text{)}\right)^{k \in K'}}{C \cup \bigcup_{k \in K} D_k \vdash t(u) : T}\ \texttt{app}$$

- Track constraints: in red, *e.g.* if P types an abstraction at position $a \in \mathbb{N}^*$, we must have $a \cdot 0 \in \mathrm{supp}(P)$.

s-Derivations

The set Deriv of rigid derivations is *coinductively* generated by:

$$\dfrac{}{x : (T)_k \vdash x : T}\ \texttt{ax} \qquad \dfrac{C \vdash t : T\ \color{red}{(\text{tr. } 0)}}{C - x \vdash \lambda x.t : C(x) \to T}\ \texttt{abs}$$

$$\dfrac{C \vdash t : (S_k)_{k \in K} \to T\ \color{red}{(\text{tr. } 1)} \qquad \left(D_k \vdash u : S'_k\ \color{red}{(\text{tr. } k)}\right)^{k \in K'}}{C \cup \bigcup_{k \in K} D_k \vdash t(u) : T}\ \texttt{app}$$

- Track constraints: in red, *e.g.* if P types an abstraction at position $a \in \mathbb{N}^*$, we must have $a \cdot 0 \in \mathrm{supp}(P)$.
- Application constraint 1: $(S_k)_{k \in K} = (S'_k)_{k' \in K'}$, also written $\mathrm{L} = \mathrm{R}$

s-Derivations

The set Deriv of rigid derivations is *coinductively* generated by:

$$\frac{}{x : (T)_k \vdash x : T}\ \texttt{ax} \qquad \frac{C \vdash t : T\ \textcolor{red}{(\text{tr. } 0)}}{C - x \vdash \lambda x.t : C(x) \to T}\ \texttt{abs}$$

$$\frac{C \vdash t : (S_k)_{k \in K} \to T\ \textcolor{red}{(\text{tr. } 1)} \qquad \left(D_k \vdash u : S'_k\ \textcolor{red}{(\text{tr. } k)}\right)^{k \in K'}}{C \cup \bigcup_{k \in K} D_k \vdash t(u) : T}\ \texttt{app}$$

- Track constraints: in red, *e.g.* if P types an abstraction at position $a \in \mathbb{N}^*$, we must have $a \cdot 0 \in \operatorname{supp}(P)$.
- Application constraint 1: $(S_k)_{k \in K} = (S'_k)_{k' \in K'}$, also written $\mathrm{L} = \mathrm{R}$
- Application constraint 2: the contexts must be disjoint, so that no **track conflict** occurs.

Main features

- Subject reduction is deterministic:

MAIN FEATURES

- Subject reduction is deterministic:
  - Assume P types $(\lambda x.r)s$. If there is an axiom rule typing x on track 5 (#5-`ax`), by typing constraint, there will also be an argument derivation P_5 typing s on track 5, concluded by exactly the same type S_5

MAIN FEATURES

- Subject reduction is deterministic:
  - Assume P types $(\lambda x.r)s$. If there is an axiom rule typing x on track 5 (#5-`ax`), by typing constraint, there will also be an argument derivation P_5 typing s on track 5, concluded by exactly the same type S_5
  - During reduction, #5-`ax` will be replaced by P_5, even if there are other P_k concluded by $S = S_5$

MAIN FEATURES

- Subject reduction is deterministic:
  - Assume P types $(\lambda x.r)s$. If there is an axiom rule typing x on track 5 (#5-`ax`), by typing constraint, there will also be an argument derivation P_5 typing s on track 5, concluded by exactly the same type S_5
  - During reduction, #5-`ax` will be replaced by P_5, even if there are other P_k concluded by $S = S_5$

- **Question 2:** loss of expressivity compared to multiset intersection systems ?

MAIN FEATURES

- Subject reduction is deterministic:
  - Assume P types $(\lambda x.r)s$. If there is an axiom rule typing x on track 5 (#5-`ax`), by typing constraint, there will also be an argument derivation P_5 typing s on track 5, concluded by exactly the same type S_5
  - During reduction, #5-`ax` will be replaced by P_5, even if there are other P_k concluded by $S = S_5$

- **Question 2:** loss of expressivity compared to multiset intersection systems ?
- Every symbol is identified (notion of **biposition**): possibility of trace a type through typing rules, of residual of a type after subj. red.

SYSTEM $\mathscr{M}$

- We set $\text{Types}_{\mathscr{M}} := \text{Types}\,/\equiv$ and $\mathscr{M}(\text{Types}_{\mathscr{M}}) := \mathtt{S}(\text{Types})/\equiv$.

SYSTEM $\mathscr{M}$

- We set $\text{Types}_{\mathscr{M}} := \text{Types} / \equiv$ and $\mathscr{M}(\text{Types}_{\mathscr{M}}) := \mathtt{S}(\text{Types}) / \equiv$.
- Countable sum of multisets types: not a problem.

SYSTEM $\mathscr{M}$

- We set $\mathrm{Types}_{\mathscr{M}} := \mathrm{Types}/\equiv$ and $\mathscr{M}(\mathrm{Types}_{\mathscr{M}}) := \mathtt{S}(\mathrm{Types})/\equiv$.
- Countable sum of multisets types: not a problem.
- Rules of System $\mathscr{M}$: the same as the rules of $\mathscr{M}_0$ but taken coinductively and using the (multiset) types of $\mathrm{Types}_{\mathscr{M}}$ and $\mathscr{M}(\mathrm{Types}_{\mathscr{M}})$.

SYSTEM $\mathscr{M}$

- We set $\text{Types}_{\mathscr{M}} := \text{Types} / \equiv$ and $\mathscr{M}(\text{Types}_{\mathscr{M}}) := \mathtt{S}(\text{Types})/\equiv$.
- Countable sum of multisets types: not a problem.
- Rules of System $\mathscr{M}$: the same as the rules of $\mathscr{M}_0$ but taken coinductively and using the (multiset) types of $\text{Types}_{\mathscr{M}}$ and $\mathscr{M}(\text{Types}_{\mathscr{M}})$.
- The $\mathtt{app}$-rule also relies on the multiset equality $[\sigma_i]_{i \in I} = [\sigma'_i]_{i \in I'}$.

SYSTEM $\mathscr{M}$

- We set $\text{Types}_{\mathscr{M}} := \text{Types} / \equiv$ and $\mathscr{M}(\text{Types}_{\mathscr{M}}) := \mathtt{S}(\text{Types})/\equiv$.
- Countable sum of multisets types: not a problem.
- Rules of System $\mathscr{M}$: the same as the rules of $\mathscr{M}_0$ but taken coinductively and using the (multiset) types of $\text{Types}_{\mathscr{M}}$ and $\mathscr{M}(\text{Types}_{\mathscr{M}})$.
- The app-rule also relies on the multiset equality $[\sigma_i]_{i\in I} = [\sigma'_i]_{i\in I'}$.
- System $\mathscr{M}$ (as system $\mathtt{S}$) is unsound ($\Delta\,\Delta$ is typable).

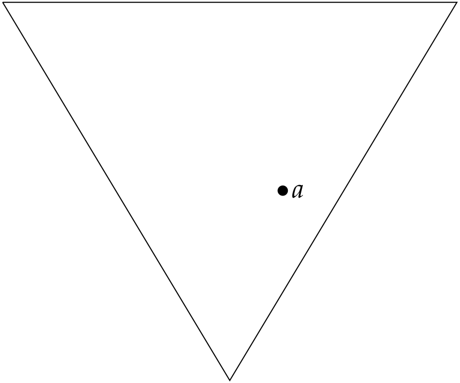
SYSTEM $\mathscr{M}$

- We set $\mathrm{Types}_{\mathscr{M}} := \mathrm{Types} / \equiv$ and $\mathscr{M}(\mathrm{Types}_{\mathscr{M}}) := \mathtt{S}(\mathrm{Types})/ \equiv$.
- Countable sum of multisets types: not a problem.
- Rules of System $\mathscr{M}$: the same as the rules of $\mathscr{M}_0$ but taken coinductively and using the (multiset) types of $\mathrm{Types}_{\mathscr{M}}$ and $\mathscr{M}(\mathrm{Types}_{\mathscr{M}})$.
- The $\mathtt{app}$-rule also relies on the multiset equality $[\sigma_i]_{i\in I} = [\sigma'_i]_{i\in I'}$.
- System $\mathscr{M}$ (as system $\mathtt{S}$) is unsound ($\Delta\,\Delta$ is typable).
- Use of multisets: cannot distinguish two occ. of the same type in a multiset, trace a type inside a derivation, define residuals of a type after sub. red.

Plan

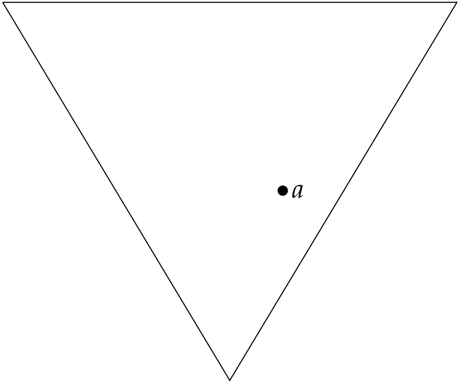
Global Typing Constraints

- P is a tree, $A := \mathrm{supp}(P)$ and $P(a) = C(a) \vdash t|_a : T(a)$ for all a.



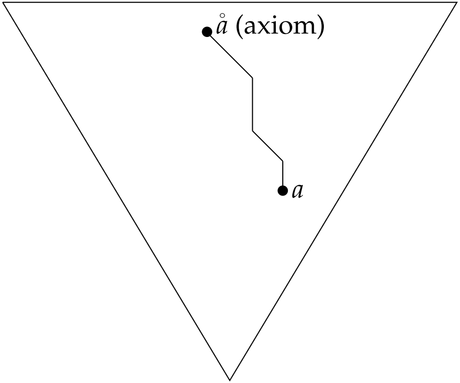
GLOBAL TYPING CONSTRAINTS

- P is a tree, $A := \mathrm{supp}(P)$ and $P(a) = C(a) \vdash t|_a : T(a)$ for all a.



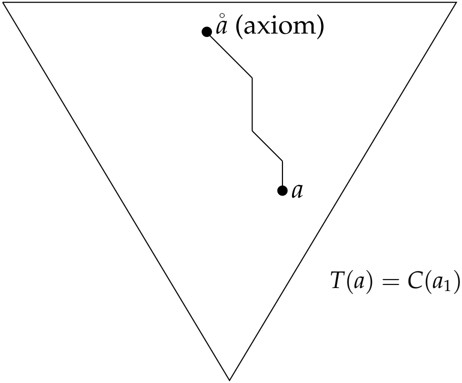
GLOBAL TYPING CONSTRAINTS

- P is a tree, $A := \mathrm{supp}(P)$ and $P(a) = C(a) \vdash t|_a : T(a)$ for all a.



GLOBAL TYPING CONSTRAINTS

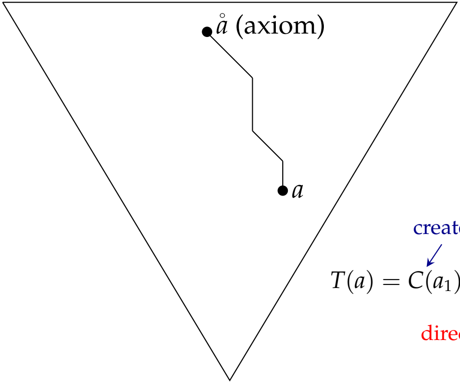
- P is a tree, $A := \mathrm{supp}(P)$ and $P(a) = C(a) \vdash t|_a : T(a)$ for all a.



$$T(a) = C(a_1)(y_1) \to \ldots \to C(a_p)(y_p) \to \mathrm{Hd}^q(T(\mathring{a}))$$

GLOBAL TYPING CONSTRAINTS

- P is a tree, $A := \mathrm{supp}(P)$ and $P(a) = C(a) \vdash t|_a : T(a)$ for all a.



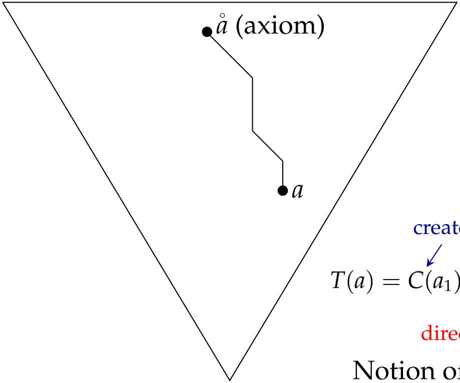
created by abstraction $\ominus$

$$T(a) = C(a_1)(y_1) \to \ldots \to C(a_p)(y_p) \to \mathrm{Hd}^q(T(\mathring{a}))$$

directly created in an axiom $\oplus$

GLOBAL TYPING CONSTRAINTS

- P is a tree, $A := \mathrm{supp}(P)$ and $P(a) = C(a) \vdash t|_a : T(a)$ for all a.



created by abstraction $\ominus$

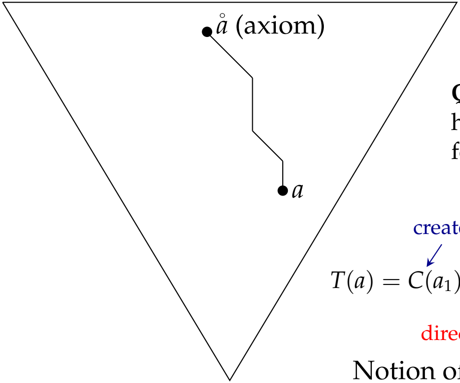
$$T(a) = C(a_1)(y_1) \to \ldots \to C(a_p)(y_p) \to \mathrm{Hd}^q(T(\mathring{a}))$$

directly created in an axiom $\oplus$

Notion of **referent biposition**

Global Typing Constraints

- P is a tree, $A := \mathrm{supp}(P)$ and $P(a) = C(a) \vdash t|_a : T(a)$ for all a.



Question 1:
how do we get $\mathrm{L}(a) = \mathrm{R}(a)$ for all `app`-node a?

created by abstraction $\ominus$

$$T(a) = C(a_1)(y_1) \to \ldots \to C(a_p)(y_p) \to \mathrm{Hd}^q(T(\mathring{a}))$$

directly created in an axiom $\oplus$

Notion of **referent biposition**

THE PROBLEM OF COLLAPSE

$$T(a) = C(a_1)(y_1) \to \ldots \to C(a_p)(y_p) \to \mathrm{Hd}^p(T(\mathring{a}))$$

- Every $\mathtt{S}$-derivation P can be seen as a set of symbols, pointed by **bipositions** (a position points to a jugdment inside P, a biposition points to a type symbol ($\alpha, \to$) inside a jugdment inside P).
- Evey biposition (or so...) comes from a biposition in a type given in an axiom rule.
  Notion of **referent biposition** (set $\mathrm{ref}(P)$).
- In order to represent a $\mathscr{M}$-derivation Π by a $\mathtt{S}$-derivation P, we must associate to all axioms rules a parser $T(a)$ s.t. the **syntactic** equality $\mathrm{L}(a) = \mathrm{R}(a)$ holds for **every** application node.
- For now, let us just loosen the **synctactic equality** condition in the $\mathtt{app}$-rule.

HYBRID DERIVATIONS

Type system $\mathtt{S_h}$ is obtained from $\mathtt{S}$ by replacing the $\mathtt{app}$-rule by:

$$\frac{C \vdash t : (S_k)_{k \in K} \to T \;\text{(tr. 1)} \qquad \left(D_k \vdash u : S'_k \;\text{(tr. } k\text{)}\right)^{k \in K'}}{C \cup \bigcup_{k \in K} D_k \vdash t(u) : T}\,\mathtt{happ}$$

Constraint : $(S_k)_{k \in K} \equiv (S'_k)_{k' \in K'}$

HYBRID DERIVATIONS

Type system $\mathtt{S_h}$ is obtained from $\mathtt{S}$ by replacing the $\mathtt{app}$-rule by:

$$\dfrac{C \vdash t : (S_k)_{k\in K} \to T \; \textcolor{red}{(\text{tr. } 1)} \qquad \left(D_k \vdash u : S'_k \; \textcolor{red}{(\text{tr. } k)}\right)^{k \in K'}}{C \cup \bigcup_{k\in K} D_k \vdash t(u) : T}\;\mathtt{happ}$$

Constraint : $(S_k)_{k\in K} \equiv (S'_k)_{k'\in K'}$

- Thus, condition $\mathrm{L}(a) = \mathrm{R}(a)$ has been replaced by $\mathrm{L}(a) \equiv \mathrm{R}(a)$.

HYBRID DERIVATIONS

Type system $\mathtt{S_h}$ is obtained from $\mathtt{S}$ by replacing the $\mathtt{app}$-rule by:

$$\dfrac{C \vdash t : (S_k)_{k \in K} \to T \ (\text{tr. } 1) \qquad \left(D_k \vdash u : S'_k \ (\text{tr. } k) \right)^{k \in K'}}{C \cup \bigcup_{k \in K} D_k \vdash t(u) : T} \ \mathtt{happ}$$

Constraint : $(S_k)_{k \in K} \equiv (S'_k)_{k' \in K'}$

- Thus, condition $\mathrm{L}(a) = \mathrm{R}(a)$ has been replaced by $\mathrm{L}(a) \equiv \mathrm{R}(a)$.
- Each hybrid derivation P collapses into a $\mathscr{M}$-derivation Π.

HYBRID DERIVATIONS

Type system $\mathtt{S_h}$ is obtained from $\mathtt{S}$ by replacing the $\mathtt{app}$-rule by:

$$\frac{C \vdash t : (S_k)_{k \in K} \to T \ \textcolor{red}{(\text{tr. } 1)} \qquad \left(D_k \vdash u : S'_k \ \textcolor{red}{(\text{tr. } k)} \right)^{k \in K'}}{C \cup \bigcup_{k \in K} D_k \vdash t(u) : T} \ \mathtt{happ}$$

Constraint : $(S_k)_{k \in K} \equiv (S'_k)_{k' \in K'}$

- Thus, condition $\mathrm{L}(a) = \mathrm{R}(a)$ has been replaced by $\mathrm{L}(a) \equiv \mathrm{R}(a)$.
- Each hybrid derivation P collapses into a $\mathscr{M}$-derivation Π.
- For any $\mathscr{M}$-derivation Π, easy to find a hybrid P s.t. $\overline{P} = \Pi$.

OPERABLE DERIVATION

- Let P be a hybrid derivation typing t.

OPERABLE DERIVATION

- Let P be a hybrid derivation typing t.
  - If a corresponds to a redex $(\lambda x.r)s$ inside t, a root isomorphism $\rho_a : \mathrm{L}(a) \to \mathrm{R}(a)$ tells us how to perform subject reduction.

OPERABLE DERIVATION

- Let P be a hybrid derivation typing t.
  - If a corresponds to a redex $(\lambda x.r)s$ inside t, a root isomorphism $\rho_a : \mathrm{L}(a) \to \mathrm{R}(a)$ tells us how to perform subject reduction.
  - Say $\rho_a(5) = 7$. Then, above a, there is an x-axiom rule on track 5 (#5-`ax`) and argument derivation $P|_{a\cdot 7}$ on track 7.

OPERABLE DERIVATION

- Let P be a hybrid derivation typing t.
  - If a corresponds to a redex $(\lambda x.r)s$ inside t, a root isomorphism $\rho_a : \mathrm{L}(a) \to \mathrm{R}(a)$ tells us how to perform subject reduction.
  - Say $\rho_a(5) = 7$. Then, above a, there is an x-axiom rule on track 5 (#5-`ax`) and argument derivation $P|_{a \cdot 7}$ on track 7.
  - Then, during reduction, #5-`ax` must be replaced by $P|_{a \cdot 7}$

OPERABLE DERIVATION

- Let P be a hybrid derivation typing t.
  - If a corresponds to a redex $(\lambda x.r)s$ inside t, a root isomorphism $\rho_a : \mathrm{L}(a) \to \mathrm{R}(a)$ tells us how to perform subject reduction.
  - Say $\rho_a(5) = 7$. Then, above a, there is an x-axiom rule on track 5 (#5-`ax`) and argument derivation $P|_{a\cdot 7}$ on track 7.
  - Then, during reduction, #5-`ax` must be replaced by $P|_{a\cdot 7}$
- **Interfaces:**
  - A **complete interface** is given by a family of (full) sequence type isomorphisms $\phi_a : \mathrm{L}(a) \to \mathrm{R}(a)$ when a ranges over the `app`-nodes of P.

OPERABLE DERIVATION

- Let P be a hybrid derivation typing t.
  - If a corresponds to a redex $(\lambda x.r)s$ inside t, a root isomorphism $\rho_a : \mathrm{L}(a) \to \mathrm{R}(a)$ tells us how to perform subject reduction.
  - Say $\rho_a(5) = 7$. Then, above a, there is an x-axiom rule on track 5 (#5-`ax`) and argument derivation $P|_{a\cdot 7}$ on track 7.
  - Then, during reduction, #5-`ax` must be replaced by $P|_{a\cdot 7}$
- **Interfaces:**
  - A **complete interface** is given by a family of (full) sequence type isomorphisms $\phi_a : \mathrm{L}(a) \to \mathrm{R}(a)$ when a ranges over the `app`-nodes of P.
  - If b is the pos. of a redex, notion of residuals (of positions, bipositions and interfaces) after firing the redex P.

OPERABLE DERIVATION

- Let P be a hybrid derivation typing t.
  - If a corresponds to a redex $(\lambda x.r)s$ inside t, a root isomorphism $\rho_a : \mathrm{L}(a) \to \mathrm{R}(a)$ tells us how to perform subject reduction.
  - Say $\rho_a(5) = 7$. Then, above a, there is an x-axiom rule on track 5 (#5-`ax`) and argument derivation $P|_{a\cdot 7}$ on track 7.
  - Then, during reduction, #5-`ax` must be replaced by $P|_{a\cdot 7}$
- **Interfaces:**
  - A **complete interface** is given by a family of (full) sequence type isomorphisms $\phi_a : \mathrm{L}(a) \to \mathrm{R}(a)$ when a ranges over the `app`-nodes of P.
  - If b is the pos. of a redex, notion of residuals (of positions, bipositions and interfaces) after firing the redex P.
- An **operable derivation** is a hybrid derivation endowed with a complete interface (for each `app`-rule).

Representation Lemma

Lemma

Let Π a $\mathscr{M}$-derivation typing t and a reduction sequence $\mathscr{R}$ (of length $\leqslant \omega$) and P a hybrid representative of Π.
Any reduction choice sequence along $\mathscr{R}$ can be built-in inside a complete interface for P.

REPRESENTATION LEMMA

Lemma

Let Π a $\mathscr{M}$-derivation typing t and a reduction sequence $\mathscr{R}$ (of length $\leqslant \omega$) and P a hybrid representative of Π.
Any reduction choice sequence along $\mathscr{R}$ can be built-in inside a complete interface for P.

Intuition of the Proof:

- Consider a reduction sequence $t_0 \xrightarrow{b_0} t_1 \xrightarrow{b_1} t_2 \xrightarrow{b_2} \ldots$.

REPRESENTATION LEMMA

Lemma

Let Π a $\mathscr{M}$-derivation typing t and a reduction sequence $\mathscr{R}$ (of length $\leqslant \omega$) and P a hybrid representative of Π.
Any reduction choice sequence along $\mathscr{R}$ can be built-in inside a complete interface for P.

Intuition of the Proof:

- Consider a reduction sequence $t_0 \xrightarrow{b_0} t_1 \xrightarrow{b_1} t_2 \xrightarrow{b_2} \ldots$.
- Reduction step by reduction step, choose an interface I_i representing the reduction choice (w.r.t. the derivation P_i typing t_i the i-th of the sequence). It produces a reduced derivation P_{i+1} typing t_{i+1}.

REPRESENTATION LEMMA

Lemma

Let Π a $\mathscr{M}$-derivation typing t and a reduction sequence $\mathscr{R}$ (of length $\leqslant \omega$) and P a hybrid representative of Π.
Any reduction choice sequence along $\mathscr{R}$ can be built-in inside a complete interface for P.

Intuition of the Proof:

- Consider a reduction sequence $t_0 \xrightarrow{b_0} t_1 \xrightarrow{b_1} t_2 \xrightarrow{b_2} \ldots$.
- Reduction step by reduction step, choose an interface I_i representing the reduction choice (w.r.t. the derivation P_i typing t_i the i-th of the sequence). It produces a reduced derivation P_{i+1} typing t_{i+1}.
- Since each interface isomorphism of the reduced derivation is a residual an interface isomorphism, interface I_i can be lifted to P.

Plan

RESTATEMENT

Theorem

For all $\mathscr{M}$-derivation Π, there is a trivial $\mathtt{S}$-derivation P that collapses into Π.

RESTATEMENT

Theorem
For all $\mathscr{M}$-derivation Π, there is a trivial $\mathtt{S}$-derivation P that collapses into Π.

Claim
Every operable derivation P is isomorphic to a trivial derivation.

RESTATEMENT

Theorem

For all $\mathscr{M}$-derivation Π, there is a trivial $\mathtt{S}$-derivation P that collapses into Π.

Claim

Every operable derivation P is isomorphic to a trivial derivation.

Question: what is a isomorphism of o.d. $\Psi : P_1 \to P_2$?

RESTATEMENT

Theorem

For all $\mathscr{M}$-derivation Π, there is a trivial $\mathtt{S}$-derivation P that collapses into Π.

Claim

Every operable derivation P is isomorphic to a trivial derivation.

Question: what is a isomorphism of o.d. $\Psi : P_1 \to P_2$?

- A well-behaved bijection from $\operatorname{supp}(P_1)$ to $\operatorname{supp}(P_2)$.

RESTATEMENT

Theorem

For all $\mathscr{M}$-derivation Π, there is a trivial $\mathtt{S}$-derivation P that collapses into Π.

Claim

Every operable derivation P is isomorphic to a trivial derivation.

Question: what is a isomorphism of o.d. $\Psi : P_1 \to P_2$?

- A well-behaved bijection from $\mathrm{supp}(P_1)$ to $\mathrm{supp}(P_2)$.
- Between each associated axioms rules of P_1 and P_2, a type isomorphism (w.r.t. the former bijection).

RESTATEMENT

Theorem
For all $\mathscr{M}$-derivation Π, there is a trivial $\mathtt{S}$-derivation P that collapses into Π.

Claim
Every operable derivation P is isomorphic to a trivial derivation.

Question: what is a isomorphism of o.d. $\Psi : P_1 \to P_2$?

- A well-behaved bijection from $\mathrm{supp}(P_1)$ to $\mathrm{supp}(P_2)$.
- Between each associated axioms rules of P_1 and P_2, a type isomorphism (w.r.t. the former bijection).
- Commutation with interface isomorphisms of P_1 and P_2.

A FEW IDEAS

- Mainly a matter of finding good track values for referent bipositions.

A FEW IDEAS

- Mainly a matter of finding good track values for referent bipositions.
- Typability (in $\mathtt{S}$) $\not\Rightarrow$ normalizability. Reasoning on NF and expanding cannot work.

A FEW IDEAS

- Mainly a matter of finding good track values for referent bipositions.
- Typability (in $\mathtt{S}$) $\not\Rightarrow$ normalizability. Reasoning on NF and expanding cannot work.
- Each interface isomorphism ϕ_a induces a partial function $\widetilde{\phi_a}$ from $\mathrm{ref}(P)$ to $\mathrm{ref}(P)$: $\widetilde{\phi_a}$ tells us which tracks should be equal to have $\mathrm{L}(a) = \mathrm{R}(a)$.

A FEW IDEAS

- Mainly a matter of finding good track values for referent bipositions.
- Typability (in $\mathtt{S}$) $\not\Rightarrow$ normalizability. Reasoning on NF and expanding cannot work.
- Each interface isomorphism ϕ_a induces a partial function $\widetilde{\phi_a}$ from $\mathrm{ref}(P)$ to $\mathrm{ref}(P)$: $\widetilde{\phi_a}$ tells us which tracks should be equal to have $\mathrm{L}(a) = \mathrm{R}(a)$.
- This induces a first order theory on track values at ref. biposition. We must check that this theory does not equate two tracks of **brother** referents.

A FEW IDEAS

- Mainly a matter of finding good track values for referent bipositions.
- Typability (in $\mathtt{S}$) $\not\Rightarrow$ normalizability. Reasoning on NF and expanding cannot work.
- Each interface isomorphism ϕ_a induces a partial function $\widetilde{\phi_a}$ from $\mathrm{ref}(P)$ to $\mathrm{ref}(P)$: $\widetilde{\phi_a}$ tells us which tracks should be equal to have $\mathrm{L}(a) = \mathrm{R}(a)$.
- This induces a first order theory on track values at ref. biposition. We must check that this theory does not equate two tracks of **brother** referents.
- Each referent biposition is in $\mathrm{dom}(\widetilde{\phi_a}) \cup \mathrm{codom}(\widetilde{\phi_a})$ for at most one a in positive polarity (resp. negative polarity). Uniqueness of **consumption**.

A FEW IDEAS

- Mainly a matter of finding good track values for referent bipositions.
- Typability (in $\mathtt{S}$) $\not\Rightarrow$ normalizability. Reasoning on NF and expanding cannot work.
- Each interface isomorphism ϕ_a induces a partial function $\widetilde{\phi_a}$ from $\mathrm{ref}(P)$ to $\mathrm{ref}(P)$: $\widetilde{\phi_a}$ tells us which tracks should be equal to have $\mathrm{L}(a) = \mathrm{R}(a)$.
- This induces a first order theory on track values at ref. biposition. We must check that this theory does not equate two tracks of **brother** referents.
- Each referent biposition is in $\mathrm{dom}(\widetilde{\phi_a}) \cup \mathrm{codom}(\widetilde{\phi_a})$ for at most one a in positive polarity (resp. negative polarity). Uniqueness of **consumption**.
- When a referent biposition occurs negatively in $\mathrm{dom}(\widetilde{\phi_a})$, then a redex is hiding somewhere. It can be avoided by an *ad hoc* reduction strategy (**collapsing strategy**).

A Few Ideas

- Mainly a matter of finding good track values for referent bipositions.
- Typability (in $\mathtt{S}$) $\not\Rightarrow$ normalizability. Reasoning on NF and expanding cannot work.
- Each interface isomorphism ϕ_a induces a partial function $\widetilde{\phi_a}$ from $\mathrm{ref}(P)$ to $\mathrm{ref}(P)$: $\widetilde{\phi_a}$ tells us which tracks should be equal to have $\mathrm{L}(a) = \mathrm{R}(a)$.
- This induces a first order theory on track values at ref. biposition. We must check that this theory does not equate two tracks of **brother** referents.
- Each referent biposition is in $\mathrm{dom}(\widetilde{\phi_a}) \cup \mathrm{codom}(\widetilde{\phi_a})$ for at most one a in positive polarity (resp. negative polarity). Uniqueness of **consumption**.
- When a referent biposition occurs negatively in $\mathrm{dom}(\widetilde{\phi_a})$, then a redex is hiding somewhere. It can be avoided by an *ad hoc* reduction strategy (**collapsing strategy**).
- At last, we notice that $\widetilde{\phi_a}(r_1) = r_2$ implies $\mathtt{ad}(r_1) < \mathtt{ad}(r_2)$ when r_1 occurs with a positive polarity.

Plan

SUMMARY AND FUTURE WORKS

- System $\mathtt{S}$ (i=sequence) is very **low-level** compared to system $\mathscr{M}$ (i=multiset). A $\mathtt{S}$-derivation can easily **collapse** into a $\mathscr{M}$-derivation.

SUMMARY AND FUTURE WORKS

- System $\mathtt{S}$ (i=sequence) is very **low-level** compared to system $\mathscr{M}$ (i=multiset). A $\mathtt{S}$-derivation can easily **collapse** into a $\mathscr{M}$-derivation.
- System $\mathtt{S}$ is more **fine-grained** (derivations can be parsed, proper notions of residuals, possibility to express a useful validity criterion w.r.t. WN).

Summary and Future Works

- System $\mathtt{S}$ (i=sequence) is very **low-level** compared to system $\mathscr{M}$ (i=multiset). A $\mathtt{S}$-derivation can easily **collapse** into a $\mathscr{M}$-derivation.
- System $\mathtt{S}$ is more **fine-grained** (derivations can be parsed, proper notions of residuals, possibility to express a useful validity criterion w.r.t. WN).
- However, the use of **syntactic** equality (in system $\mathtt{S}$) seems very limitative and constraining compared to the use of **multiset** equality (in system $\mathscr{M}$).

SUMMARY AND FUTURE WORKS

- System $\mathtt{S}$ (i=sequence) is very **low-level** compared to system $\mathscr{M}$ (i=multiset). A $\mathtt{S}$-derivation can easily **collapse** into a $\mathscr{M}$-derivation.
- System $\mathtt{S}$ is more **fine-grained** (derivations can be parsed, proper notions of residuals, possibility to express a useful validity criterion w.r.t. WN).
- However, the use of **syntactic** equality (in system $\mathtt{S}$) seems very limitative and constraining compared to the use of **multiset** equality (in system $\mathscr{M}$).
- Representation Theorem: actually **no loss of expressivity** in system $\mathtt{S}$ since every $\mathscr{M}$-derivation can be represented by mean of a $\mathtt{S}$-derivation alongside with its heuristic dynamic features (reduction choices).

SUMMARY AND FUTURE WORKS

- System $\mathtt{S}$ (i=sequence) is very **low-level** compared to system $\mathscr{M}$ (i=multiset). A $\mathtt{S}$-derivation can easily **collapse** into a $\mathscr{M}$-derivation.
- System $\mathtt{S}$ is more **fine-grained** (derivations can be parsed, proper notions of residuals, possibility to express a useful validity criterion w.r.t. WN).
- However, the use of **syntactic** equality (in system $\mathtt{S}$) seems very limitative and constraining compared to the use of **multiset** equality (in system $\mathscr{M}$).
- Representation Theorem: actually **no loss of expressivity** in system $\mathtt{S}$ since every $\mathscr{M}$-derivation can be represented by mean of a $\mathtt{S}$-derivation alongside with its heuristic dynamic features (reduction choices).
- Proving that every term is typable in $\mathscr{M}$?

SUMMARY AND FUTURE WORKS

- System $\mathtt{S}$ (i=sequence) is very **low-level** compared to system $\mathscr{M}$ (i=multiset). A $\mathtt{S}$-derivation can easily **collapse** into a $\mathscr{M}$-derivation.
- System $\mathtt{S}$ is more **fine-grained** (derivations can be parsed, proper notions of residuals, possibility to express a useful validity criterion w.r.t. WN).
- However, the use of **syntactic** equality (in system $\mathtt{S}$) seems very limitative and constraining compared to the use of **multiset** equality (in system $\mathscr{M}$).
- Representation Theorem: actually **no loss of expressivity** in system $\mathtt{S}$ since every $\mathscr{M}$-derivation can be represented by mean of a $\mathtt{S}$-derivation alongside with its heuristic dynamic features (reduction choices).
- Proving that every term is typable in $\mathscr{M}$?
- Modify $\mathtt{S}$ to obtain a type theoretic characterization of **strongly normalizing (SN)** terms in Λ^{001} ?

Questions

Thank you for your attention !