Infinitary Intersection Types as Sequences

(A New Answer to Klop's Problem)

Pierre VIAL
IRIF, Paris 7

Elica meeting, Bologna

October 7, 2016

PLAN

HEREDITARY HEAD-NORMALIZATION

- **Head Normal Forms (HNF):** terms t of the form:

$$\lambda x_1 \dots x_p.x\, u_1 \dots u_q \qquad\qquad (p, q \geqslant 0)$$

HEREDITARY HEAD-NORMALIZATION

- **Head Normal Forms (HNF):** terms t of the form:

$$\lambda x_1 \ldots x_p . \boxed{x}\, u_1 \ldots u_q \qquad (p, q \geqslant 0)$$

head variable

head arguments

HEREDITARY HEAD-NORMALIZATION

- **Head Normal Forms (HNF):** terms t of the form:

$$\lambda x_1 \ldots x_p.\boxed{x}\, u_1 \ldots u_q \qquad (p, q \geqslant 0)$$

head variable head arguments

- A term is **head-normalizing (HN)** if it can be reduced to a HNF (in a finite number of steps)

HEREDITARY HEAD-NORMALIZATION

- **Head Normal Forms (HNF):** terms t of the form:

$$\lambda x_1 \ldots x_p.\boxed{x}\, u_1 \ldots u_q \qquad (p, q \geqslant 0)$$

head variable

head arguments

- A term is **head-normalizing (HN)** if it can be reduced to a HNF (in a finite number of steps)

- **Normal Forms (NF):** induction

$$t ::= \lambda x_1 \ldots x_p.x\, t_1 \ldots t_q \qquad (p, q \geqslant 0)$$

.

HEREDITARY HEAD-NORMALIZATION

- **Head Normal Forms (HNF):** terms t of the form:

$$\lambda x_1 \ldots x_p.\boxed{x}\, u_1 \ldots u_q \qquad (p, q \geqslant 0)$$

head variable head arguments

- A term is **head-normalizing (HN)** if it can be reduced to a HNF (in a finite number of steps)

- **Normal Forms (NF):** induction

$$t ::= \lambda x_1 \ldots x_p.x\, t_1 \ldots t_q \qquad (p, q \geqslant 0)$$

- A term is **weakly normalizing (WN)** if it can be reduced to a NF (in a finite number of steps)

.

HEREDITARY HEAD-NORMALIZATION

- **Head Normal Forms (HNF):** terms t of the form:

$$\lambda x_1 \dots x_p.\boxed{x}\, u_1 \dots u_q \qquad (p, q \geqslant 0)$$

head variable head arguments

- A term is **head-normalizing (HN)** if it can be reduced to a HNF (in a finite number of steps)

- **Normal Forms (NF):** induction

$$t ::= \lambda x_1 \dots x_p.x\, t_1 \dots t_q \qquad (p, q \geqslant 0)$$

- A term is **weakly normalizing (WN)** if it can be reduced to a NF (in a finite number of steps)
- Inductively, a term is WN if it is HN and all the head arguments are themselves WN.

HEREDITARY HEAD-NORMALIZATION

- **Head Normal Forms (HNF):** terms t of the form:

$$\lambda x_1 \ldots x_p.\boxed{x}\, u_1 \ldots u_q \qquad (p, q \geqslant 0)$$

head variable head arguments

- A term is **head-normalizing (HN)** if it can be reduced to a HNF (in a finite number of steps)

- Coinductively, a term is **hereditary head-normalizing (HHN)** if it can be reduced to a HNF and all the head arguments are themselves HHN.

.

KLOP'S PROBLEM

- The set of HN terms (resp. WN) terms have been *statically* characterized by various **intersection type assignement systems (ITS)**.

KLOP'S PROBLEM

- The set of HN terms (resp. WN) terms have been *statically* characterized by various **intersection type assignement systems (ITS)**.

- **Klop's Problem** [**early 90s**]**:** can the set of HHN terms can be characterized by an ITS ?

KLOP'S PROBLEM

- The set of HN terms (resp. WN) terms have been *statically* characterized by various **intersection type assignement systems (ITS)**.

- **Klop's Problem** [**early 90s**]**:** can the set of HHN terms can be characterized by an ITS ?

- **Tatsuta** [**07**]**:** an inductive ITS cannot do it.

KLOP'S PROBLEM

- The set of HN terms (resp. WN) terms have been *statically* characterized by various **intersection type assignement systems (ITS)**.
- **Klop's Problem** [**early 90s**]**:** can the set of HHN terms can be characterized by an ITS ?
- **Tatsuta** [**07**]**:** an inductive ITS cannot do it.
- Can a coinductive ITS characterize the set of HHN terms?

PURPOSES OF THIS TALK

- Present the key notions of **truncations** and **approximability** (meant to avoid *irrelevant* derivations).

PURPOSES OF THIS TALK

- Present the key notions of **truncations** and **approximability** (meant to avoid *irrelevant* derivations).
- Understand why **commutative intersection** (here, Gardner/de Carvalho's **multiset intersection**) is **unfit** to express those key notions.

PURPOSES OF THIS TALK

- Present the key notions of **truncations** and **approximability** (meant to avoid *irrelevant* derivations).
- Understand why **commutative intersection** (here, Gardner/de Carvalho's **multiset intersection**) is **unfit** to express those key notions.
- Present the coinductive type assignment system $\mathtt{S}$: intersection types are **sequences** of types, instead of *sets* of types (idempotent intersection fw.) or *multisets* of types (regular non-idempotent fw.).

Plan

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

Types (τ, σ_i): $\tau, \sigma_i := \alpha \in \mathscr{X} \mid [\sigma_i]_{i\in I} \to \tau$.

Context (Γ, Δ): assigns *intersection* types to variables.

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

$$\frac{\Gamma \vdash t : [\sigma_i]_{i\in I} \to \tau \qquad \left(\Delta_i \vdash u : \sigma_i\right)^{i\in I}}{\Gamma + \sum_{i\in I} \Delta_i \vdash t(u) : \tau}\ \text{app}$$

Remark

- Multiset equality: $[\sigma, \tau, \sigma] = [\sigma, \sigma, \tau] \neq [\sigma, \tau]$
- Multiplicative rules: accumulation of typing information .
- Possibility to **forget** the argument (empty multiset).

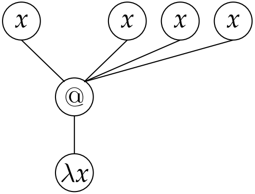
ALTERNATIVE PRESENTATION

Standard presentation

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

ALTERNATIVE PRESENTATION

Alternative presentation

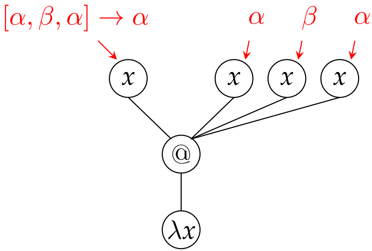


$\lambda x.xx$

- Indicate the arity of application rules.

ALTERNATIVE PRESENTATION

Alternative presentation

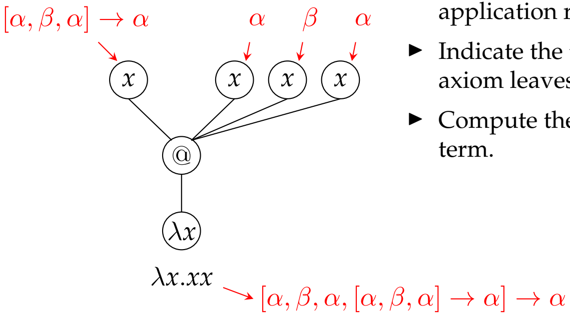


$\lambda x.xx$

- Indicate the arity of application rules.
- Indicate the types given in axiom leaves.

ALTERNATIVE PRESENTATION

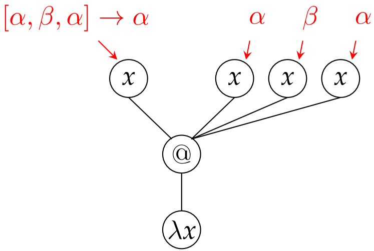
Alternative presentation



- Indicate the arity of application rules.
- Indicate the types given in axiom leaves.
- Compute the type of the term.

ALTERNATIVE PRESENTATION

Alternative presentation



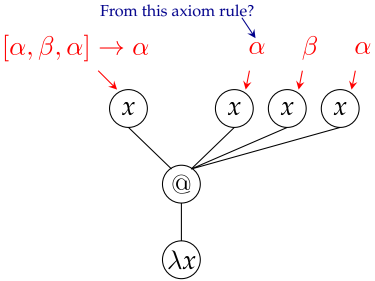
$\lambda x.xx$ $\to$ $[\alpha, \beta, \alpha, [\alpha, \beta, \alpha] \to \alpha] \to \alpha$

Where does this α come from?

- Indicate the arity of application rules.
- Indicate the types given in axiom leaves.
- Compute the type of the term.

Alternative Presentation

Alternative presentation



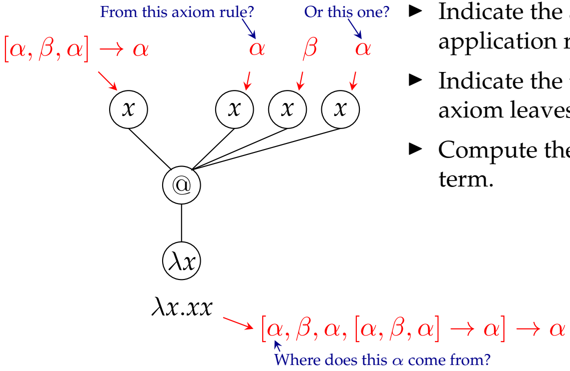
$\lambda x.xx$ $\to$ $[\alpha, \beta, \alpha, [\alpha, \beta, \alpha] \to \alpha] \to \alpha$

Where does this α come from?

- Indicate the arity of application rules.
- Indicate the types given in axiom leaves.
- Compute the type of the term.

ALTERNATIVE PRESENTATION

Alternative presentation



- Indicate the arity of application rules.
- Indicate the types given in axiom leaves.
- Compute the type of the term.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

“association”

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

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

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

$$\Pi_r \qquad \left(\dfrac{}{\cancel{x : [\sigma_i] \vdash x :}\ \boxed{\sigma_i}}\,\texttt{ax} \right)_{i \in I}$$

“association”

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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 derivation reducts Π'.

NORMALIZABILITY RESULTS

Proposition

A term is HN iff it is typable in $\mathscr{M}_0$.

NORMALIZABILITY RESULTS

Proposition

A term is HN iff it is typable in $\mathscr{M}_0$.

Proposition

A term is WN iff it is typable in $\mathscr{M}_0$ by using an **unforgetful** judgment.

Normalizability Results

Proposition

A term is HN iff it is typable in $\mathscr{M}_0$.

Proposition

A term is WN iff it is typable in $\mathscr{M}_0$ by using an **unforgetful** judgment.

Definition

A judgement $\Gamma \vdash t : \tau$ is **unforgetful** if there is no negative occurrence of $[\,]$ in Γ and no positive occurrence of $[\,]$ in τ.

NORMALIZABILITY RESULTS

Proposition

A term is HN iff it is typable in $\mathscr{M}_0$.

Proposition

A term is WN iff it is typable in $\mathscr{M}_0$ by using an **unforgetful** judgment.

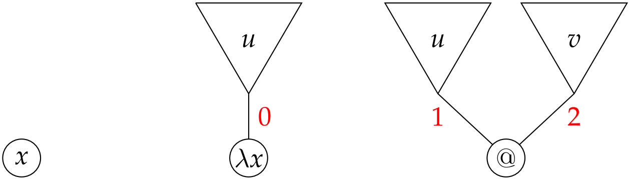
Definition

A judgement $\Gamma \vdash t : \tau$ is **unforgetful** if there is no negative occurrence of $[\,]$ in Γ and no positive occurrence of $[\,]$ in τ.

- $[\,]$ occurs negatively in $[\,] \to \tau$
- If $[\,]$ occurs negatively in σ_2 then $[\,]$ occurs positively in $[\sigma_1, \sigma_2, \sigma_3] \to \tau$ and so on.

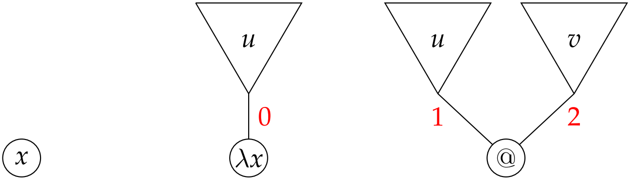
PLAN

∞-TERMS



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

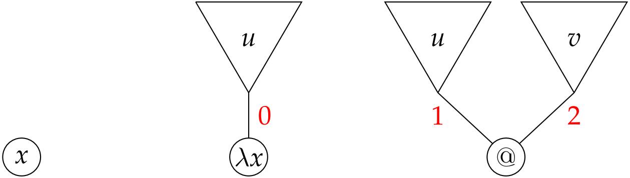
∞-TERMS



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

- **Position**: finite sequence in $\{0,1,2\}^*$, *e.g.* $0 \cdot 0 \cdot 2 \cdot 1 \cdot 2$.

∞-TERMS



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

- **Position**: finite sequence in $\{0, 1, 2\}^*$, *e.g.* $0 \cdot 0 \cdot 2 \cdot 1 \cdot 2$.
- **Applicative Depth (a.d.):** number of $\nearrow$-edges *e.g.*

$$\mathtt{ad}(1 \cdot 2 \cdot 2 \cdot 0 \cdot 2 \cdot 1 \cdot 2) = 4$$

001-TERMS

Λ^{001}: the set of ∞-terms t s.t.:

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

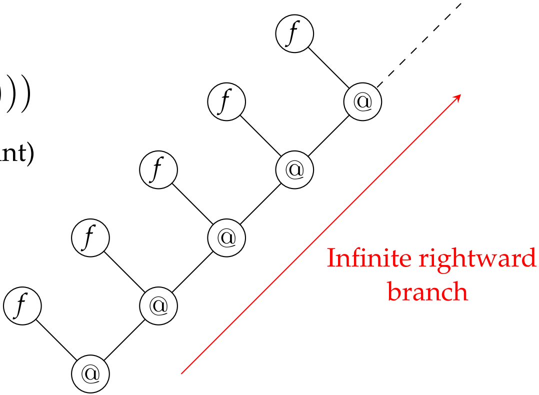
001-TERMS

Λ^{001}: the set of ∞-terms t s.t.:

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

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

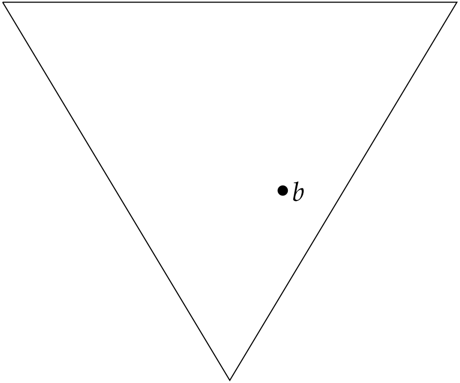
i.e. $f^\omega = f(f^\omega)$ (fixpoint)



001-TERMS

Λ^{001}: the set of ∞-terms t s.t.:

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

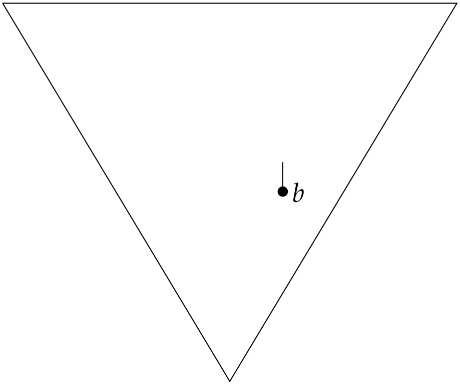


- Start from $b \in \mathtt{supp}(t)$

001-TERMS

Λ^{001}: the set of ∞-terms t s.t.:

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

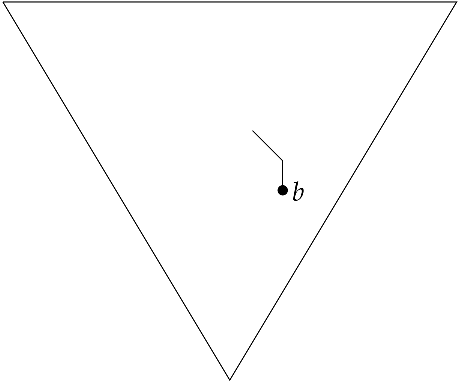


- Start from $b \in \mathtt{supp}(t)$
- Move $\uparrow$ or $\nwarrow$

001-TERMS

Λ^{001}: the set of ∞-terms t s.t.:

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

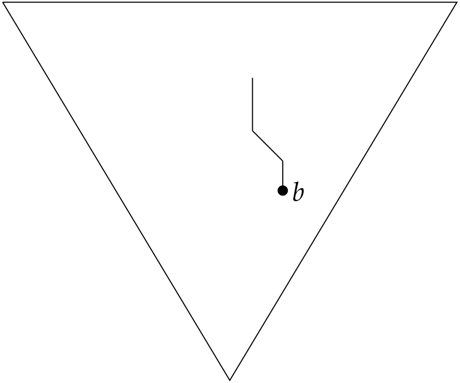


- Start from $b \in \mathtt{supp}(t)$
- Move $\uparrow$ or $\nwarrow$

001-TERMS

Λ^{001}: the set of ∞-terms t s.t.:

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

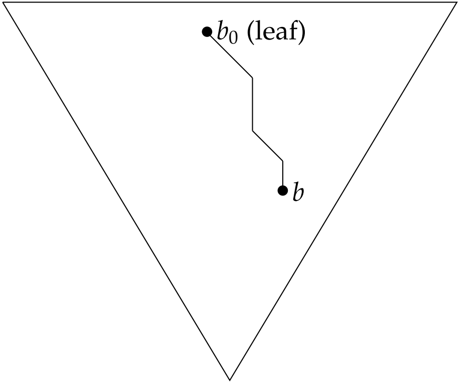


- Start from $b \in \mathtt{supp}(t)$
- Move $\uparrow$ or $\nwarrow$

001-TERMS

Λ^{001}: the set of ∞-terms t s.t.:

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



- Start from $b \in \mathtt{supp}(t)$
- Move $\uparrow$ or $\nwarrow$
- A leaf b_0 must be reached

STRONG CONVERGENCE

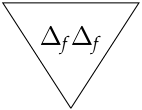
Definition

A reduction sequence $t_0 \xrightarrow{b_0} t_1 \xrightarrow{b_1} t_2 \xrightarrow{b_2} \ldots \xrightarrow{b_{n-1}} t_n \xrightarrow{b_n} \ldots$ is **strongly converging** if it is of finite length or if $\lim \mathtt{ad}(b_n) = \infty$.

STRONG CONVERGENCE

$$\Delta_f := \lambda x.f(xx) \qquad \Delta_f\Delta_f\text{: "Curry"}$$

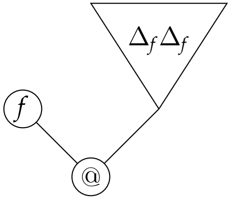
$$\Delta_f\Delta_f \to f(\Delta_f\Delta_f) \to f^2(\Delta_f\Delta_f) \to f^3(\Delta_f\Delta_f) \to f^4(\Delta_f\Delta_f) \to \ldots \to^\infty f^\omega$$



STRONG CONVERGENCE

$$\Delta_f := \lambda x.f(xx) \qquad \Delta_f\Delta_f\text{: "Curry"}$$

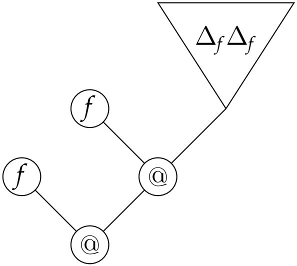
$$\Delta_f\Delta_f \to f(\Delta_f\Delta_f) \to f^2(\Delta_f\Delta_f) \to f^3(\Delta_f\Delta_f) \to f^4(\Delta_f\Delta_f) \to \ldots \to^\infty f^\omega$$



STRONG CONVERGENCE

$$\Delta_f := \lambda x.f(xx) \qquad\qquad \Delta_f\Delta_f\text{: "Curry"}$$

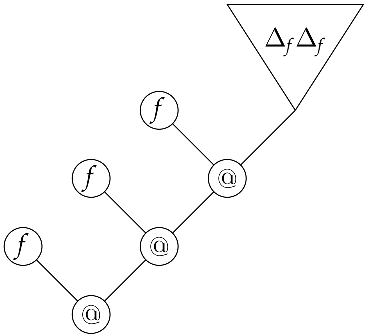
$$\Delta_f\Delta_f \to f(\Delta_f\Delta_f) \to f^2(\Delta_f\Delta_f) \to f^3(\Delta_f\Delta_f) \to f^4(\Delta_f\Delta_f) \to \ldots \to^\infty f^\omega$$



Strong Convergence

$$\Delta_f := \lambda x.f(xx) \qquad \Delta_f\Delta_f\text{: "Curry"}$$

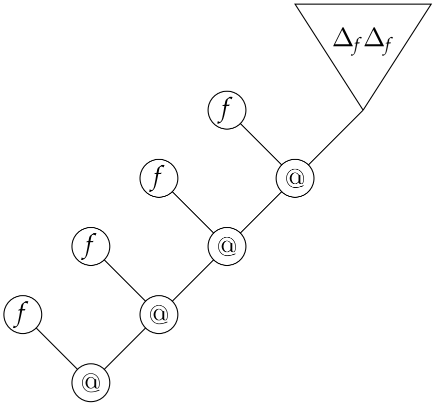
$$\Delta_f\Delta_f \to f(\Delta_f\Delta_f) \to f^2(\Delta_f\Delta_f) \to f^3(\Delta_f\Delta_f) \to f^4(\Delta_f\Delta_f) \to \ldots \to^\infty f^\omega$$



STRONG CONVERGENCE

$\Delta_f := \lambda x.f(xx)$ $\Delta_f\Delta_f$: "Curry"

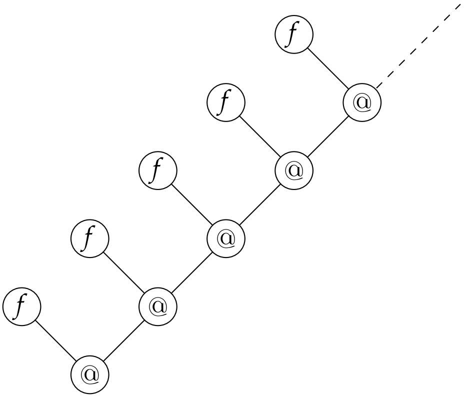
$$\Delta_f\Delta_f \to f(\Delta_f\Delta_f) \to f^2(\Delta_f\Delta_f) \to f^3(\Delta_f\Delta_f) \to f^4(\Delta_f\Delta_f) \to \ldots \to^\infty f^\omega$$



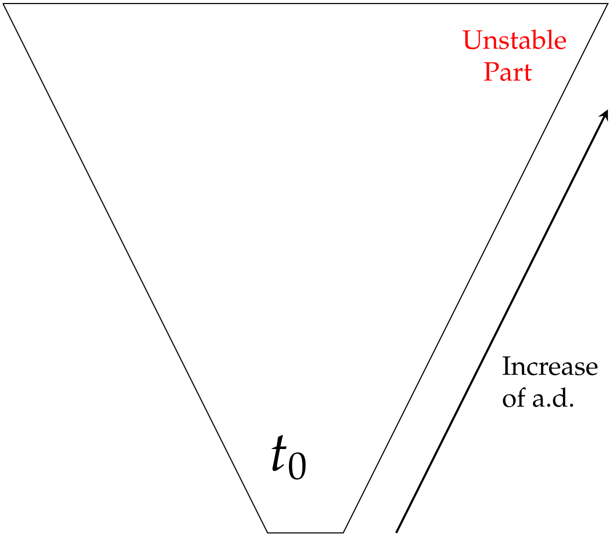
STRONG CONVERGENCE

$\Delta_f := \lambda x.f(xx)$ $\Delta_f\Delta_f$: "Curry"

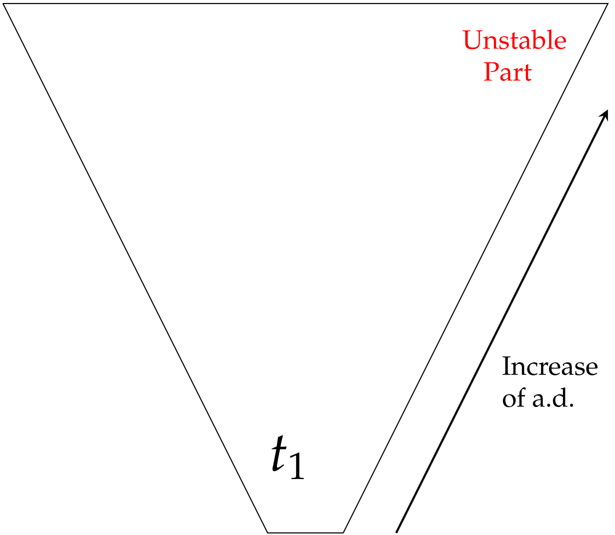
$$\Delta_f\Delta_f \to f(\Delta_f\Delta_f) \to f^2(\Delta_f\Delta_f) \to f^3(\Delta_f\Delta_f) \to f^4(\Delta_f\Delta_f) \to \ldots \to^\infty f^\omega$$



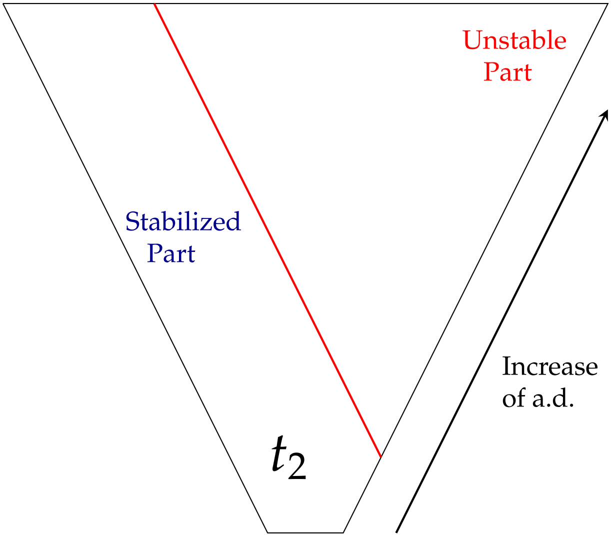
STRONG CONVERGENCE



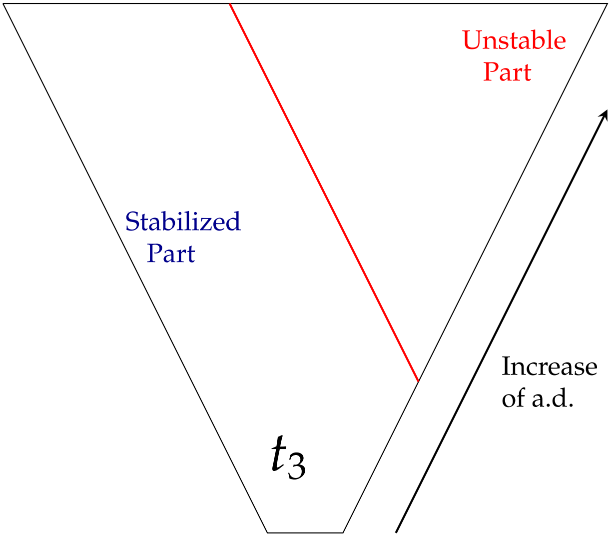
Strong Convergence



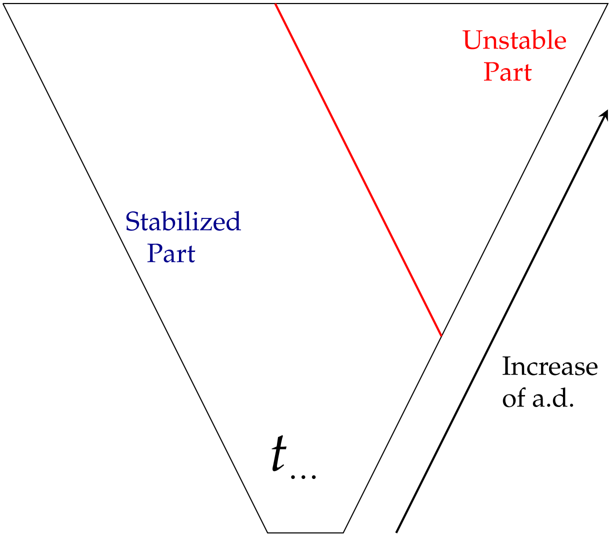
STRONG CONVERGENCE



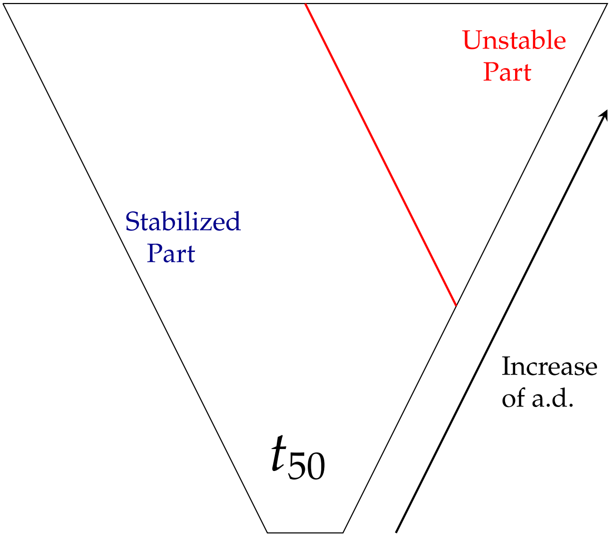
STRONG CONVERGENCE



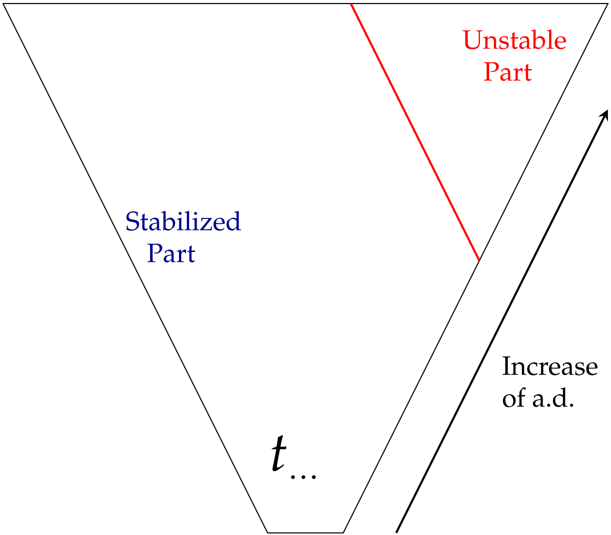
STRONG CONVERGENCE



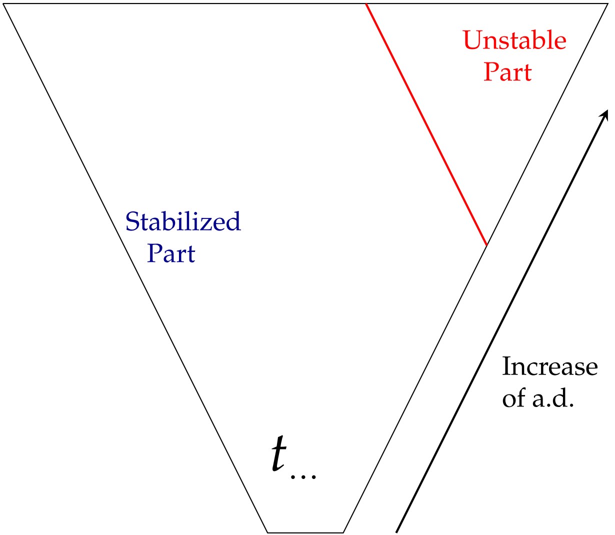
STRONG CONVERGENCE



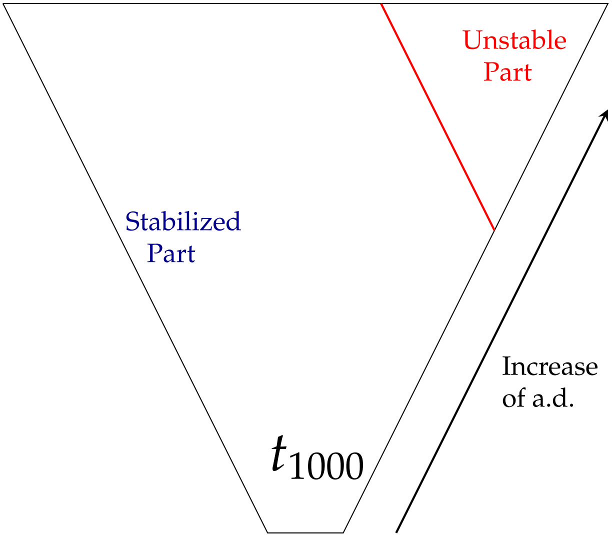
STRONG CONVERGENCE



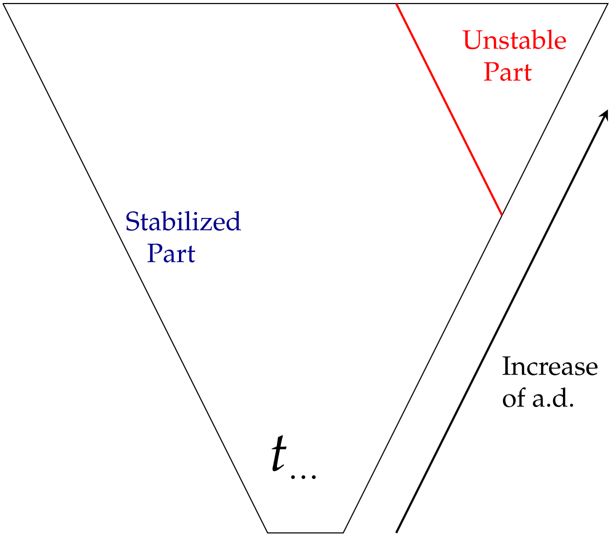
STRONG CONVERGENCE



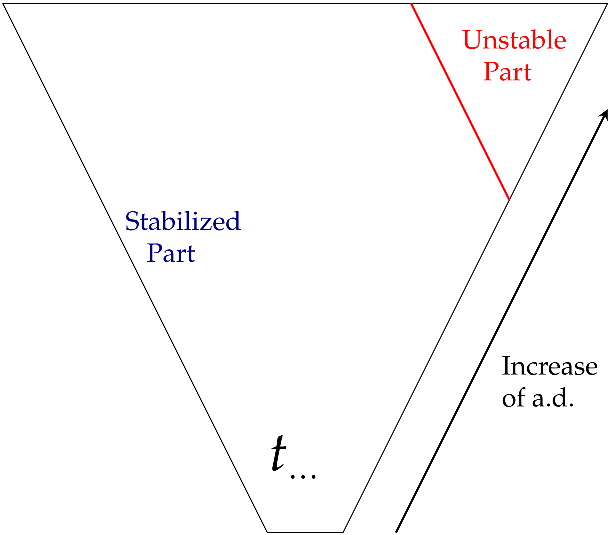
STRONG CONVERGENCE



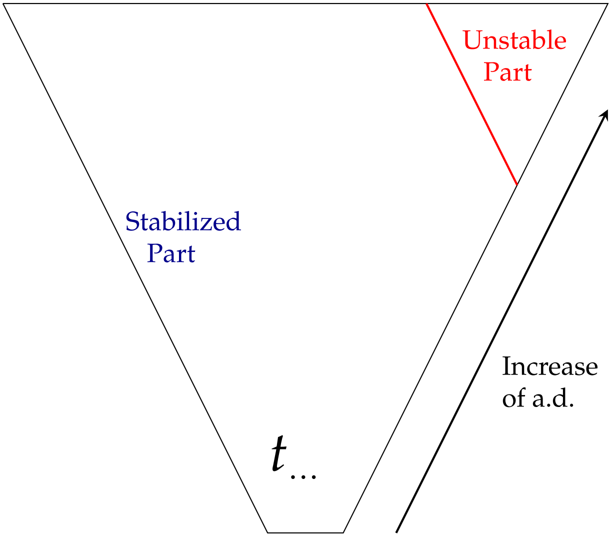
STRONG CONVERGENCE



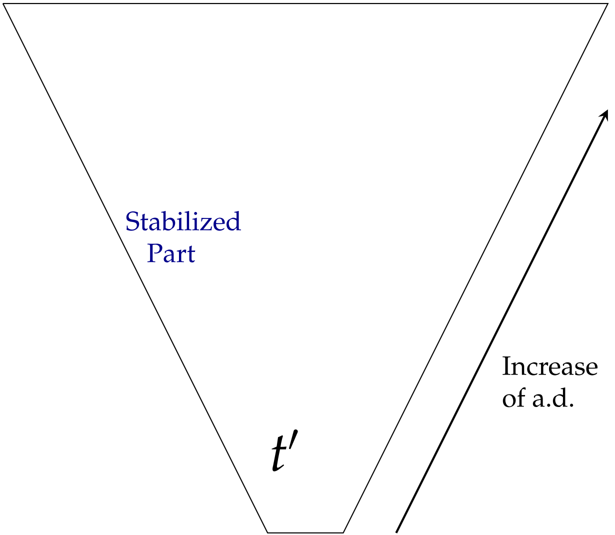
Strong Convergence



STRONG CONVERGENCE



Strong Convergence



STRONG CONVERGENCE

Conclusion

STRONG CONVERGENCE

Conclusion

A **strongly converging reduction sequence (s.c.r.s)** allows us to define its **limit**.

Infinitary Normalization

- The notions of redex and head-normalizability do not change.

INFINITARY NORMALIZATION

- The notions of redex and head-normalizability do not change.
- The NF of Λ^{001} are generated by the *coinductive* grammar:

$$t = \lambda x_1 \ldots \lambda x_p . x\, t_1 \ldots t_q \qquad (p, q \geqslant 0)$$

INFINITARY NORMALIZATION

- The notions of redex and head-normalizability do not change.
- The NF of Λ^{001} are generated by the *coinductive* grammar:

$$t = \lambda x_1 \dots \lambda x_p . x\, t_1 \dots t_q \qquad (p, q \geqslant 0)$$

Definition (Infinitary WN)

A 001-term is WN if it can be reduced to a NF through at least one s.c.r.s.

INFINITARY NORMALIZATION

- The notions of redex and head-normalizability do not change.
- The NF of Λ^{001} are generated by the *coinductive* grammar:

$$t = \lambda x_1 \ldots \lambda x_p . x\, t_1 \ldots t_q \qquad (p,\, q \geqslant 0)$$

Definition (Infinitary WN)

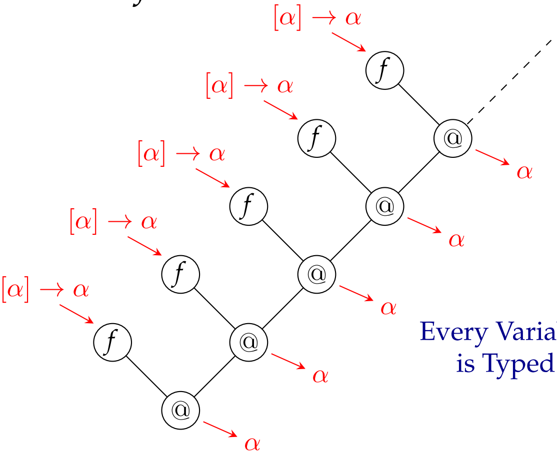
A 001-term is WN if it can be reduced to a NF through at least one s.c.r.s.

- Thus, a (finite) term is HHN iff it is 001-WN.

PLAN

TRUNCATION (FIGURES)

$$\Pi' \triangleright \Gamma \vdash f^\omega : \alpha$$

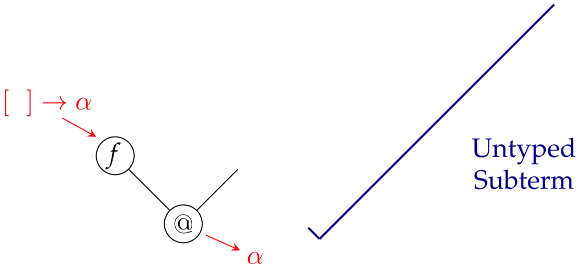


Every Variable
is Typed

$$\Gamma = f : [[\alpha] \to \alpha]_\omega \text{ (infinite multiplicity)}$$

TRUNCATION (FIGURES)

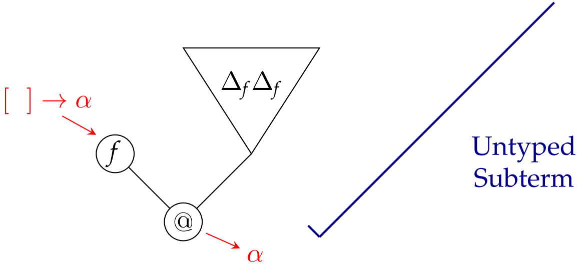
We can use the same derivation frame Π_1^* to type $f(\ldots)$



$\Gamma_1 = f : [[\ ] \to \alpha]$

Truncation (Figures)

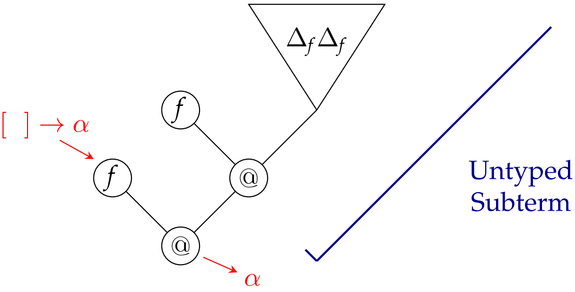
$$\Pi^1_1 \quad \rhd \Gamma_1 \vdash f(\Delta_f \Delta_f) : \alpha$$



$$\Gamma_1 = f : [[\ \ ] \to \alpha]$$

TRUNCATION (FIGURES)

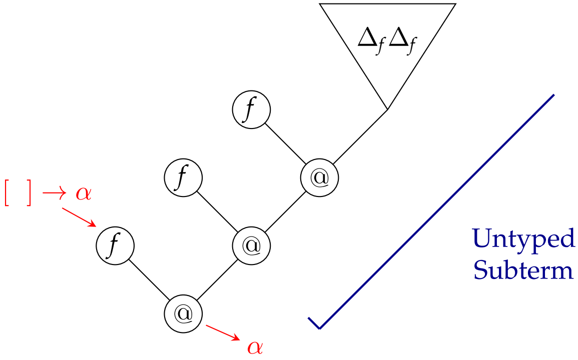
$$\Pi_1^2 \;\; \rhd \Gamma_1 \vdash f(f(\Delta_f \Delta_f)) : \alpha$$



$\Gamma_1 = f : [[\ ] \to \alpha]$

Truncation (Figures)

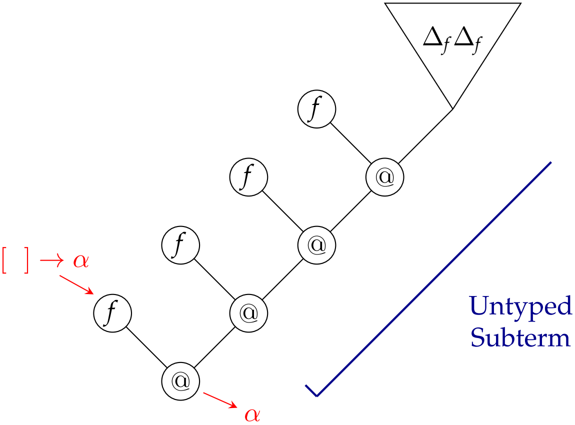
$$\Pi_1^3 \;\; \rhd \Gamma_1 \vdash f^3(\Delta_f \Delta_f) : \alpha$$



$$\Gamma_1 = f : [[\ \ ] \to \alpha]$$

Truncation (Figures)

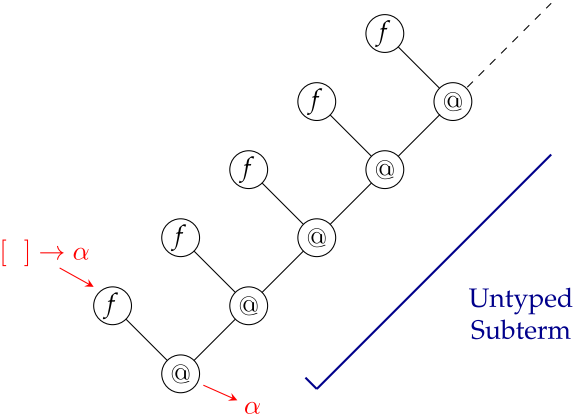
$$\Pi^4_1 \;\; \rhd \Gamma_1 \vdash f^4(\Delta_f \Delta_f) : \alpha$$



$$\Gamma_1 = f : [[\;\;] \to \alpha]$$

Truncation (Figures)

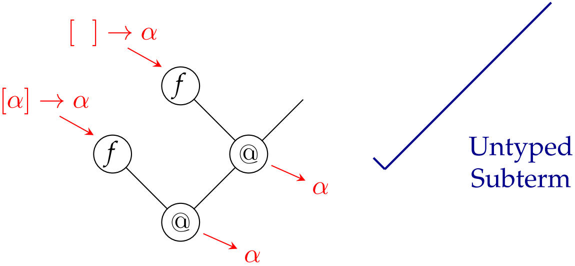
$$\Pi_1' \rhd \Gamma_1 \vdash f^\omega : \alpha$$



$$\Gamma_1 = f : [[\ ] \to \alpha]$$

TRUNCATION (FIGURES)

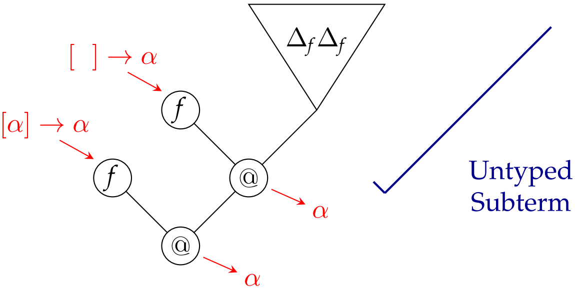
We can use the same derivation frame Π_2^* to type $f(f(\ldots))$



$\Gamma_2 = f : [[\alpha] \to \alpha] + [[\ ] \to \alpha]$

TRUNCATION (FIGURES)

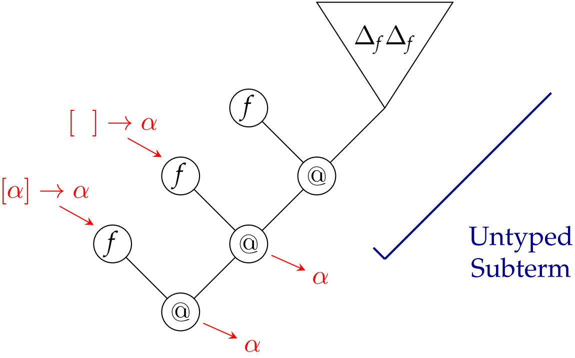
$$\Pi_2^2 \quad \triangleright \Gamma_2 \vdash f(f(\Delta_f \Delta_f)) : \alpha$$



$$\Gamma_2 = f : [[\alpha] \to \alpha] + [[\;] \to \alpha]$$

TRUNCATION (FIGURES)

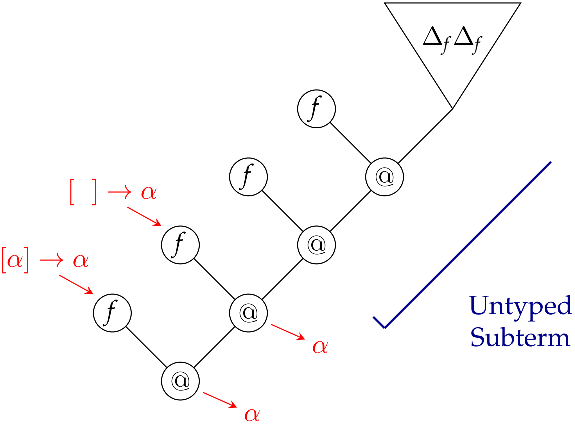
$$\Pi_2^3 \quad \triangleright \Gamma_2 \vdash f^3(\Delta_f \Delta_f) : \alpha$$



$$\Gamma_2 = f : [[\alpha] \to \alpha] + [[\ ] \to \alpha]$$

TRUNCATION (FIGURES)

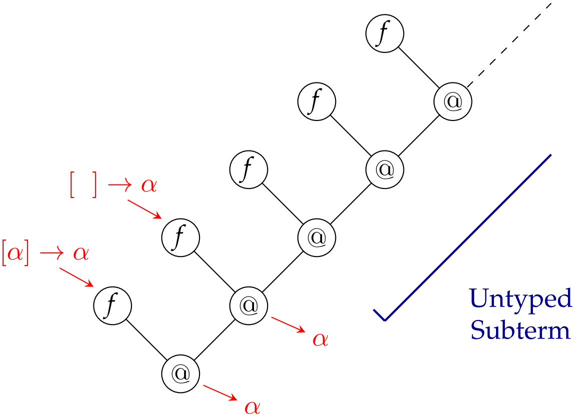
$$\Pi_2^4 \;\; \triangleright \Gamma_2 \vdash f^4(\Delta_f \Delta_f) : \alpha$$



$$\Gamma_2 = f : [[\alpha] \to \alpha] + [[\;\;] \to \alpha]$$

TRUNCATION (FIGURES)

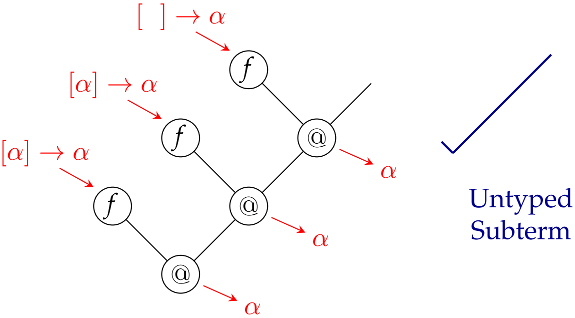
$$\Pi'_2 \rhd \Gamma_2 \vdash f^\omega : \alpha$$



$$\Gamma_2 = f : [[\alpha] \to \alpha] + [[\;\;] \to \alpha]$$

TRUNCATION (FIGURES)

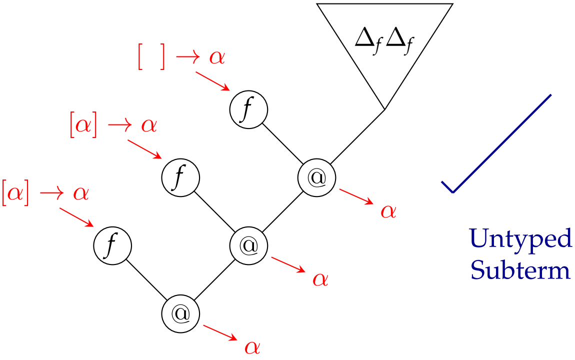
We can use the same derivation frame Π_3^* to type $f^3(\ldots)$



$$\Gamma_3 = f : [[\alpha] \to \alpha]_2 + [[\ \ ] \to \alpha]$$

TRUNCATION (FIGURES)

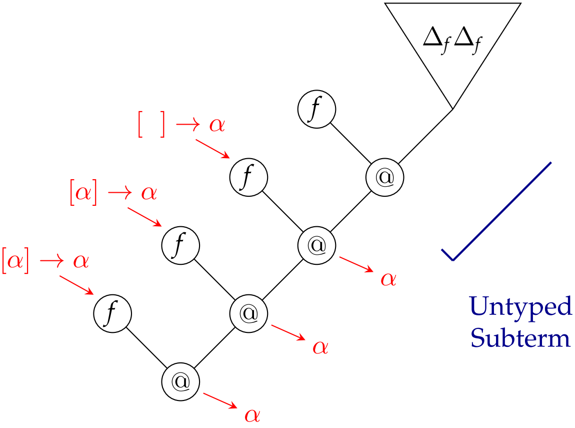
$$\Pi_3^3 \;\; \triangleright \Gamma_3 \vdash f^3(\Delta_f \Delta_f) : \alpha$$



$$\Gamma_3 = f : [[\alpha] \to \alpha]_2 + [[\;\;] \to \alpha]$$

TRUNCATION (FIGURES)

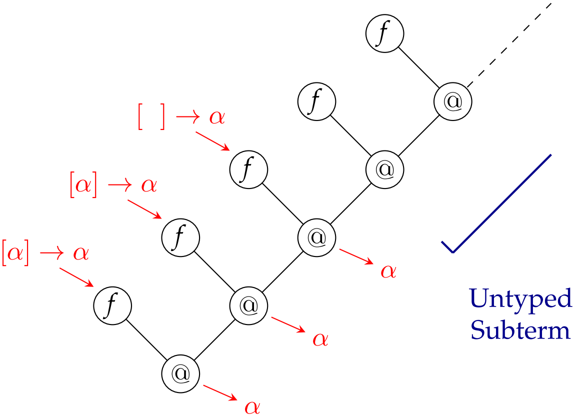
$$\Pi_3^4 \quad \triangleright \Gamma_3 \vdash f^4(\Delta_f \Delta_f) : \alpha$$



$$\Gamma_3 = f : [[\alpha] \to \alpha]_2 + [[\ ] \to \alpha]$$

TRUNCATION (FIGURES)

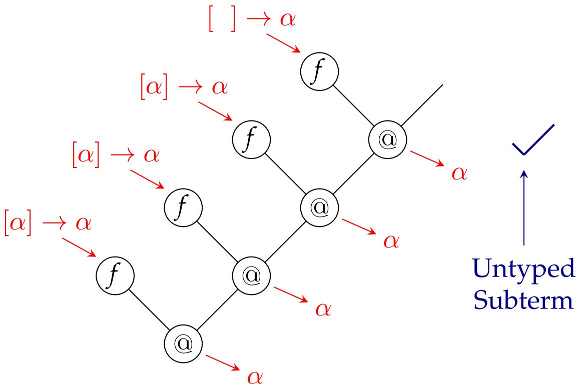
$$\Pi_3' \rhd \Gamma_3 \vdash f^\omega : \alpha$$



$$\Gamma_3 = f : [[\alpha] \to \alpha]_2 + [[\ ] \to \alpha]$$

TRUNCATION (FIGURES)

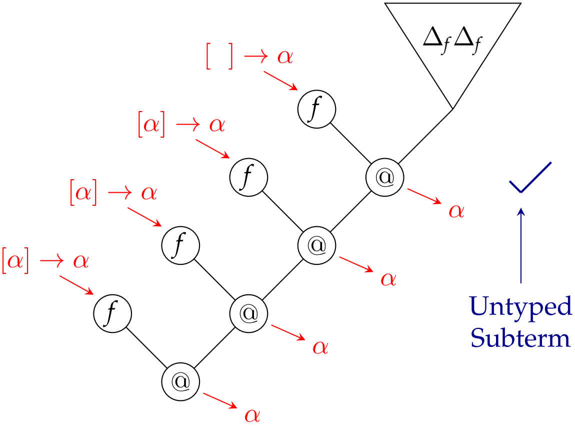
We can use the same derivation frame Π_4^* to type $f^4(\ldots)$



$$\Gamma_4 = f : [[\alpha] \to \alpha]_3 + [[\ ] \to \alpha]$$

TRUNCATION (FIGURES)

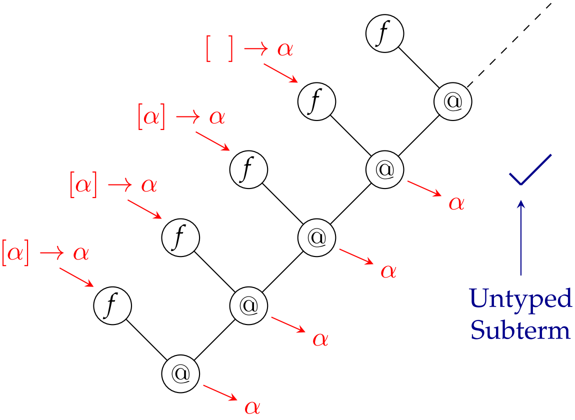
$$\Pi^4_4 \ \triangleright \Gamma_4 \vdash f^4(\Delta_f \Delta_f) : \alpha$$



$$\Gamma_4 = f : [[\alpha] \to \alpha]_3 + [[\ ] \to \alpha]$$

Truncation (Figures)

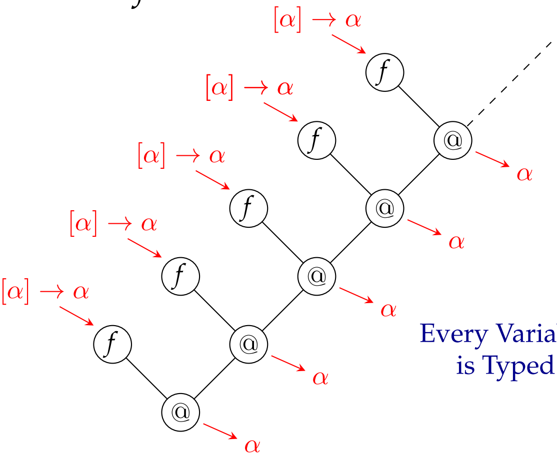
$$\Pi'_4 \rhd \Gamma_4 \vdash f^{\omega} : \alpha$$



$$\Gamma_4 = f : [[\alpha] \to \alpha]_3 + [[\ ] \to \alpha]$$

TRUNCATION (FIGURES)

$$\Pi' \triangleright \Gamma \vdash f^{\omega} : \alpha$$

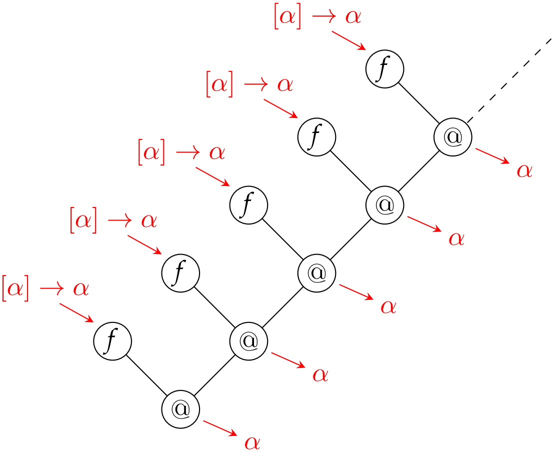


Every Variable
is Typed

$$\Gamma = f : [[\alpha] \to \alpha]_{\omega} \text{ (infinite multiplicity)}$$

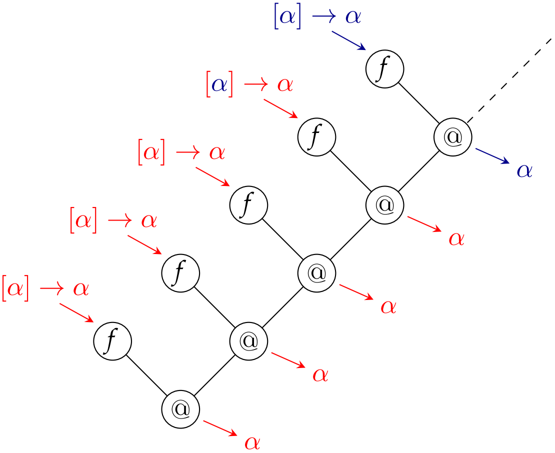
TRUNCATION (FIGURES)

Π' can be **truncated** into Π'_4:



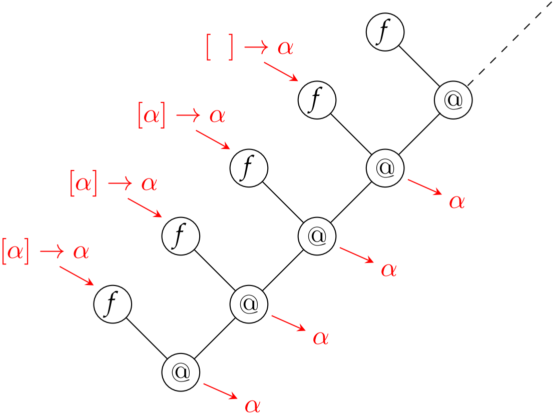
TRUNCATION (FIGURES)

Π' can be **truncated** into Π'_4:



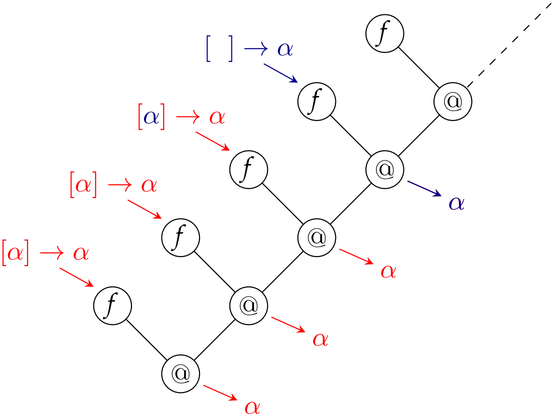
TRUNCATION (FIGURES)

Π' can be **truncated** into Π'_4:



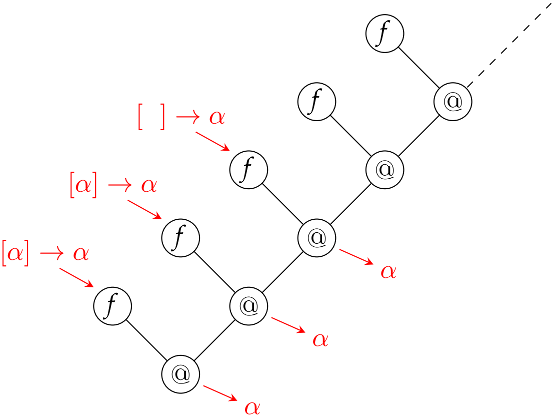
TRUNCATION (FIGURES)

Π' can be **truncated** into Π'_3:



Truncation (Figures)

Π' can be **truncated** into Π'_3:



INFINITARY SUBJECT EXPANSION

- How do we perform ∞-subject expansion on Π' (typing f^ω)?

INFINITARY SUBJECT EXPANSION

- How do we perform ∞-subject expansion on Π' (typing f^ω)?
  - Π', that types f^ω, cannot be expanded (yet).

INFINITARY SUBJECT EXPANSION

- How do we perform ∞-subject expansion on Π' (typing f^ω)?
  - Π', that types f^ω, cannot be expanded (yet).
  - Π'_n, that also types f^ω, cannot be expanded (yet).

INFINITARY SUBJECT EXPANSION

- How do we perform ∞-subject expansion on Π' (typing f^ω)?
  - Π', that types f^ω, cannot be expanded (yet).
  - Π'_n, that also types f^ω, cannot be expanded (yet).
  - But Π^k_n, that types $f^k(\Delta_f \Delta_f)$, can be expanded.

INFINITARY SUBJECT EXPANSION

- How do we perform ∞-subject expansion on Π' (typing f^ω)?
  - Π', that types f^ω, cannot be expanded (yet).
  - Π'_n, that also types f^ω, cannot be expanded (yet).
  - But Π^k_n, that types $f^k(\Delta_f \Delta_f)$, can be expanded.
  - Π^k_n yields a derivation Π_n typing $\Delta_f \Delta_f$ (after k exp-steps).

INFINITARY SUBJECT EXPANSION

- How do we perform ∞-subject expansion on Π' (typing f^ω)?
  - Π', that types f^ω, cannot be expanded (yet).
  - Π'_n, that also types f^ω, cannot be expanded (yet).
  - But Π^k_n, that types $f^k(\Delta_f\Delta_f)$, can be expanded.
  - Π^k_n yields a derivation Π_n typing $\Delta_f\Delta_f$ (after k exp-steps).
  - We can build a **"join"** of the Π_n, thus producing an infinite unforgetful derivation Π typing $\Delta_f\Delta_f$.

Infinitary Subject Expansion

- How do we perform ∞-subject expansion on Π' (typing f^ω)?
  - Π', that types f^ω, cannot be expanded (yet).
  - Π'_n, that also types f^ω, cannot be expanded (yet).
  - But Π^k_n, that types $f^k(\Delta_f \Delta_f)$, can be expanded.
  - Π^k_n yields a derivation Π_n typing $\Delta_f \Delta_f$ (after k exp-steps).
  - We can build a **"join"** of the Π_n, thus producing an infinite unforgetful derivation Π typing $\Delta_f \Delta_f$.
- Derivation Π features a type γ coinductively defined by the fixpoint equation $\gamma = [\gamma]_\omega \to \alpha$.

INFINITARY SUBJECT EXPANSION

- How do we perform ∞-subject expansion on Π' (typing f^ω)?
  - Π', that types f^ω, cannot be expanded (yet).
  - Π'_n, that also types f^ω, cannot be expanded (yet).
  - But Π^k_n, that types $f^k(\Delta_f \Delta_f)$, can be expanded.
  - Π^k_n yields a derivation Π_n typing $\Delta_f \Delta_f$ (after k exp-steps).
  - We can build a **"join"** of the Π_n, thus producing an infinite unforgetful derivation Π typing $\Delta_f \Delta_f$.
- Derivation Π features a type γ coinductively defined by the fixpoint equation $\gamma = [\gamma]_\omega \to \alpha$.
- Type γ allows to type $\Delta\Delta$. Need for a **validity criterion**.

APPROXIMABILITY (HEURISTIC)

- Informally, see a derivation Π as a set of symbols (type variables α or $\rightarrow$ that we found inside each jugdment of P).

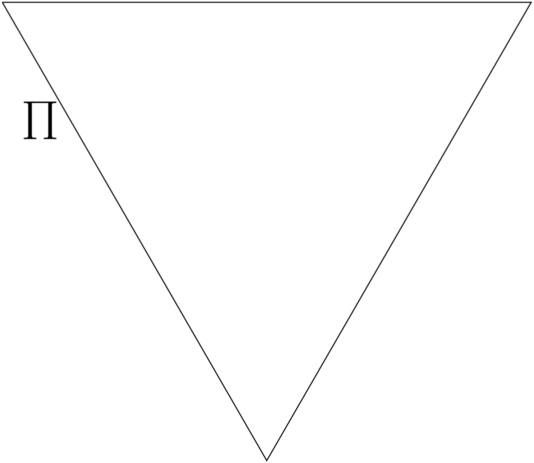
APPROXIMABILITY (HEURISTIC)

- Informally, see a derivation Π as a set of symbols (type variables α or $\rightarrow$ that we found inside each jugdment of P).
- A **(finite) approximation** ${}^{\mathrm{f}}\Pi$ of a derivation Π is a finite subset of symbols of Π which is itself a derivation. We write ${}^{\mathrm{f}}\Pi \leqslant \Pi$.

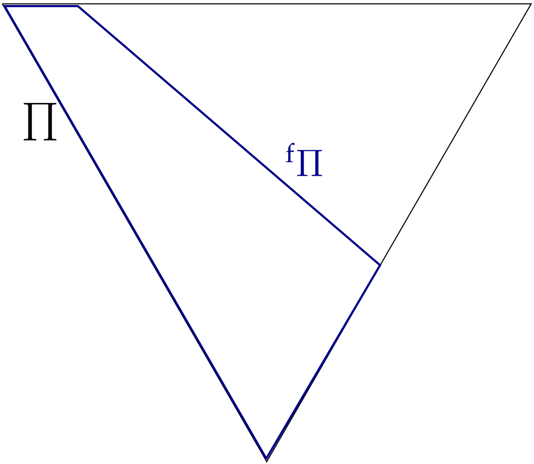
APPROXIMABILITY (HEURISTIC)

- Informally, see a derivation Π as a set of symbols (type variables α or $\rightarrow$ that we found inside each jugdment of P).
- A **(finite) approximation** ${}^{f}\Pi$ of a derivation Π is a finite subset of symbols of Π which is itself a derivation. We write ${}^{f}\Pi \leqslant \Pi$.
- A derivation Π is said to be **approximable** if for all finite subset B of symbols of Π, there is an approximation ${}^{f}\Pi \leqslant \Pi$ that contains B.

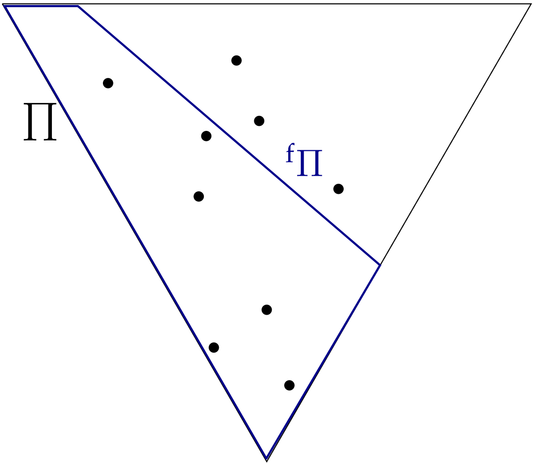
APPROXIMABILITY (FIGURE)



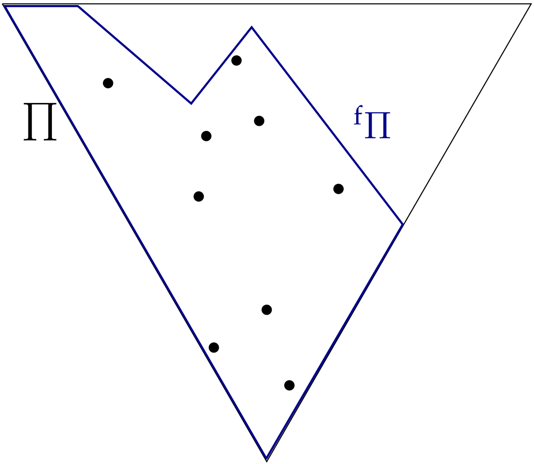
APPROXIMABILITY (FIGURE)



APPROXIMABILITY (FIGURE)

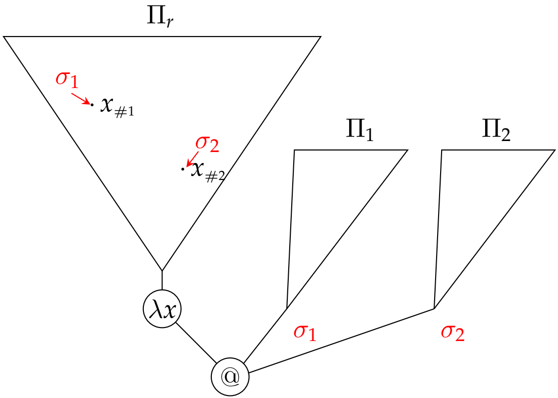


APPROXIMABILITY (FIGURE)



NON-DETERMINISM AND TRUNCATION

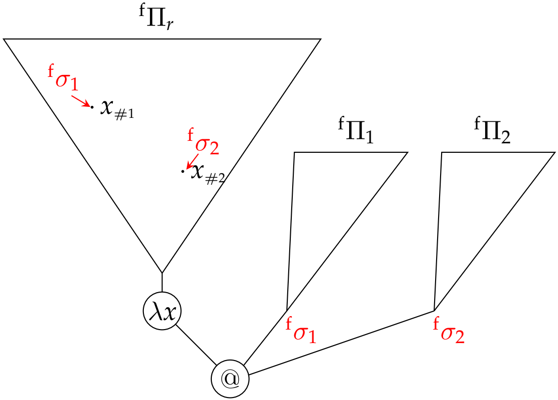
$(\lambda x.r)s$



NON-DETERMINISM AND TRUNCATION

$(\lambda x.r)s$

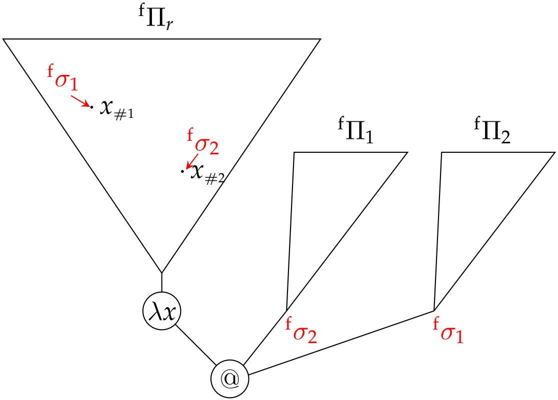
Truncation possibly affects every type nested inside Π.



NON-DETERMINISM AND TRUNCATION

$(\lambda x.r)s$

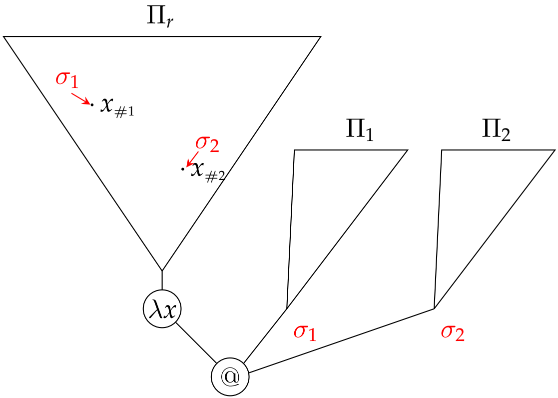
Truncation possibly affects every type nested inside Π.



NON-DETERMINISM AND TRUNCATION

$(\lambda x.r)s$

Assume $\sigma_1 = \sigma_2$.

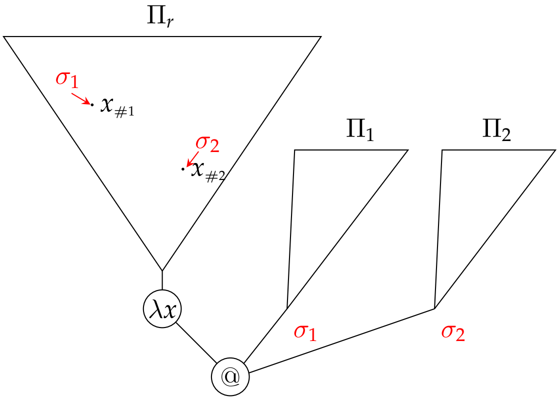


NON-DETERMINISM AND TRUNCATION

$(\lambda x.r)s$

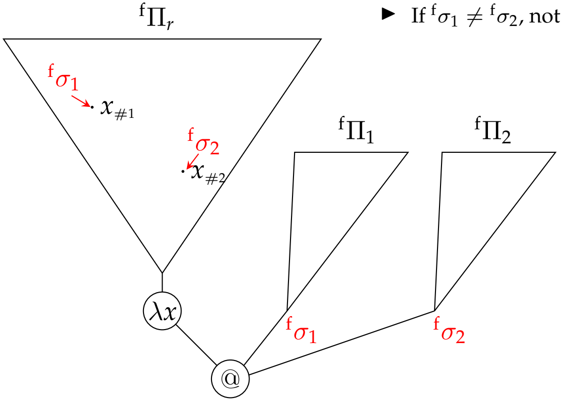
Assume $\sigma_1 = \sigma_2$.

- Possible in Π: $\#1 \mapsto \Pi_2, \#2 \mapsto \Pi_1$



NON-DETERMINISM AND TRUNCATION

$(\lambda x.r)s$



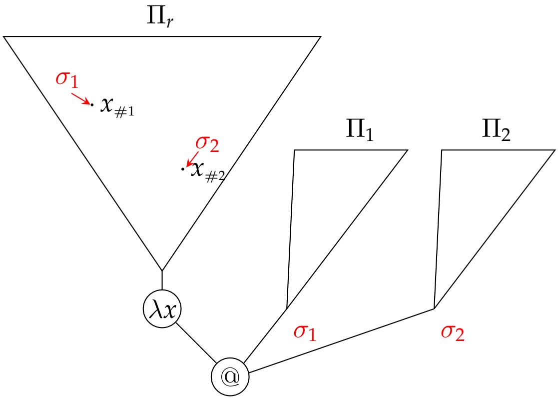
Assume $\sigma_1 = \sigma_2$.

- Possible in Π:
  $\#1 \mapsto \Pi_2, \#2 \mapsto \Pi_1$
- If ${}^{\mathrm{f}}\sigma_1 \neq {}^{\mathrm{f}}\sigma_2$, not in ${}^{\mathrm{f}}P$.

NON-DETERMINISM AND TRUNCATION

$(\lambda x.r)s$

Assume $\sigma_1 \neq \sigma_2$

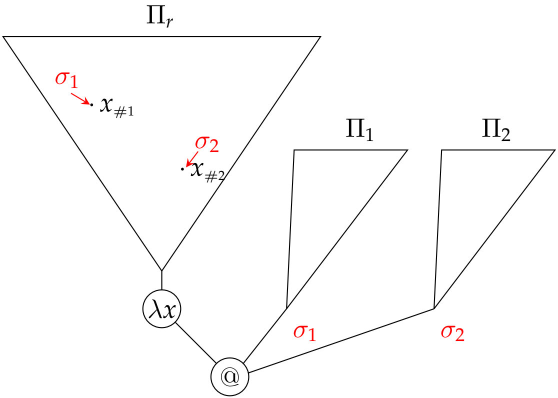


Non-Determinism and Truncation

$$(\lambda x.r)s$$

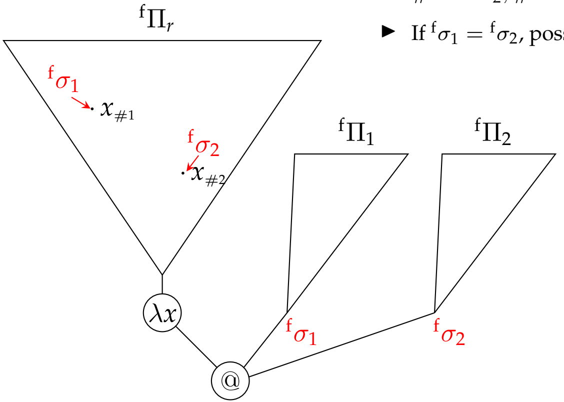
Assume $\sigma_1 \neq \sigma_2$

- Not possible in Π: $\#1 \mapsto \Pi_2, \#2 \mapsto \Pi_1$



NON-DETERMINISM AND TRUNCATION

$(\lambda x.r)s$



Assume $\sigma_1 \neq \sigma_2$

- Not possible in Π: $\#1 \mapsto \Pi_2, \#2 \mapsto \Pi_1$
- If ${}^{\mathrm{f}}\sigma_1 = {}^{\mathrm{f}}\sigma_2$, possible in ${}^{\mathrm{f}}P$.

PLAN

TYPES OF S

TYPES OF S

- **Type (metavariable S, T)**: coinductive grammar

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

TYPES OF S

- **Type (metavariable S, T)**: coinductive grammar

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

- **Sequence Type**:
  - Intersection type replacing multiset types.

TYPES OF S

- **Type (metavariable S, T)**: coinductive grammar

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

- **Sequence Type**:
  - Intersection type replacing multiset types.
  - $F = (T_k)_{k \in K}$ where T_k types and $K \subset \mathbb{N} - \{0, 1\}$.

TYPES OF S

- **Type (metavariable S, T)**: coinductive grammar

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

- **Sequence Type**:
  - Intersection type replacing multiset types.
  - $F = (T_k)_{k \in K}$ where T_k types and $K \subset \mathbb{N} - \{0, 1\}$.
- *Example (Sequence Type):* $(T_k)_{k=2,3,8}$ with $T_2 = T_8 = S$ and $T_3 = S'$.

TYPES OF S

- **Type (metavariable S, T)**: coinductive grammar

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

- **Sequence Type**:
  - Intersection type replacing multiset types.
  - $F = (T_k)_{k \in K}$ where T_k types and $K \subset \mathbb{N} - \{0, 1\}$.
- *Example (Sequence Type):* $(T_k)_{k=2,3,8}$ with $T_2 = T_8 = S$ and $T_3 = S'$.

$$F = \begin{array}{ccc} S & S' & S \\ | & | & | \\ 8 & 3 & 2 \end{array} \longleftarrow \text{argument tracks } (2,3,8)$$

TYPES OF S

- **Type (metavariable S, T)**: coinductive grammar

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

- **Sequence Type**:
  - Intersection type replacing multiset types.
  - $F = (T_k)_{k \in K}$ where T_k types and $K \subset \mathbb{N} - \{0, 1\}$.
- *Example (Sequence Type):* $(T_k)_{k=2,3,8}$ with $T_2 = T_8 = S$ and $T_3 = S'$.

$$F = \begin{array}{ccc} S & S' & S \\ 8 & 3 & 2 \end{array} \longleftarrow \text{argument tracks } (2,3,8)$$

- *Example (Arrow Type):* $\ldots \to T$.

TYPES OF S

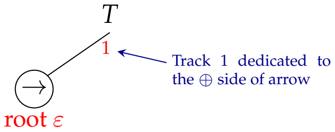
- **Type (metavariable S, T)**: coinductive grammar

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

- **Sequence Type**:
  - Intersection type replacing multiset types.
  - $F = (T_k)_{k\in K}$ where T_k types and $K \subset \mathbb{N} - \{0,1\}$.
- *Example (Sequence Type):* $(T_k)_{k=2,3,8}$ with $T_2 = T_8 = S$ and $T_3 = S'$.

$$F = \begin{array}{ccc} S & S' & S \\ | & | & | \\ 8 & 3 & 2 \end{array} \longleftarrow \text{argument tracks } (2,3,8)$$

- *Example (Arrow Type):* $\ldots \to T$.



TYPES OF S

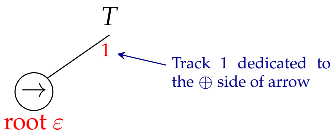
- **Type (metavariable S, T)**: coinductive grammar

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

- **Sequence Type**:
  - Intersection type replacing multiset types.
  - $F = (T_k)_{k \in K}$ where T_k types and $K \subset \mathbb{N} - \{0, 1\}$.
- *Example (Sequence Type):* $(T_k)_{k=2,3,8}$ with $T_2 = T_8 = S$ and $T_3 = S'$.

$$F = \begin{array}{ccc} S & S' & S \\ | & | & | \\ 8 & 3 & 2 \end{array} \longleftarrow \text{argument tracks } (2, 3, 8)$$

- *Example (Arrow Type):* $F \to T$.



TYPES OF S

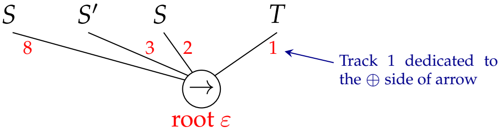
- **Type (metavariable S, T)**: coinductive grammar

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

- **Sequence Type**:
  - Intersection type replacing multiset types.
  - $F = (T_k)_{k \in K}$ where T_k types and $K \subset \mathbb{N} - \{0, 1\}$.
- *Example (Sequence Type):* $(T_k)_{k=2,3,8}$ with $T_2 = T_8 = S$ and $T_3 = S'$.

$$F = \begin{array}{ccc} S & S' & S \\ 8 & 3 & 2 \end{array} \longleftarrow \text{argument tracks } (2,3,8)$$

- *Example (Arrow Type):* $F \to T$.



DERIVATIONS OF S

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

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

$$\frac{C \vdash t : (S_k)_{k \in K} \to T \qquad (D_k \vdash u : S_k)_{k \in K}}{C \cup \bigcup_{k \in K} D_k \vdash t\,u : T}\ \text{app}$$

DERIVATIONS OF S

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

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

$$\frac{C \vdash t : (S_k)_{k \in K} \to T \qquad (D_k \vdash u : S'_k)_{k \in K'} \qquad (S_k)_{k \in K} = (S'_k)_{k \in K'}}{C \cup \bigcup_{k \in K} D_k \vdash t\, u : T} \text{ app}$$

- For the `app`-rule: **syntactic** equality.

DERIVATIONS OF S

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

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

$$\frac{C \vdash t : (S_k)_{k \in K} \to T \qquad (D_k \vdash u : S_k)_{k \in K}}{C \cup \bigcup_{k \in K} D_k \vdash t\,u : T}\ \text{app}$$

- For the `app`-rule: **syntactic** equality.
- **Warning !**
  If $\mathrm{Rt}(C)$ and the $\mathrm{Rt}(D_k)$ are not pairwise disjoint, contexts are incompatible.

DERIVATIONS OF S

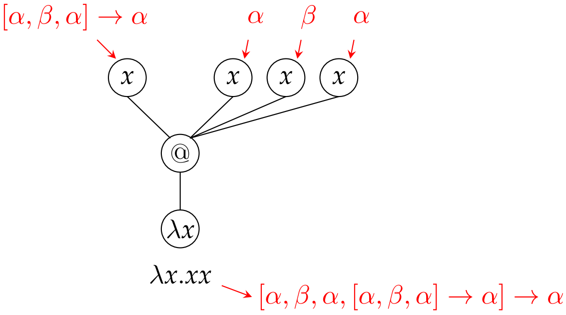
The set `Deriv` of rigid derivations is *coinductively* generated by:

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

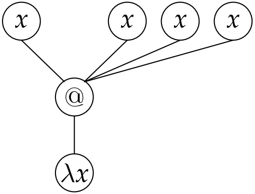
$$\dfrac{C \vdash t : (S_k)_{k \in K} \to T \qquad (D_k \vdash u : S_k)_{k \in K}}{C \cup \bigcup_{k \in K} D_k \vdash t\,u : T}\ \text{app}$$

- For the `app`-rule: **syntactic** equality.
- **Warning !**
  If $\mathrm{Rt}(C)$ and the $\mathrm{Rt}(D_k)$ are not pairwise disjoint, contexts are incompatible.
- Parsing: premise of `abs` on tr. 0, left premise of `app` on tr. 1.

RIGID DERIVATION (EXAMPLE)

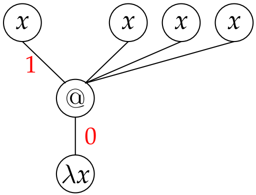


RIGID DERIVATION (EXAMPLE)



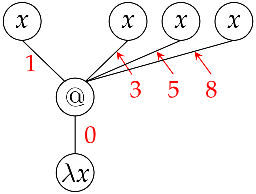
$\lambda x.xx$

Rigid Derivation (Example)



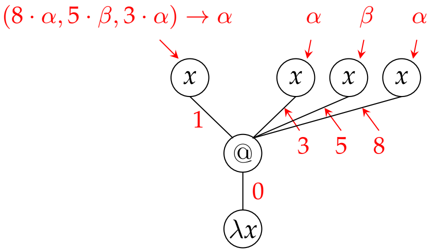
$\lambda x.xx$

RIGID DERIVATION (EXAMPLE)



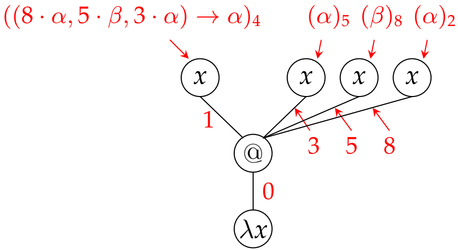
$\lambda x.xx$

RIGID DERIVATION (EXAMPLE)



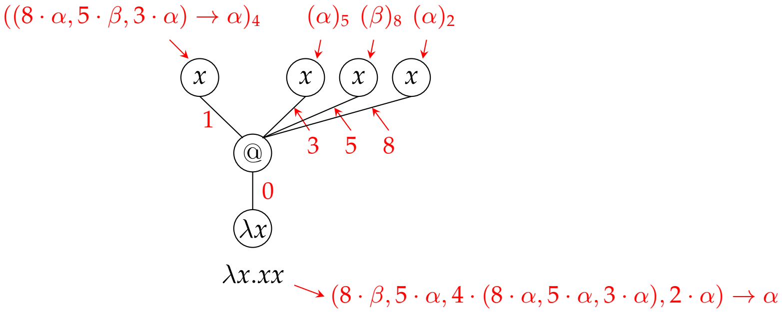
$\lambda x.xx$

RIGID DERIVATION (EXAMPLE)



$\lambda x.xx$

RIGID DERIVATION (EXAMPLE)



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 $S = S_5$

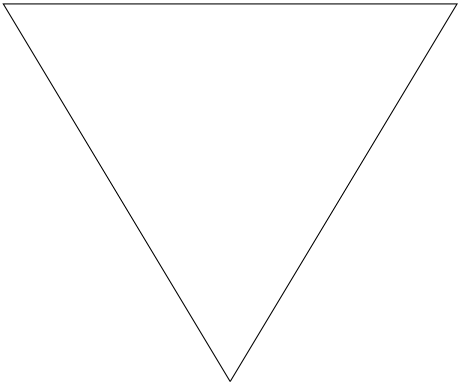
MAIN FEATURES

- 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 $S = S_5$

- **Trackability:** every symbol used inside P can be pointed at univocuously. Notion of *biposition*.

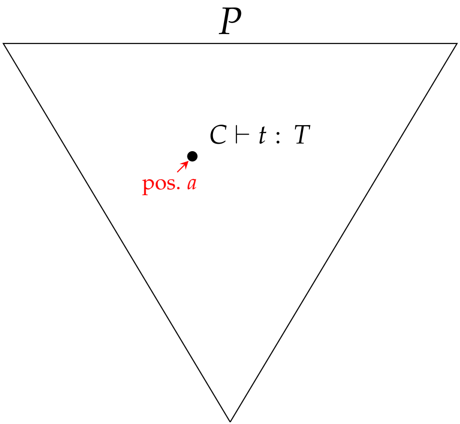
BISUPPORT OF A DERIVATION

P



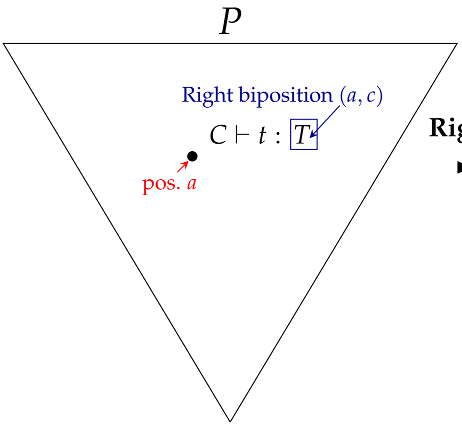
$A := \mathtt{supp}(P) \subset \mathbb{N}^*$

BISUPPORT OF A DERIVATION



$A := \mathtt{supp}(P) \subset \mathbb{N}^*$

BISUPPORT OF A DERIVATION

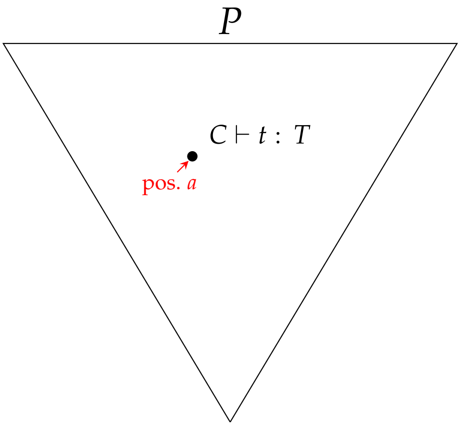


$A := \mathtt{supp}(P) \subset \mathbb{N}^*$

Right of $\vdash$

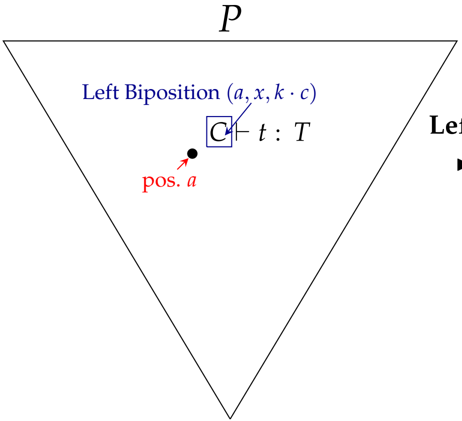
- $c \in \mathtt{supp}(T)$

BISUPPORT OF A DERIVATION



$A := \mathtt{supp}(P) \subset \mathbb{N}^*$

BISUPPORT OF A DERIVATION

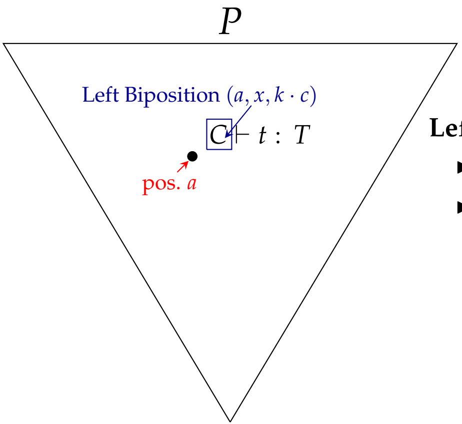


$A := \mathtt{supp}(P) \subset \mathbb{N}^*$

Left of $\vdash$

- x variable.

BISUPPORT OF A DERIVATION

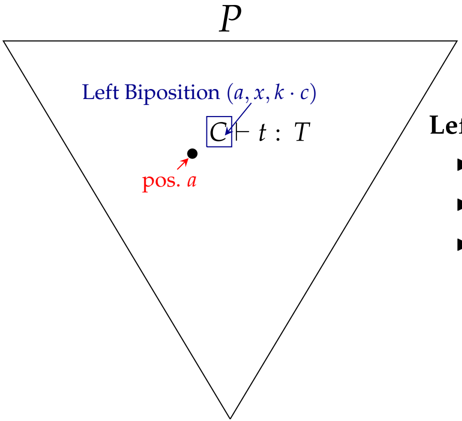


$A := \mathtt{supp}(P) \subset \mathbb{N}^*$

Left of $\vdash$

- x variable.
- $(T_k)_{k \in K} := C(x)$

BISUPPORT OF A DERIVATION

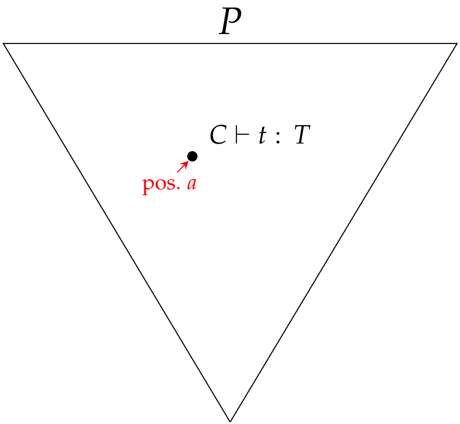


$A := \mathtt{supp}(P) \subset \mathbb{N}^*$

Left of $\vdash$

- x variable.
- $(T_k)_{k \in K} := C(x)$
- $k \in K$ and $c \in \mathtt{supp}(T_k)$.

BISUPPORT OF A DERIVATION



$A := \mathtt{supp}(P) \subset \mathbb{N}^*$

Bisupport of P: the set of (right or left) bipositions

APPROXIMABILITY

- Every symbol inside a rigid derivation P has a **biposition** (a position pointing inside a type nested in a judgment of P).

Approximability

- Every symbol inside a rigid derivation P has a **biposition** (a position pointing inside a type nested in a judgment of P).

- A **finite part** B of P is *finite* subset of $\texttt{bisupp}(P)$.

APPROXIMABILITY

- Every symbol inside a rigid derivation P has a **biposition** (a position pointing inside a type nested in a judgment of P).
- A **finite part** B of P is *finite* subset of $\texttt{bisupp}(P)$.
- A **finite approximation** of P is a (finite) derivation induced by P on a finite part of P.

APPROXIMABILITY

- Every symbol inside a rigid derivation P has a **biposition** (a position pointing inside a type nested in a judgment of P).
- A **finite part** B of P is *finite* subset of $\texttt{bisupp}(P)$.
- A **finite approximation** of P is a (finite) derivation induced by P on a finite part of P.
- A rigid derivation P is said to be **approximable** if for all finite part B of P, there is a finite approximation ${}^{\mathrm{f}}P \leqslant P$ s.t. ${}^{\mathrm{f}}P$ contains B.

CHARACTERIZATION OF INFINITARY WN

Theorem

A 001-term t is WN iff t is unforgetfully typable by means of an approximable derivation.

CHARACTERIZATION OF INFINITARY WN

Theorem

A 001-term t is WN iff t is unforgetfully typable by means of an approximable derivation.

Argument 1: If a term is typable by an approximable derivation, then it is head normalizing. Unforgetfulness makes HN hereditary.

CHARACTERIZATION OF INFINITARY WN

Theorem
A 001-term t is WN iff t is unforgetfully typable by means of an approximable derivation.

Argument 1: If a term is typable by an approximable derivation, then it is head normalizing. Unforgetfulness makes HN hereditary.
Argument 2: Subject reduction holds for s.c.r.s. (with or without approximability condition).

CHARACTERIZATION OF INFINITARY WN

Theorem
A 001-term t is WN iff t is unforgetfully typable by means of an approximable derivation.

Argument 1: If a term is typable by an approximable derivation, then it is head normalizing. Unforgetfulness makes HN hereditary.
Argument 2: Subject reduction holds for s.c.r.s. (with or without approximability condition).
Argument 3: Every NF can be typed by quantitative unforgetful derivations and every quantitative derivation typing a NF is approximable.

Characterization of infinitary WN

Theorem

A 001-term t is WN iff t is unforgetfully typable by means of an approximable derivation.

Argument 1: If a term is typable by an approximable derivation, then it is head normalizing. Unforgetfulness makes HN hereditary.
Argument 2: Subject reduction holds for s.c.r.s. (with or without approximability condition).
Argument 3: Every NF can be typed by quantitative unforgetful derivations and every quantitative derivation typing a NF is approximable.
Argument 4: Subject expansion property holds for s.c.r.s. (assuming approximability only).

PLAN

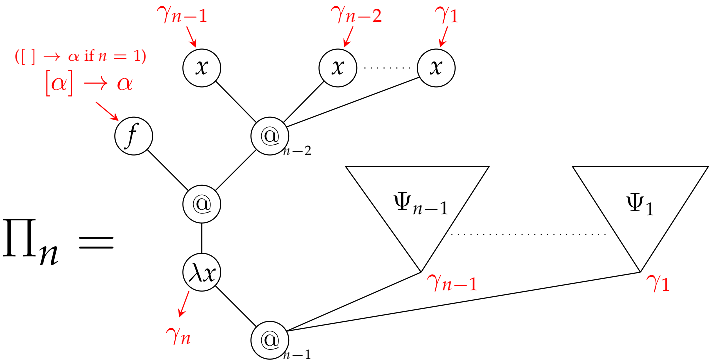
FUTURE WORK?

- Representation Theorem: every $\mathscr{M}$-derivation is the collapse of a $\mathtt{S}$-derivation (already done, HOR 2016).
- Can we reformulate approximability ?
- Can infinitary Strong Normalization be characterized ?
- Is every term typable in $\mathtt{S}$ (without approximability) ?
  Yes ! $\mathtt{S}$ is completely unsound (difficult because of relevance).
- *Categorical Adaptation* of this framework (ongoing work with D. Mazza and L. Pellisier).

QUESTIONS

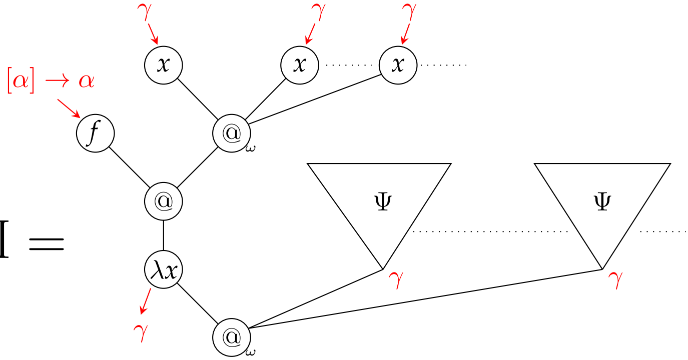
Thank you for your attention !

Expanded Derivations



- Ψ_n is the left subderivation.
- $\gamma_1 = [\,] \to \alpha$ and $\gamma_n = [\gamma_i]_{1 \leqslant i \leqslant n-1} \to \alpha$.
- Π_n is the n-expanded of Π_n^n, the $(n+1)$-expanded of Π_n^{n+1},..., the ∞-expanded of Π'_n.

Expanded Derivations

$\Pi =$ 

- Ψ is the left subderivation.
- $\gamma = [\gamma]_\omega \to \alpha$.
- Π_n is the n-expanded of Π_n^n, the $(n+1)$-expanded of Π_n^{n+1},..., the ∞-expanded of Π'_n.

SYSTEM $\mathscr{M}$

- The relation $\equiv$ (between types or seq. types) is defined coinductively:

SYSTEM $\mathscr{M}$

- The relation $\equiv$ (between types or seq. types) is defined coinductively:
  - $\alpha \equiv \alpha$.

SYSTEM $\mathscr{M}$

- 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 $T \equiv T'$.

SYSTEM $\mathscr{M}$

- 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. $\forall k \in K,\ S_k \equiv S'_{\sigma(k)}$.

SYSTEM $\mathscr{M}$

- 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. $\forall k \in K,\ S_k \equiv S'_{\sigma(k)}$.
- We set $\mathtt{Types}_{\mathscr{M}} = \mathtt{Types} \,/ \equiv$.
  The set of multiset types is $\mathtt{STypes} \,/ \equiv$.

SYSTEM $\mathscr{M}$

- 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. $\forall k \in K,\ S_k \equiv S'_{\sigma(k)}$.
- We set $\mathtt{Types}_{\mathscr{M}} = \mathtt{Types} / \equiv$.
  The set of multiset types is $\mathtt{STypes} / \equiv$.
- To obtain System $\mathscr{M}$, take the rules of $\mathscr{M}_0$ coinductively (with those types and multiset types).