Some Uses of Infinitary Intersection Types as Sequences

Pierre VIAL
IRIF, Paris 7

Rencontres Chocola

December 1, 2016

INVARIANTS OF EXECUTION

- In the course of its execution, a program passes through different states.

INVARIANTS OF EXECUTION

- In the course of its execution, a program passes through different states.
- Finding a **denotation** to a program = assigning to it an **invariant of execution** (*i.e.* an object that must the same for all its states).

INVARIANTS OF EXECUTION

- In the course of its execution, a program passes through different states.
- Finding a **denotation** to a program = assigning to it an **invariant of execution** (*i.e.* an object that must the same for all its states).
- The denotation of a program gives us some informations about its behaviour. Usually, **dynamical information** (related to its execution).

INVARIANTS OF EXECUTION

- In the course of its execution, a program passes through different states.
- Finding a **denotation** to a program = assigning to it an **invariant of execution** (*i.e.* an object that must the same for all its states).
- The denotation of a program gives us some informations about its behaviour. Usually, **dynamical information** (related to its execution).
- Usually, the information by a denotation implies that the concerned program is **terminating**.

INVARIANTS OF EXECUTION

- In the course of its execution, a program passes through different states.
- Finding a **denotation** to a program = assigning to it an **invariant of execution** (*i.e.* an object that must the same for all its states).
- The denotation of a program gives us some informations about its behaviour. Usually, **dynamical information** (related to its execution).
- Usually, the information by a denotation implies that the concerned program is **terminating**.
- Another use of denotations: **equating** or **separating programs** *i.e.* two states that have different denotations cannot be instances of the same program.

TYPES AS INVARIANTS OF EXECUTION

- λ-terms: programs, β-reduction step: execution step.

TYPES AS INVARIANTS OF EXECUTION

- λ-terms: programs, β-reduction step: execution step.
- **Normalizability:** termination.
  Many variants: head-n, weak-n, strong-n,...

TYPES AS INVARIANTS OF EXECUTION

- λ-terms: programs, β-reduction step: execution step.
- **Normalizability:** termination.
  Many variants: head-n, weak-n, strong-n,...
- **Types:** check *statically* (without reducing) that a term is normalizable (**soundness** of a type system).

TYPES AS INVARIANTS OF EXECUTION

- λ-terms: programs, β-reduction step: execution step.
- **Normalizability:** termination.
  Many variants: head-n, weak-n, strong-n,...
- **Types:** check *statically* (without reducing) that a term is normalizable (**soundness** of a type system).
- Typing: assigning formulas (called *types*) to variables.
  The type of a λ-term can be computed, if some *typing rules* are respected.

TYPES AS INVARIANTS OF EXECUTION

- λ-terms: programs, β-reduction step: execution step.
- **Normalizability:** termination.
  Many variants: head-n, weak-n, strong-n,...
- **Types:** check *statically* (without reducing) that a term is normalizable (**soundness** of a type system).
- Typing: assigning formulas (called *types*) to variables.
  The type of a λ-term can be computed, if some *typing rules* are respected.
- When a type system enjoys **subject reduction** and **expansion**, types are execution invariants (and they usually provide us with models of λ-calculus).

NON-TERMINATING PROGRAMS

- Often given an "empty" denotation (a model that equates all the non-terminating terms is said to be **sensible**). However:

NON-TERMINATING PROGRAMS

- Often given an "empty" denotation (a model that equates all the non-terminating terms is said to be **sensible**). However:
- Not all non-terminating progams are *meaningless*.
  (For instance, streams, a program keeping on printing the list of prime numbers, fixpoint combinators…)

NON-TERMINATING PROGRAMS

- Often given an "empty" denotation (a model that equates all the non-terminating terms is said to be **sensible**). However:
- Not all non-terminating progams are *meaningless*.
  (For instance, streams, a program keeping on printing the list of prime numbers, fixpoint combinators...)
- Some programs are non terminating but **productive**.

NON-TERMINATING PROGRAMS

- Often given an "empty" denotation (a model that equates all the non-terminating terms is said to be **sensible**). However:
- Not all non-terminating progams are *meaningless*.
  (For instance, streams, a program keeping on printing the list of prime numbers, fixpoint combinators. . .)
- Some programs are non terminating but **productive**.
- Many possible definitions or variants of sound non termination
  Klop and alii[95], Endrullis,Polonsky and alii[15]

CONTENTS OF THIS TALK

CONTENTS OF THIS TALK

- **Klop's Question:** a normalizability problem.
  A good opportunity to understand quantitative type systems and the differences between multiset and sequential constructions (**System** $\mathtt{S}$), as well as the problems raised by coinductive types

CONTENTS OF THIS TALK

- **Klop's Question:** a normalizability problem.
  A good opportunity to understand quantitative type systems and the differences between multiset and sequential constructions (**System** $\mathtt{S}$), as well as the problems raised by coinductive types
- System $\mathtt{S}$ is **completey unsound**: it types any term.
  Good news ! It provides us with an model for pure λ-calculus with new features (sensitivity to the **order** of λ-terms).

CONTENTS OF THIS TALK

- **Klop's Question:** a normalizability problem.
  A good opportunity to understand quantitative type systems and the differences between multiset and sequential constructions (**System** $\mathtt{S}$), as well as the problems raised by coinductive types

- System $\mathtt{S}$ is **completey unsound**: it types any term.
  Good news ! It provides us with an model for pure λ-calculus with new features (sensitivity to the **order** of λ-terms).

- The **collapse** of System $\mathtt{S}$ on System $\mathscr{R}$ is **surjective**.
  Every multiset based derivation is the collapse of a sequence based derivation. No loss of expressivity while resorting to $\mathtt{S}$.

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 $\wedge$ (collects the types assigned to a variable).

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

WHAT KIND OF INTERSECTION?

Intersection $\wedge$ (collects the types assigned to a variable).
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 $\wedge$ (collects the types assigned to a variable).
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 $\wedge$ (collects the types assigned to a variable).
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 $\wedge$ (collects the types assigned to a variable).
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**

MULTISETS VS SEQUENCES

- $\mathscr{M}(X)$: **multisets** of elements of x.

MULTISETS VS SEQUENCES

- $\mathscr{M}(X)$: **multisets** of elements of x.
  - $[a, b, a] = [a, b, b] \neq [a, b]$

MULTISETS VS SEQUENCES

- $\mathscr{M}(X)$: **multisets** of elements of x.
  - $[a, b, a] = [a, b, b] \neq [a, b]$
  - $[a, b, b] + [a, c] := [a, a, b, b, c]$

MULTISETS VS SEQUENCES

- $\mathscr{M}(X)$: **multisets** of elements of x.
  - $[a, b, a] = [a, b, b] \neq [a, b]$
  - $[a, b, b] + [a, c] := [a, a, b, b, c]$
  - $[a]_3 := [a, a, a]$

MULTISETS VS SEQUENCES

- $\mathscr{M}(X)$: **multisets** of elements of x.
  - $[a,b,a] = [a,b,b] \neq [a,b]$
  - $[a,b,b] + [a,c] := [a,a,b,b,c]$
  - $[a]_3 := [a,a,a]$
- $\mathtt{S}(X)$: **sequences** of elements of x.

MULTISETS VS SEQUENCES

- $\mathscr{M}(X)$: **multisets** of elements of x.
  - $[a, b, a] = [a, b, b] \neq [a, b]$
  - $[a, b, b] + [a, c] := [a, a, b, b, c]$
  - $[a]_3 := [a, a, a]$

- $\mathtt{S}(X)$: **sequences** of elements of x.
  - $(x_k)_{k \in K}$ where $K \subset \mathbb{N} \setminus \{0, 1\}$ and $\forall k \in K,\ x_k \in K$

MULTISETS VS SEQUENCES

- $\mathscr{M}(X)$: **multisets** of elements of x.
  - $[a, b, a] = [a, b, b] \neq [a, b]$
  - $[a, b, b] + [a, c] := [a, a, b, b, c]$
  - $[a]_3 := [a, a, a]$

- $\mathtt{S}(X)$: **sequences** of elements of x.
  - $(x_k)_{k \in K}$ where $K \subset \mathbb{N} \setminus \{0, 1\}$ and $\forall k \in K,\ x_k \in K$
  - $(x_k)_{k \in K}$ with $K = \{2, 5, 8\},\ x_2 = a,\ x_3 = b,\ x_8 = a$ written:

$$(2 \cdot a,\ 3 \cdot b,\ 8 \cdot a)$$

MULTISETS VS SEQUENCES

- $\mathscr{M}(X)$: **multisets** of elements of x.
  - $[a,b,a] = [a,b,b] \neq [a,b]$
  - $[a,b,b] + [a,c] := [a,a,b,b,c]$
  - $[a]_3 := [a,a,a]$

- $\mathtt{S}(X)$: **sequences** of elements of x.
  - $(x_k)_{k\in K}$ where $K \subset \mathbb{N} \setminus \{0,1\}$ and $\forall k \in K,\ x_k \in K$
  - $(x_k)_{k\in K}$ with $K = \{2,5,8\},\ x_2 = a,\ x_3 = b,\ x_8 = a$ written:

$$(2\cdot a,\ 3\cdot b,\ 8\cdot a)$$

  - Integer $k \in K$ called a **track**.

MULTISETS VS SEQUENCES

- $\mathscr{M}(X)$: **multisets** of elements of x.
  - $[a, b, a] = [a, b, b] \neq [a, b]$
  - $[a, b, b] + [a, c] := [a, a, b, b, c]$
  - $[a]_3 := [a, a, a]$

- $\mathtt{S}(X)$: **sequences** of elements of x.
  - $(x_k)_{k \in K}$ where $K \subset \mathbb{N} \setminus \{0, 1\}$ and $\forall k \in K,\ x_k \in K$
  - $(x_k)_{k \in K}$ with $K = \{2, 5, 8\},\ x_2 = a,\ x_3 = b,\ x_8 = a$ written:

$$(2 \cdot a,\ 3 \cdot b,\ 8 \cdot a)$$

  - Integer $k \in K$ called a **track**.
  - $(2 \cdot a,\ 3 \cdot b,\ 8 \cdot a) \uplus (4 \cdot a,\ 9 \cdot c) = (2 \cdot a,\ 3 \cdot b,\ 4 \cdot a,\ 8 \cdot a,\ 9 \cdot c)$

MULTISETS VS SEQUENCES

- $\mathscr{M}(X)$: **multisets** of elements of x.
  - $[a, b, a] = [a, b, b] \neq [a, b]$
  - $[a, b, b] + [a, c] := [a, a, b, b, c]$
  - $[a]_3 := [a, a, a]$

- $\mathtt{S}(X)$: **sequences** of elements of x.
  - $(x_k)_{k \in K}$ where $K \subset \mathbb{N} \setminus \{0, 1\}$ and $\forall k \in K,\ x_k \in K$
  - $(x_k)_{k \in K}$ with $K = \{2, 5, 8\},\ x_2 = a,\ x_3 = b,\ x_8 = a$ written:

$$(2 \cdot a,\ 3 \cdot b,\ 8 \cdot a)$$

  - Integer $k \in K$ called a **track**.
  - $(2 \cdot a,\ 3 \cdot b,\ 8 \cdot a) \uplus (4 \cdot a,\ 9 \cdot c) = (2 \cdot a,\ 3 \cdot b,\ 4 \cdot a,\ 8 \cdot a,\ 9 \cdot c)$
  - $(2 \cdot a,\ 3 \cdot b,\ 8 \cdot a) \uplus (3 \cdot b,\ 9 \cdot c)$ **not** defined (**incompatibility**).

Plan

HEREDITARY HEAD-NORMALIZATION

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

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

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

.

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:

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

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)
  - 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:

$$\lambda x_1 \ldots x_p.\underbrace{\boxed{x}}_{\text{head variable}}\, \underbrace{u_1 \ldots u_q}_{\text{head arguments}} \qquad (p, q \geqslant 0)$$

- 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 QUESTION

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

KLOP'S QUESTION

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

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

KLOP'S QUESTION

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

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

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

KLOP'S QUESTION

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

- **Klop's Question** [**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?

ANSWERING KLOP'S QUESTION...

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

ANSWERING KLOP'S QUESTION...

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

- Understand why **commutative intersection** is **unfit** to express those key notions.

Answering Klop's Question...

- Present the key notions of **truncations** and **approximability** (meant to avoid *irrelevant* derivations).
- Understand why **commutative 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 $\mathscr{R}_0$ (GARDNER/DE CARVALHO)

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

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

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

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

Examples:

$$\dfrac{\dfrac{}{x : [\tau] \vdash x : \tau}\ \texttt{ax}}{\vdash \lambda x.x : [\tau] \to \tau}\ \texttt{abs} \qquad \dfrac{\dfrac{}{x : [\tau] \vdash x : \tau}\ \texttt{ax}}{x : [\tau] \vdash \lambda y.x : [\,] \to \tau}\ \texttt{abs}$$

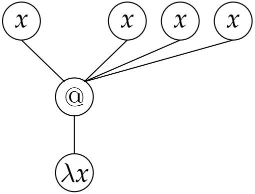
ALTERNATIVE PRESENTATION

Standard presentation

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

ALTERNATIVE PRESENTATION

Alternative presentation

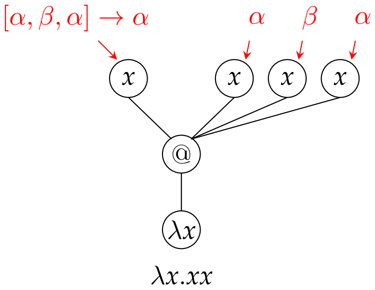


$\lambda x.xx$

- Indicate the arity of application rules.

Alternative Presentation

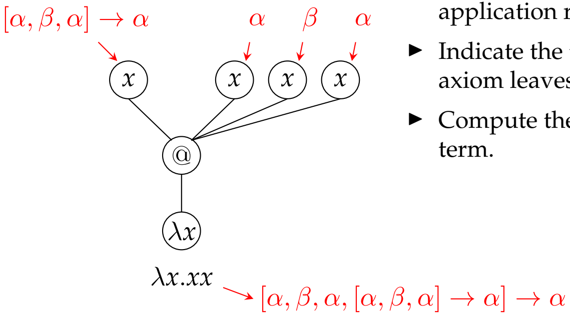
Alternative presentation



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

ALTERNATIVE PRESENTATION

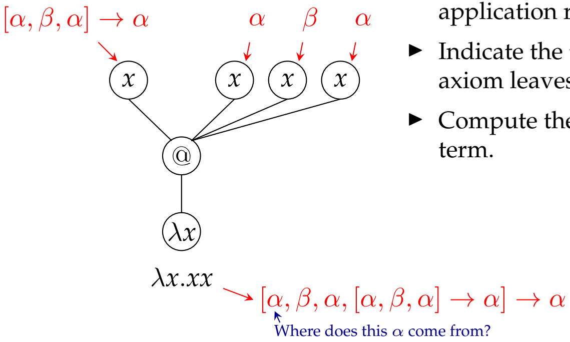
Alternative presentation



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

ALTERNATIVE PRESENTATION

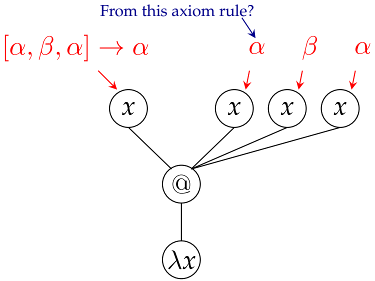
Alternative presentation



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

ALTERNATIVE PRESENTATION

Alternative presentation



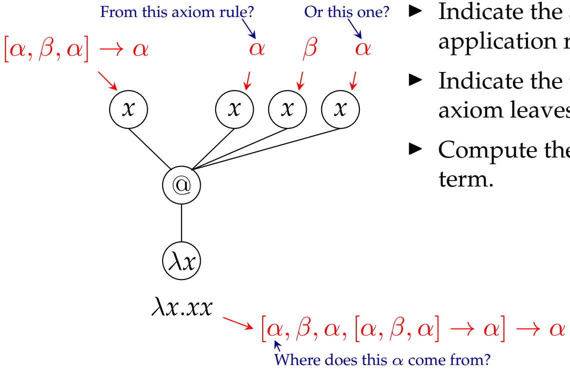
$\lambda x.xx$ $\to$ $[\alpha, \beta, \alpha, [\alpha, \beta, \alpha] \to \alpha] \to \alpha$

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 $\exists \Pi' \rhd \Gamma \vdash t' : \tau$

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

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

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

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

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

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

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

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

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

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

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

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

"association"

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

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

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

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

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

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

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

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

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

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

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

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

Vocabulary:
We say each **association** (between x-axiom leaves and arg-derivations) 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 $\exists \Pi' \rhd \Gamma \vdash t' : \tau$

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

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

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 $\mathscr{R}_0$.

NORMALIZABILITY RESULTS

Proposition

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

Proposition

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

NORMALIZABILITY RESULTS

Proposition

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

Proposition

A term is WN iff it is typable in $\mathscr{R}_0$ 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 $\mathscr{R}_0$.

Proposition

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

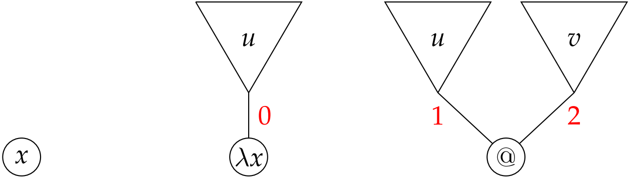
Definition

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.

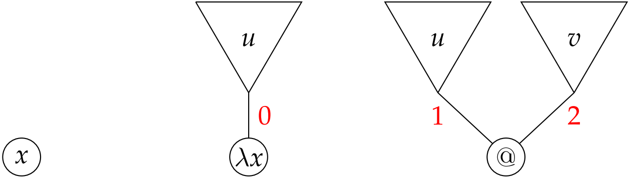
PLAN

∞-TERMS



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

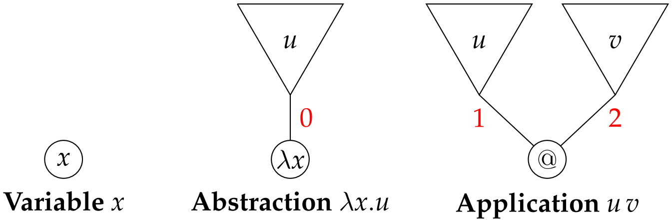
∞-TERMS



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



- **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.:

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

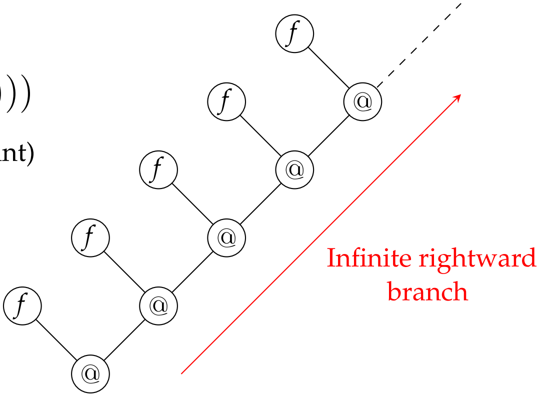
001-TERMS

Λ^{001}: the set of ∞-terms t s.t.:

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

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

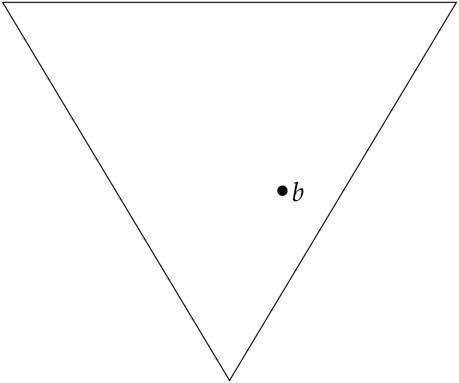
i.e. $f^{\omega} = f(f^{\omega})$ (fixpoint)



001-TERMS

Λ^{001}: the set of ∞-terms t s.t.:

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

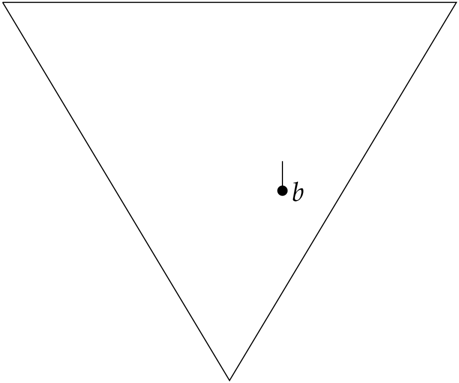


- Start from $b \in \texttt{supp}(t)$

001-TERMS

Λ^{001}: the set of ∞-terms t s.t.:

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

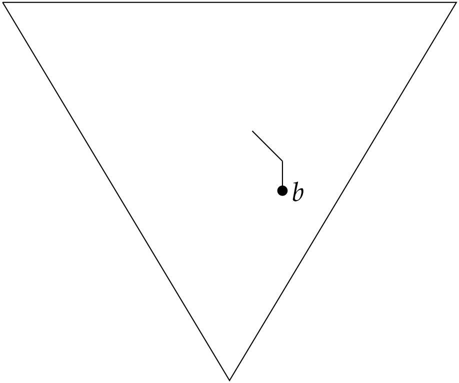


- Start from $b \in \texttt{supp}(t)$
- Move $\uparrow$ or $\nwarrow$ a.d. does not increase

001-TERMS

Λ^{001}: the set of ∞-terms t s.t.:

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

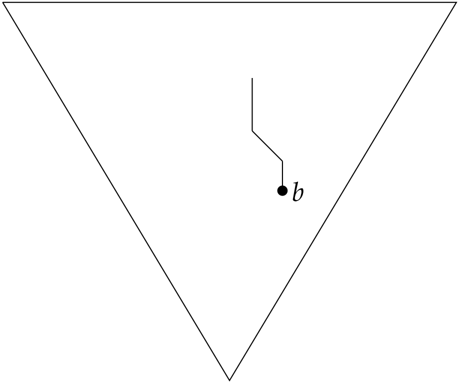


- Start from $b \in \texttt{supp}(t)$
- Move $\uparrow$ or $\nwarrow$ a.d. does not increase

001-TERMS

Λ^{001}: the set of ∞-terms t s.t.:

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

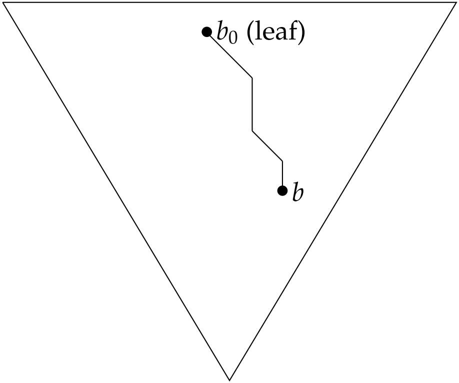


- Start from $b \in \texttt{supp}(t)$
- Move $\uparrow$ or $\nwarrow$ a.d. does not increase

001-TERMS

Λ^{001}: the set of ∞-terms t s.t.:

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



- Start from $b \in \texttt{supp}(t)$
- Move $\uparrow$ or $\nwarrow$
  a.d. does not increase
- A leaf b_0 must be reached

STRONG CONVERGENCE

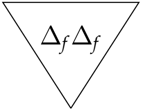
Definition

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

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

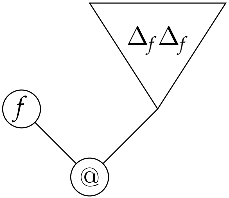
$$\Delta_f\Delta_f \to f(\Delta_f\Delta_f) \to f^2(\Delta_f\Delta_f) \to f^3(\Delta_f\Delta_f) \to f^4(\Delta_f\Delta_f) \to \ldots \to^\infty f^\omega$$



STRONG CONVERGENCE

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

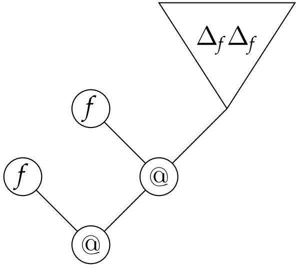
$$\Delta_f\Delta_f \to f(\Delta_f\Delta_f) \to f^2(\Delta_f\Delta_f) \to f^3(\Delta_f\Delta_f) \to f^4(\Delta_f\Delta_f) \to \ldots \to^\infty f^\omega$$



STRONG CONVERGENCE

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

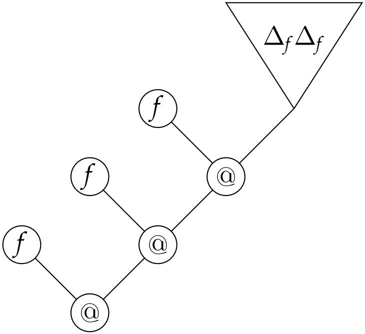
$$\Delta_f\Delta_f \to f(\Delta_f\Delta_f) \to f^2(\Delta_f\Delta_f) \to f^3(\Delta_f\Delta_f) \to f^4(\Delta_f\Delta_f) \to \ldots \to^\infty f^\omega$$



STRONG CONVERGENCE

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

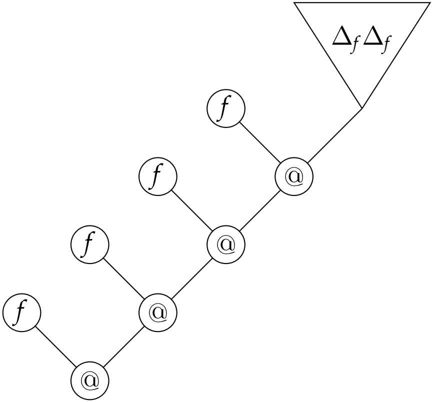
$$\Delta_f\Delta_f \to f(\Delta_f\Delta_f) \to f^2(\Delta_f\Delta_f) \to f^3(\Delta_f\Delta_f) \to f^4(\Delta_f\Delta_f) \to \ldots \to^\infty f^\omega$$



STRONG CONVERGENCE

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

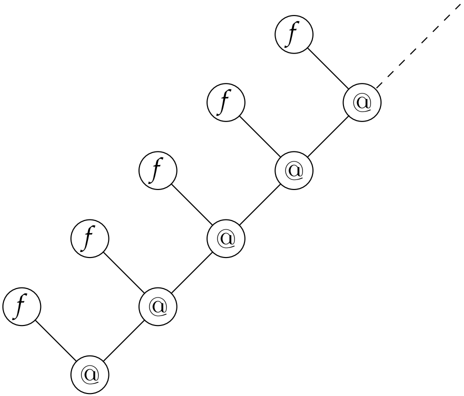
$$\Delta_f\Delta_f \to f(\Delta_f\Delta_f) \to f^2(\Delta_f\Delta_f) \to f^3(\Delta_f\Delta_f) \to f^4(\Delta_f\Delta_f) \to \ldots \to^\infty f^\omega$$



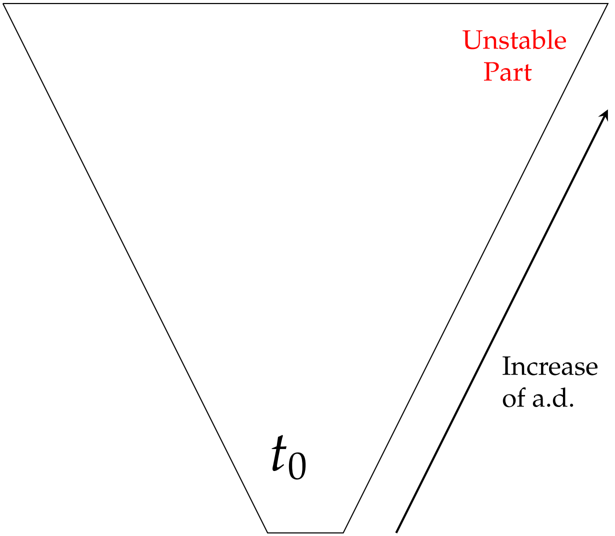
STRONG CONVERGENCE

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

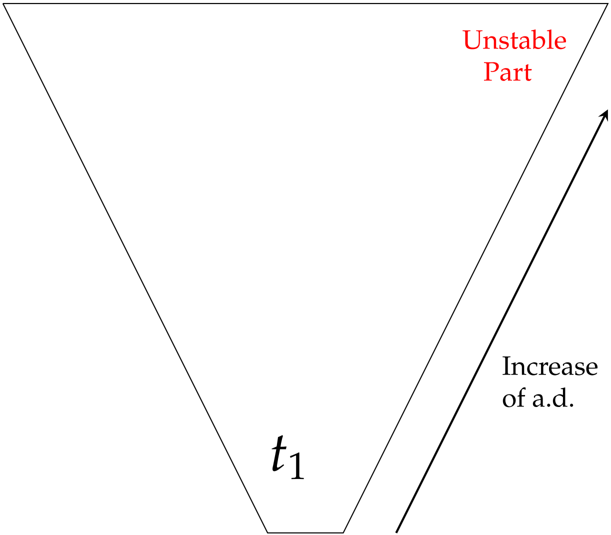
$$\Delta_f\Delta_f \to f(\Delta_f\Delta_f) \to f^2(\Delta_f\Delta_f) \to f^3(\Delta_f\Delta_f) \to f^4(\Delta_f\Delta_f) \to \ldots \to^\infty f^\omega$$



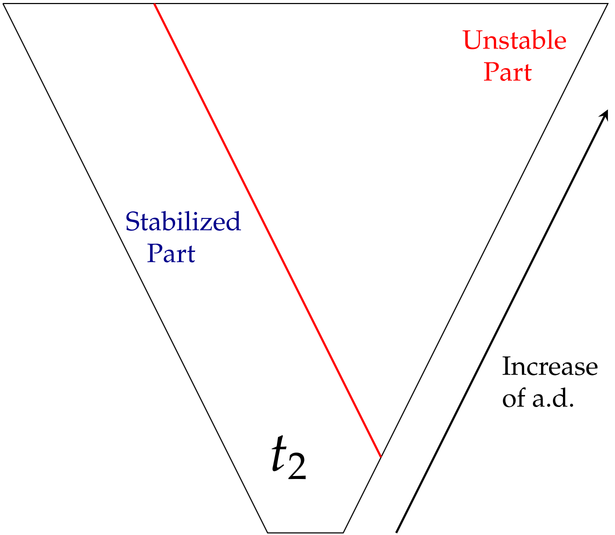
STRONG CONVERGENCE



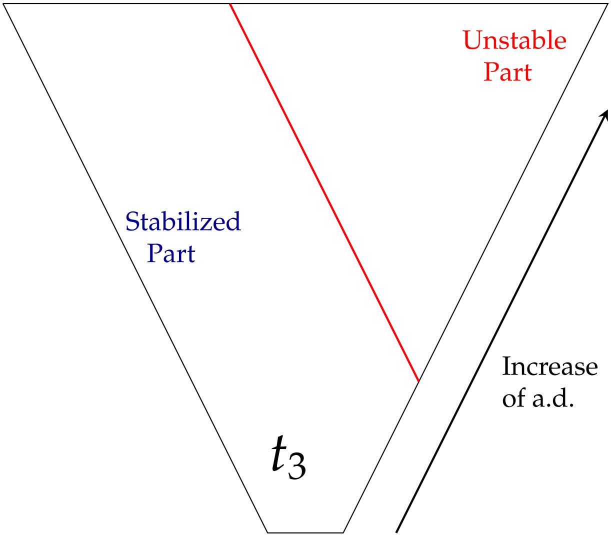
STRONG CONVERGENCE



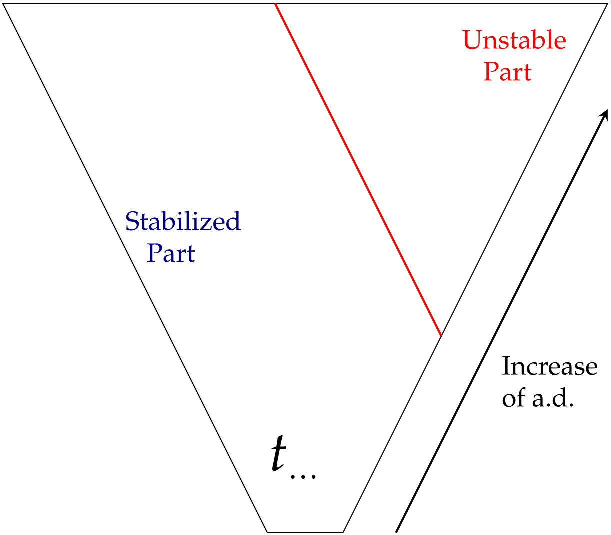
STRONG CONVERGENCE



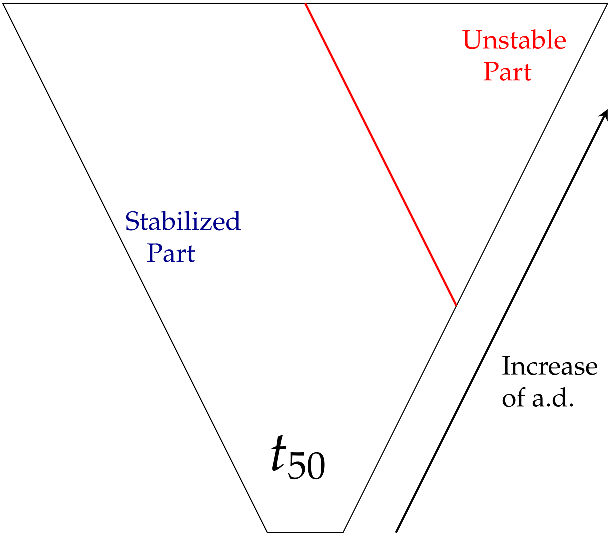
STRONG CONVERGENCE



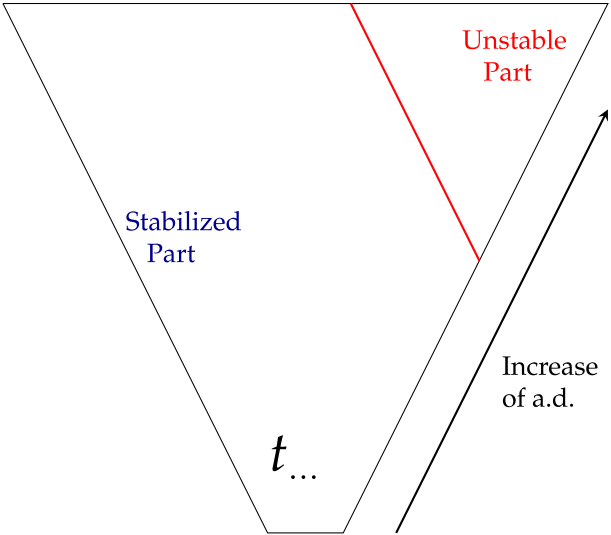
STRONG CONVERGENCE



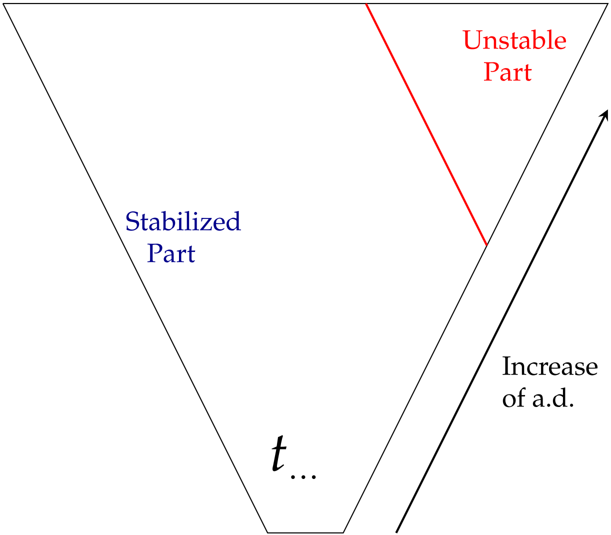
STRONG CONVERGENCE



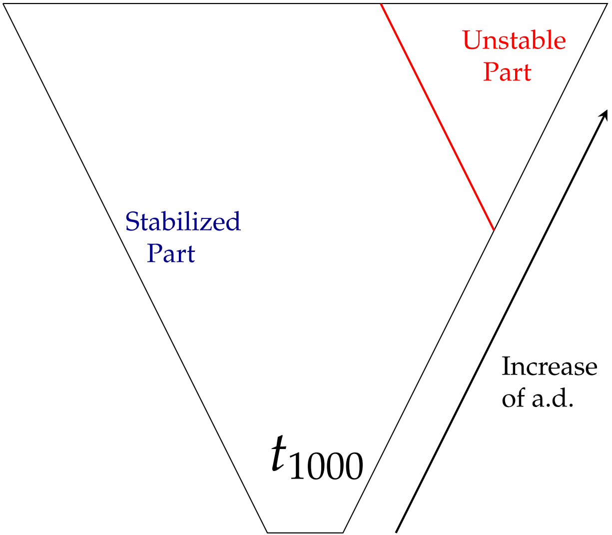
STRONG CONVERGENCE



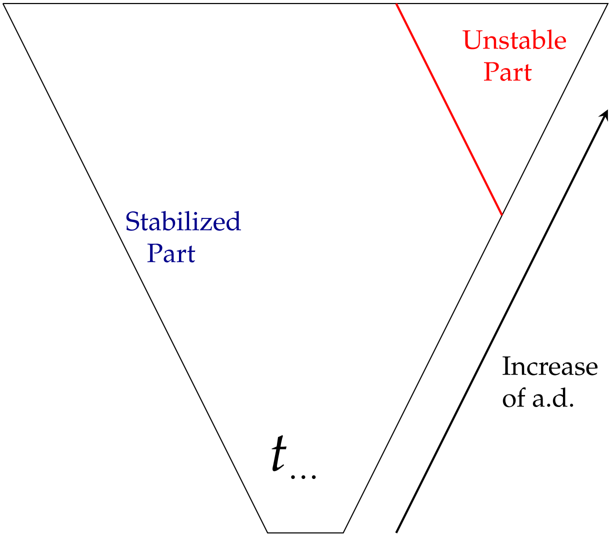
Strong Convergence



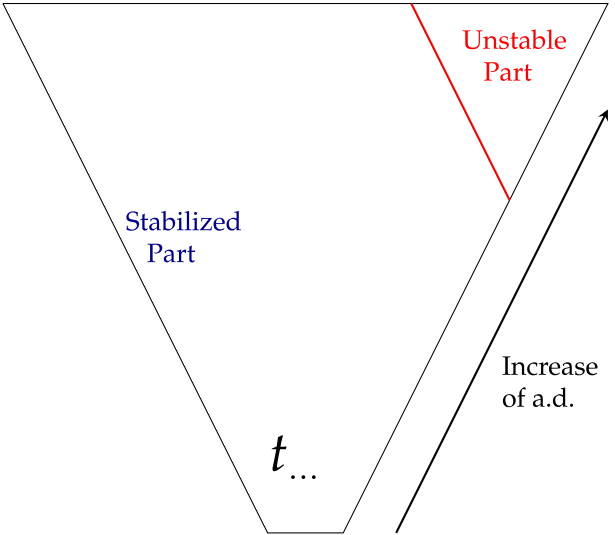
STRONG CONVERGENCE



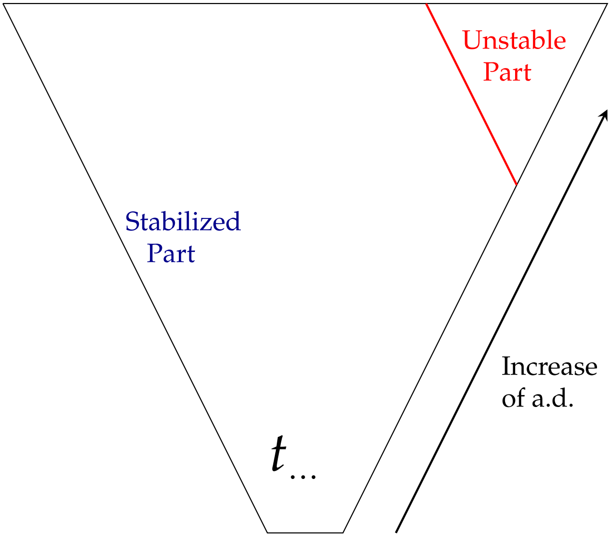
STRONG CONVERGENCE



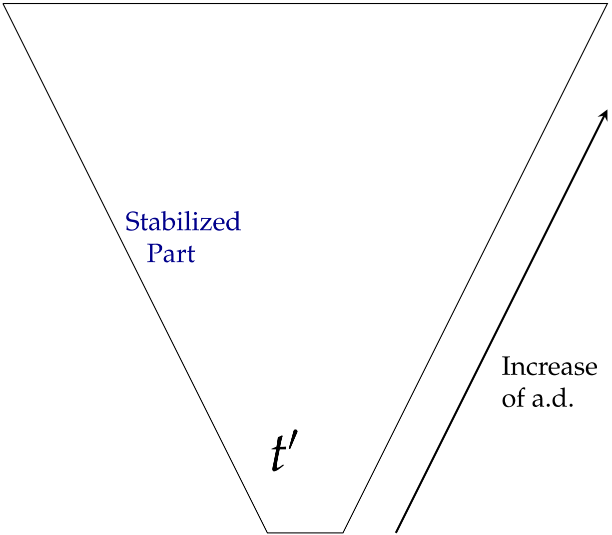
STRONG CONVERGENCE



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:

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

Definition (Infinitary WN)

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

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)

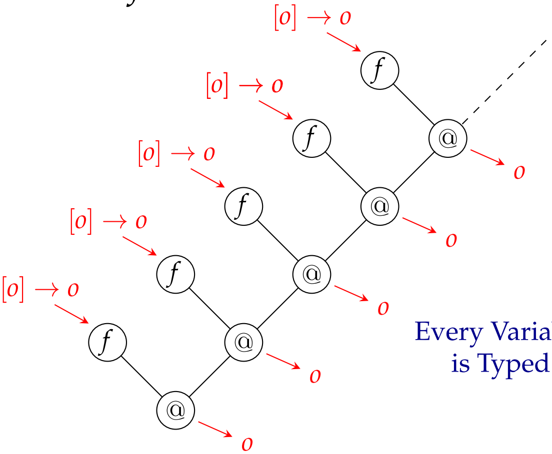
A 001-term is WN if it can be reduced to a NF through at least one s.c.r.s.

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

Plan

Truncation (Figures)

$$\Pi' \rhd \Gamma \vdash f^\omega : o$$

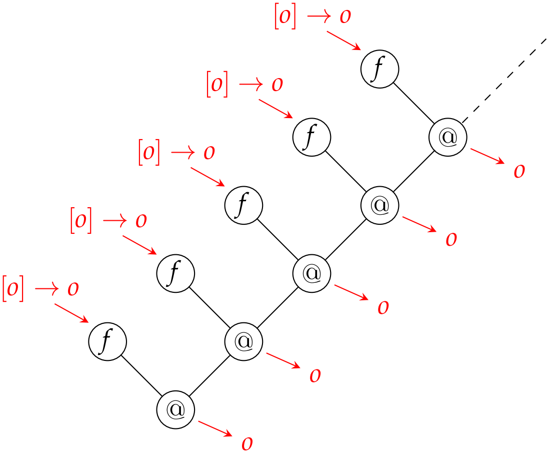


Every Variable
is Typed

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

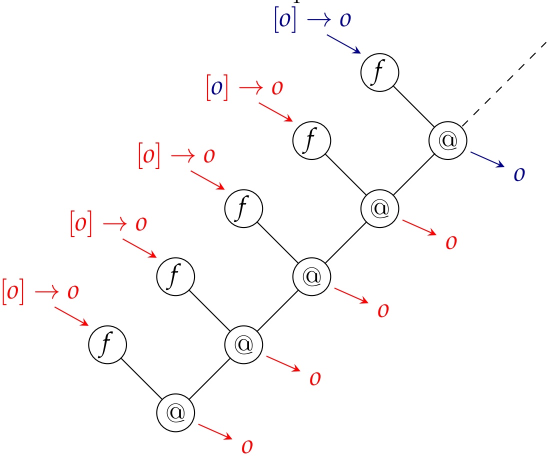
TRUNCATION (FIGURES)

Π' can be **truncated** into Π'_4:



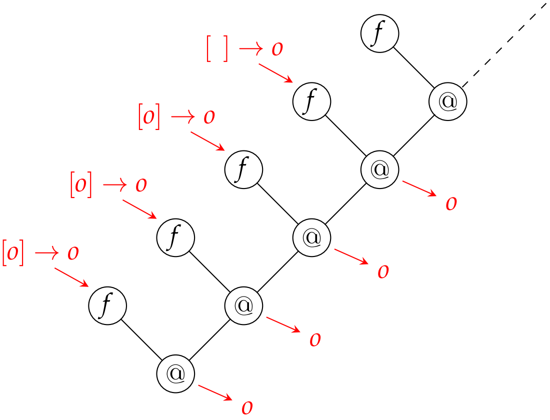
TRUNCATION (FIGURES)

Π' can be **truncated** into Π'_4:



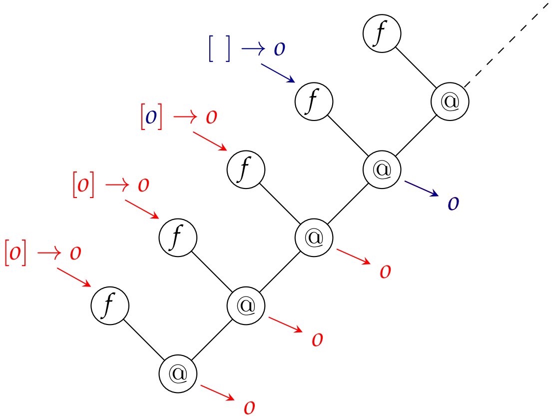
TRUNCATION (FIGURES)

Π' can be **truncated** into Π'_4:



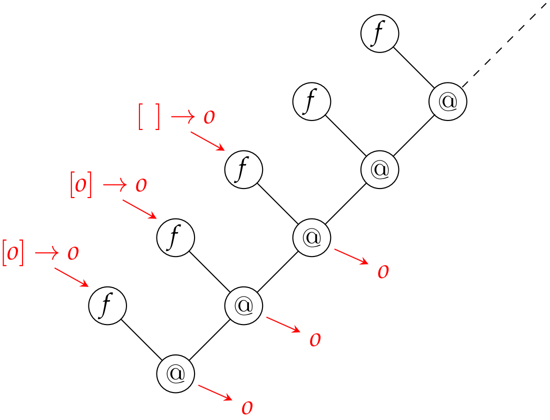
TRUNCATION (FIGURES)

Π' can be **truncated** into Π'_3:



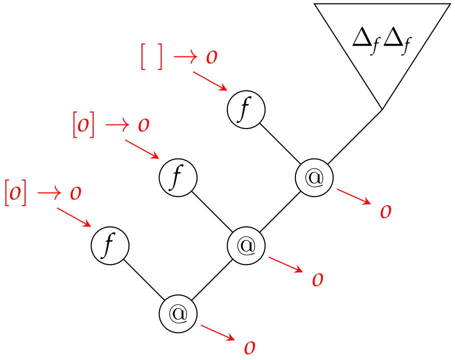
TRUNCATION (FIGURES)

Π' can be **truncated** into Π'_3:



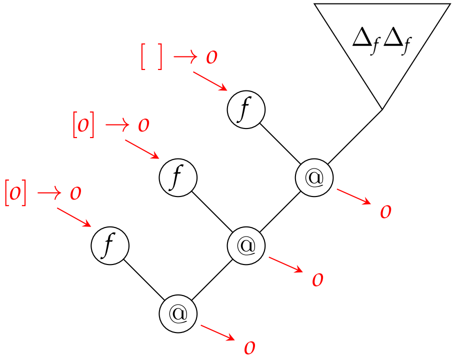
Truncation (Figures)

f^ω may be replaced by $f^3(\Delta_f\Delta_f)$ in Π_3',
yielding Π_3^3 :



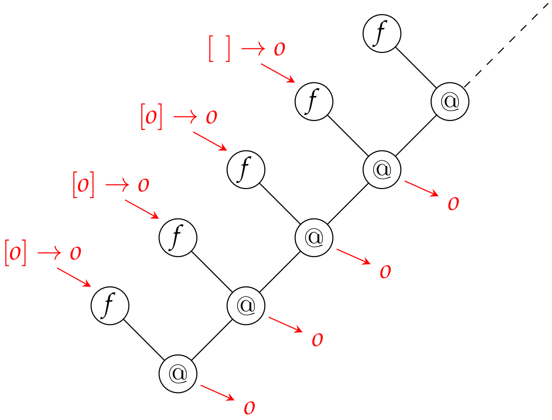
TRUNCATION (FIGURES)

Π_3^3 may be expanded 3 times,
yielding $\Pi_3 \rhd \Delta_f \Delta_f$:



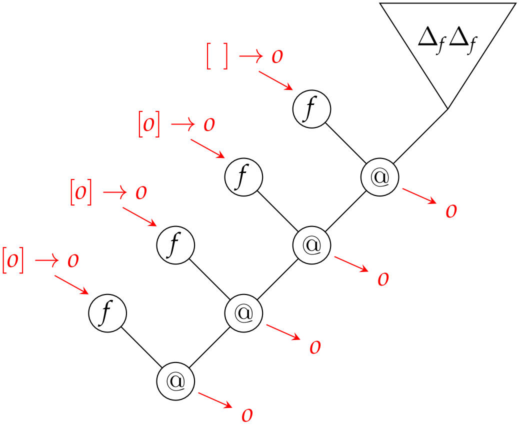
TRUNCATION (FIGURES)

Back to Π'_4, level 4 truncation of Π' :



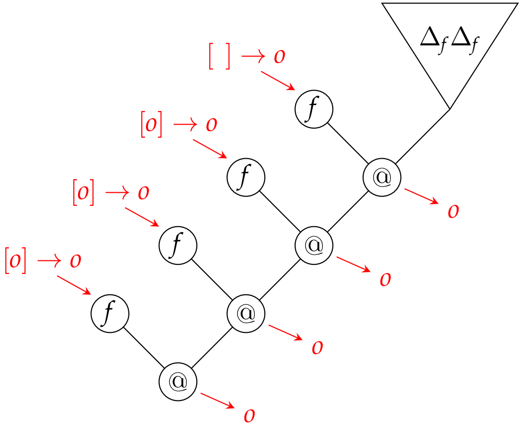
TRUNCATION (FIGURES)

f^ω may be replaced by $f^4(\Delta_f\Delta_f)$ in Π_3',
yielding Π_4^4 :



TRUNCATION (FIGURES)

Π_4^4 may be expanded 4 times,
yielding $\Pi_4 \rhd \Delta_f \Delta_f$:



Recipe for ∞-Expansion

Question: how do we expand $\Pi' \rhd f^\omega$, to get Π, typing $\Delta_f \Delta_f$?

RECIPE FOR ∞-EXPANSION

Question: how do we expand $\Pi' \rhd f^{\omega}$, to get Π, typing $\Delta_f \Delta_f$?

We have the idea of **level n truncation** of Π' and the idea of **subject substitution** (by a reduct of finite rank, in a finite derivation).

RECIPE FOR ∞-EXPANSION

Question: how do we expand $\Pi' \triangleright f^{\omega}$, to get Π, typing $\Delta_f \Delta_f$?

We have the idea of **level n truncation** of Π' and the idea of **subject substitution** (by a reduct of finite rank, in a finite derivation).

- Truncate Π' into ${}^{\mathrm{f}}\Pi'$, finite derivation typing f^{ω} (hint: replace an occ. of $[\alpha] \to \alpha$ by $[\,] \to \alpha$).

RECIPE FOR ∞-EXPANSION

Question: how do we expand $\Pi' \rhd f^\omega$, to get Π, typing $\Delta_f \Delta_f$?

We have the idea of **level n truncation** of Π' and the idea of **subject substitution** (by a reduct of finite rank, in a finite derivation).

- Truncate Π' into ${}^{\mathrm{f}}\Pi'$, finite derivation typing f^ω (hint: replace an occ. of $[\alpha] \to \alpha$ by $[\,] \to \alpha$).
- In ${}^{\mathrm{f}}\Pi'$, replace f^ω by $f^n(\Delta_f \Delta_f)$, for n great enough: you get ${}^{\mathrm{f}}\Pi'_n$.

RECIPE FOR ∞-EXPANSION

Question: how do we expand $\Pi' \rhd f^\omega$, to get Π, typing $\Delta_f \Delta_f$?

We have the idea of **level n truncation** of Π' and the idea of **subject substitution** (by a reduct of finite rank, in a finite derivation).

- Truncate Π' into ${}^{\mathrm{f}}\Pi'$, finite derivation typing f^ω (hint: replace an occ. of $[\alpha] \to \alpha$ by $[\,] \to \alpha$).
- In ${}^{\mathrm{f}}\Pi'$, replace f^ω by $f^n(\Delta_f \Delta_f)$, for n great enough: you get ${}^{\mathrm{f}}\Pi'_n$.
- Expand n times ${}^{\mathrm{f}}\Pi'_n$: you get Π_n typing $\Delta_f \Delta_f$.

RECIPE FOR ∞-EXPANSION

Question: how do we expand $\Pi' \rhd f^\omega$, to get Π, typing $\Delta_f \Delta_f$?

We have the idea of **level n truncation** of Π' and the idea of **subject substitution** (by a reduct of finite rank, in a finite derivation).

- Truncate Π' into ${}^{\mathrm{f}}\Pi'$, finite derivation typing f^ω (hint: replace an occ. of $[\alpha] \to \alpha$ by $[\,] \to \alpha$).
- In ${}^{\mathrm{f}}\Pi'$, replace f^ω by $f^n(\Delta_f \Delta_f)$, for n great enough: you get ${}^{\mathrm{f}}\Pi'_n$.
- Expand n times ${}^{\mathrm{f}}\Pi'_n$: you get Π_n typing $\Delta_f \Delta_f$.
- Take the join of all the Π_n (while $n \to \infty$): this defines Π, the desired expansion of Π'.

UNSOUNDNESS

- Expanding Π', we can get an unforgetful derivation Π typing $\Delta_f \Delta_f$.

Unsoundness

- Expanding Π', we can get an unforgetful derivation Π typing $\Delta_f \Delta_f$.
- Derivation Π features a type ρ coinductively defined by the fixpoint equation

$$\rho = [\rho]_\omega \to \rho$$

UNSOUNDNESS

- Expanding Π', we can get an unforgetful derivation Π typing $\Delta_f \Delta_f$.
- Derivation Π features a type ρ coinductively defined by the fixpoint equation
$$\rho = [\rho]_\omega \to \rho$$
- 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 o or $\rightarrow$ that we found inside each jugdment of P).

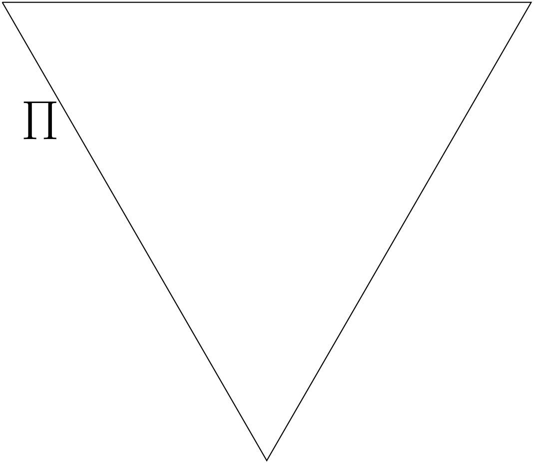
APPROXIMABILITY (HEURISTIC)

- Informally, see a derivation Π as a set of symbols (type variables o 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 ${}^{f}\Pi \leqslant \Pi$.

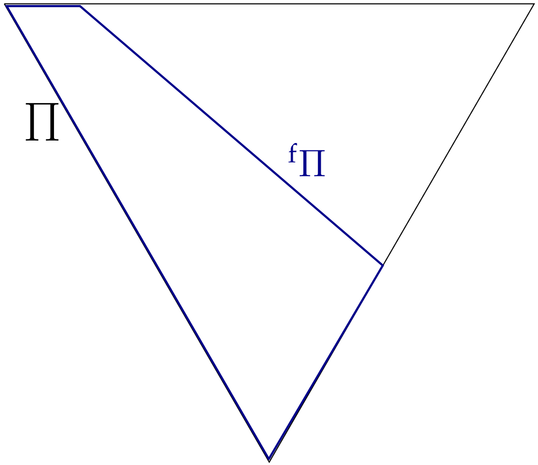
APPROXIMABILITY (HEURISTIC)

- Informally, see a derivation Π as a set of symbols (type variables o 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.

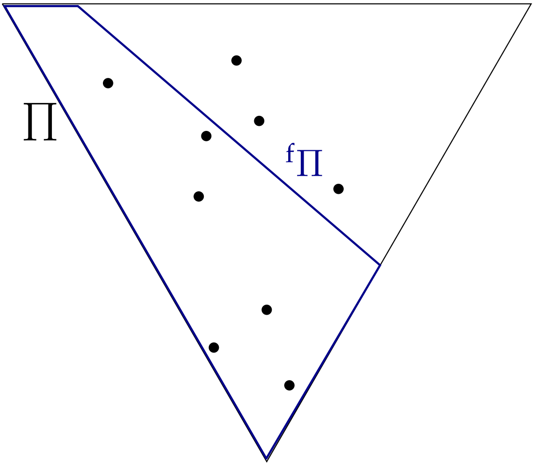
APPROXIMABILITY (FIGURE)



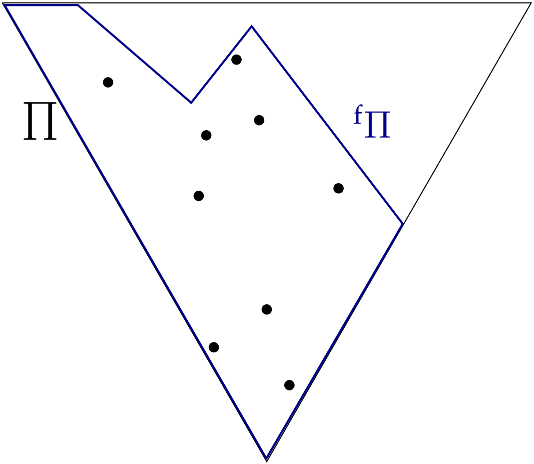
APPROXIMABILITY (FIGURE)



APPROXIMABILITY (FIGURE)

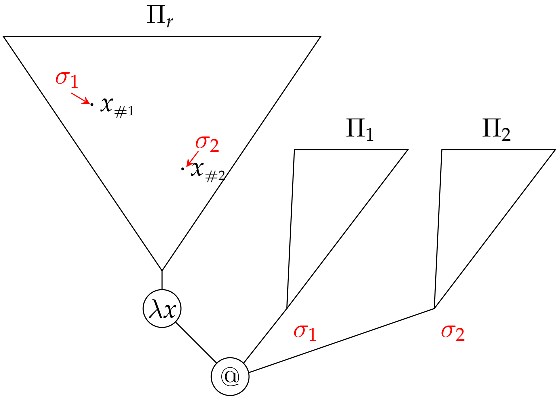


APPROXIMABILITY (FIGURE)



NON-DETERMINISM AND TRUNCATION

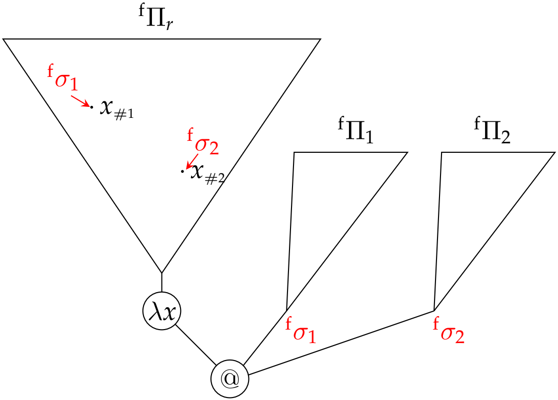
$$(\lambda x.r)s$$



NON-DETERMINISM AND TRUNCATION

$(\lambda x.r)s$

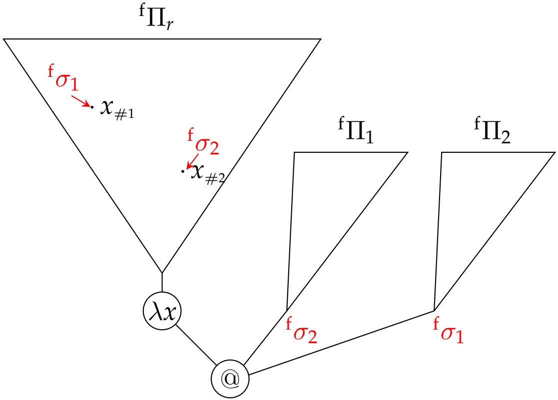
Truncation possibly affects every type nested inside Π.



NON-DETERMINISM AND TRUNCATION

$(\lambda x.r)s$

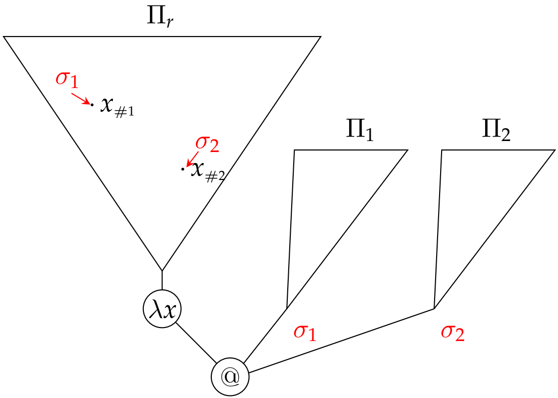
Truncation possibly affects every type nested inside Π.



NON-DETERMINISM AND TRUNCATION

$(\lambda x.r)s$

Assume $\sigma_1 = \sigma_2$.

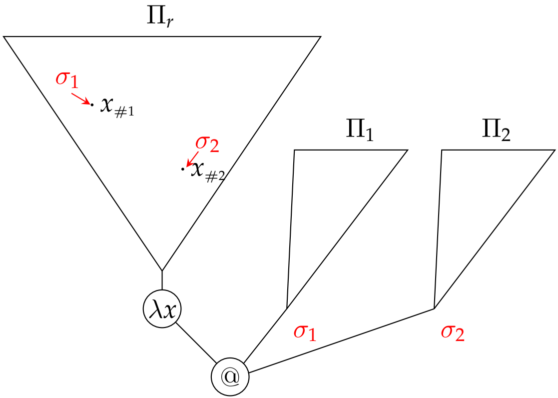


NON-DETERMINISM AND TRUNCATION

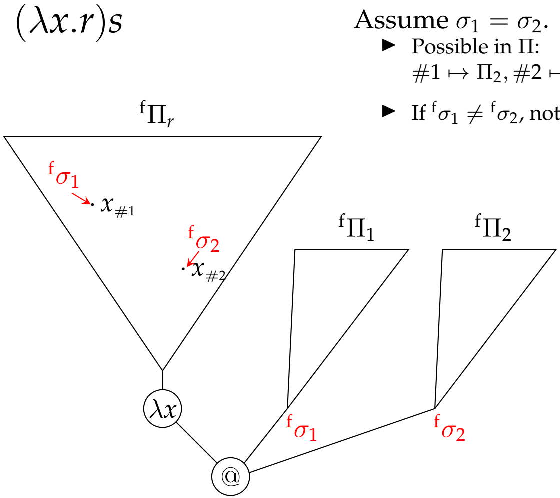
$(\lambda x.r)s$

Assume $\sigma_1 = \sigma_2$.

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



NON-DETERMINISM AND TRUNCATION



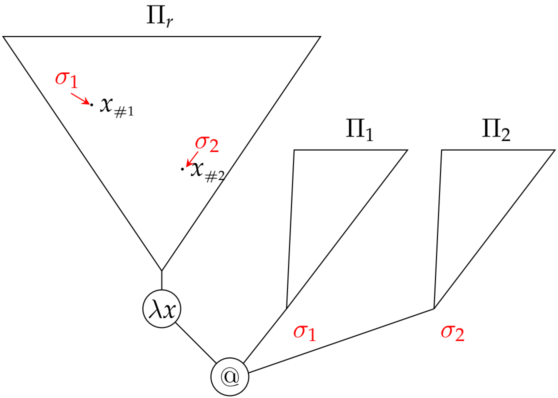
Assume $\sigma_1 = \sigma_2$.

- 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

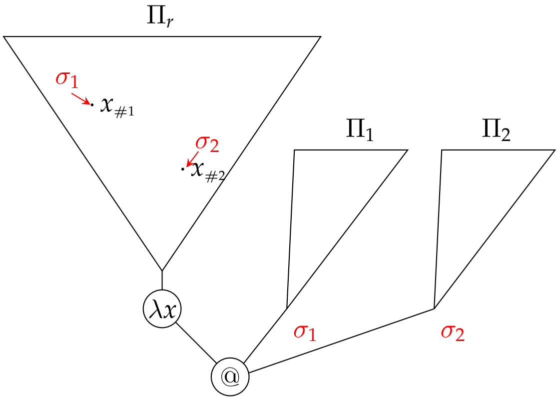
$(\lambda x.r)s$

Assume $\sigma_1 \neq \sigma_2$



NON-DETERMINISM AND TRUNCATION

$(\lambda x.r)s$

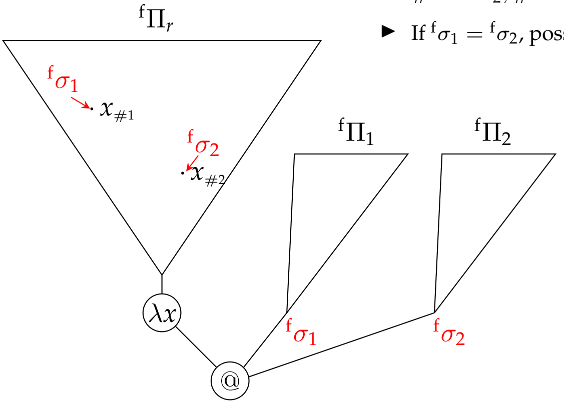


Assume $\sigma_1 \neq \sigma_2$

- 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

SEQUENTIAL INTERSECTION

- **Types:**

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

SEQUENTIAL INTERSECTION

- **Types:**
$$S_k, T \ ::= \ o \in \mathscr{O} \mid (S_k)_{k \in K} \to T$$
- **Sequence Type**:
  - Intersection type replacing multiset types.

SEQUENTIAL INTERSECTION

- **Types:**

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

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

SEQUENTIAL INTERSECTION

- **Types:**

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

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

SEQUENTIAL INTERSECTION

- **Types:**
$$S_k, T \ ::= \ o \in \mathscr{O} \mid (S_k)_{k \in K} \to T$$
- **Sequence Type**:
  - Intersection type replacing multiset types.
  - $F = (T_k)_{k \in K}$ where T_k types and $K \subset \mathbb{N} - \{0, 1\}$.
  - The integer indexes k are called **tracks**.
  - We also write $(S_k)_{k \in K} = (k \cdot S_k)_{k \in K}$.

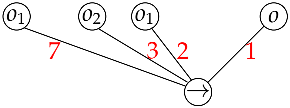
SEQUENTIAL INTERSECTION

- **Types:**

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

- **Sequence Type**:
  - Intersection type replacing multiset types.
  - $F = (T_k)_{k \in K}$ where T_k types and $K \subset \mathbb{N} - \{0, 1\}$.
  - The integer indexes k are called **tracks**.
  - We also write $(S_k)_{k \in K} = (k \cdot S_k)_{k \in K}$.

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



DERIVATIONS OF S

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

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

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

DERIVATIONS OF S

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

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

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

- 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:

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

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

- If $\mathrm{Rt}(C)$ and the $\mathrm{Rt}(D_k)$ are not pairwise disjoint, contexts are incompatible.
- Forget about the indexes: `S` collapses onto $\mathscr{D}$.

MAIN FEATURES

- **Trackability:** $\mathtt{S}$ features **pointers** called **bipositions** (every symbol used inside a derivation P can be pointed at).

MAIN FEATURES

- **Trackability:** $\mathtt{S}$ features **pointers** called **bipositions** (every symbol used inside a derivation P can be pointed at).

- Subject reduction is deterministic:

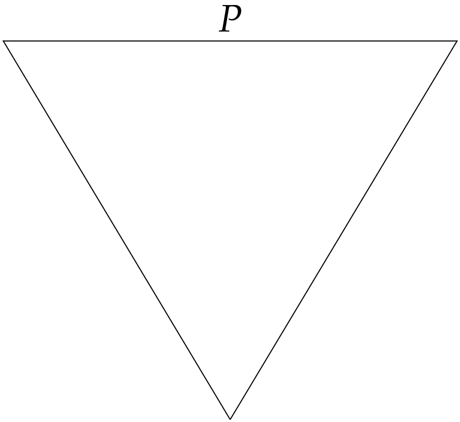
MAIN FEATURES

- **Trackability:** `S` features **pointers** called **bipositions** (every symbol used inside a derivation P can be pointed at).
- 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

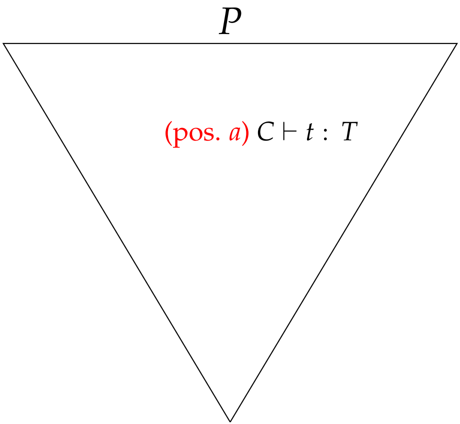
MAIN FEATURES

- **Trackability:** S features **pointers** called **bipositions** (every symbol used inside a derivation P can be pointed at).
- 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 $S = S_5$

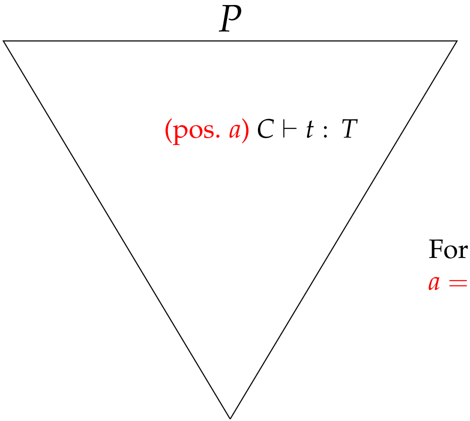
POINTERS



Pointers

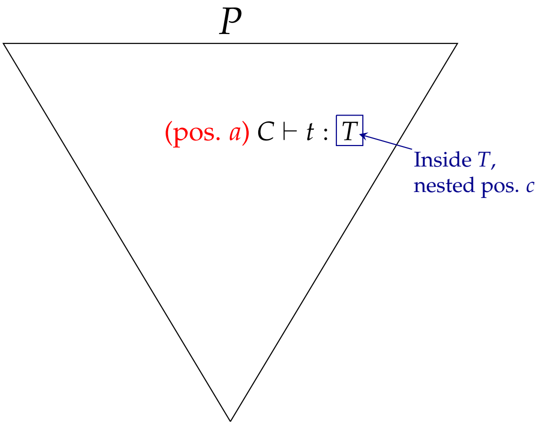


POINTERS

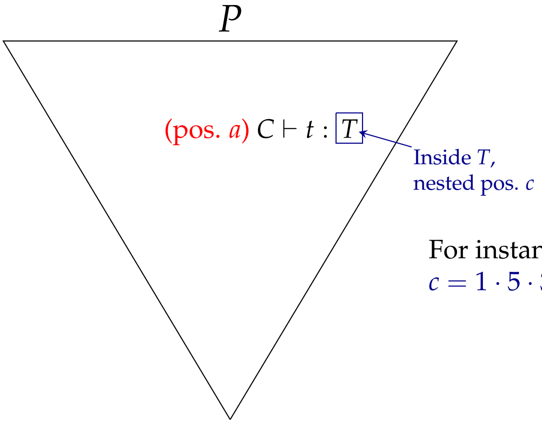


For instance
$a = 0 \cdot 1 \cdot 3 \cdot 0 \cdot 8 \cdot 1$

POINTERS

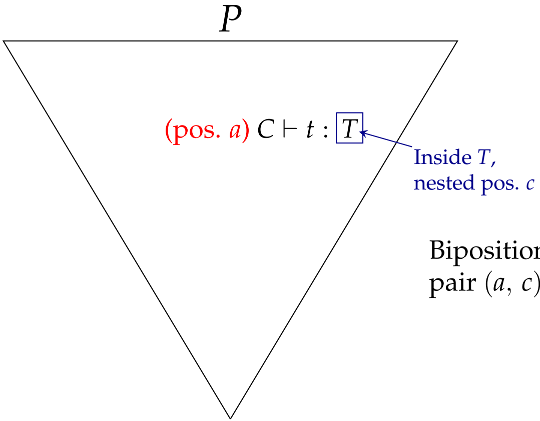


POINTERS



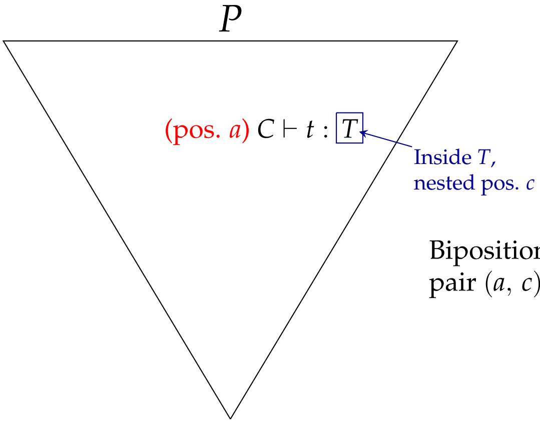
For instance
$c = 1 \cdot 5 \cdot 3 \cdot 1 \cdot 4$

Pointers



Biposition (right h.s.):
pair $(a,\ c)$

POINTERS



Biposition (right h.s.):
pair (a, c)

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

PLAN

APPROXIMABILITY

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

APPROXIMABILITY

- Every symbol inside a rigid derivation P has a **biposition** (a pointer 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 pointer inside a type nested in a judgment of P).
- A **finite part** B of P is *finite* subset of $\texttt{bisupp}(P)$.
- A **finite (or not) approximation** of P is a finite or not derivation induced by P on a finite part of P.

APPROXIMABILITY

- Every symbol inside a rigid derivation P has a **biposition** (a pointer inside a type nested in a judgment of P).
- A **finite part** B of P is *finite* subset of $\texttt{bisupp}(P)$.
- A **finite (or not) approximation** of P is a finite or not 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.

THE LATTICE OF APPROXIMATIONS

Proposition:

- The set of a $\mathtt{S}$-derivations typing a same term t is a c.p.o.
- The set of approximations of a derivation P is a complete lattice.
- The set of finite approximations of a derivation P is a lattice.

THE LATTICE OF APPROXIMATIONS

Proposition:

- The set of a $\mathtt{S}$-derivations typing a same term t is a c.p.o.
- The set of approximations of a derivation P is a complete lattice.
- The set of finite approximations of a derivation P is a lattice.

Order, meet and join are given by the set-theoretic operations $\subseteq$, $\cap$, $\cup$ on bisupports.

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

TYPABLE TERMS IN $\mathtt{S}$

- **Question:** let us drop approximability. What is the set of typable terms in $\mathtt{S}$?

TYPABLE TERMS IN $\mathtt{S}$

- **Question:** let us drop approximability. What is the set of typable terms in $\mathtt{S}$?
- We already know that $\mathtt{S}$ is **unsound** ($\mathtt{S}$ can type unproductive terms, like Ω). Two possibilities:

TYPABLE TERMS IN $\mathtt{S}$

- **Question:** let us drop approximability. What is the set of typable terms in $\mathtt{S}$?
- We already know that $\mathtt{S}$ is **unsound** ($\mathtt{S}$ can type unproductive terms, like Ω). Two possibilities:
- Some terms are typable in System $\mathtt{S}$, but some others are not: in that case, $\mathtt{S}$ will characterize a set of terms wider than the usual known sets of normalizable terms.

TYPABLE TERMS IN $\mathtt{S}$

- **Question:** let us drop approximability. What is the set of typable terms in $\mathtt{S}$?
- We already know that $\mathtt{S}$ is **unsound** ($\mathtt{S}$ can type unproductive terms, like Ω). Two possibilities:
- Some terms are typable in System $\mathtt{S}$, but some others are not: in that case, $\mathtt{S}$ will characterize a set of terms wider than the usual known sets of normalizable terms.
- Every term is typable in $\mathtt{S}$. We say that $\mathtt{S}$ is **completely unsound**. In that case, since $\mathtt{S}$ enjoys SR and SE, $\mathtt{S}$ will provide us with a new model for pure **lambda-calculus**.

We are actually in the second case:

- Every term has a non-empty denotation (including the **mute terms**).
- Terms are discriminated according to their **order** (the maximal number of abs that prefixes a reduct).

We are actually in the second case:

- Every term has a non-empty denotation (including the **mute terms**).
- Terms are discriminated according to their **order** (the maximal number of abs that prefixes a reduct).

Related works

- Jacopini[75]: **easy** terms (t is easy if it can be consistently equated to any other term)
- Berarducci[96]: **mute** terms ("The most undefined terms").
- Bucciarelli,Carraro,Favro,Salibra[15]: *Graph easy Sets of mute lambda terms*, TCS.

Relevance vs Irrelevance

- **Observation:** In system $\mathscr{R}$, $\lambda x.x$ (resp. $\lambda y.x$) can only be typed with a type of the form $[\tau] \to \tau$ (resp. $[\,] \to \tau$).

RELEVANCE VS IRRELEVANCE

- **Observation:** In system $\mathscr{R}$, $\lambda x.x$ (resp. $\lambda y.x$) can only be typed with a type of the form $[\tau] \to \tau$ (resp. $[\,] \to \tau$).
- System $\mathscr{R}$ is said to be **relevant**: *weakening* is not allowed.

Relevance vs Irrelevance

- **Observation:** In system $\mathscr{R}$, $\lambda x.x$ (resp. $\lambda y.x$) can only be typed with a type of the form $[\tau] \to \tau$ (resp. $[\,] \to \tau$).
- System $\mathscr{R}$ is said to be **relevant**: *weakening* is not allowed. For instance, a type is used when it is assigned:

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

RELEVANCE VS IRRELEVANCE

- **Observation:** In system $\mathscr{R}$, $\lambda x.x$ (resp. $\lambda y.x$) can only be typed with a type of the form $[\tau] \to \tau$ (resp. $[\,] \to \tau$).
- System $\mathscr{R}$ is said to be **relevant**: *weakening* is not allowed. For instance, a type is used when it is assigned:

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

- If we replace `ax` by `axw`:

$$\frac{i_0 \in I}{\Gamma;\ x : [\sigma_i]_{i \in I} \vdash x : \sigma_{i_0}}\ \texttt{axw}$$

... we obtain an irrelevant system, called $\mathscr{R}_{\texttt{w}}$.

RELEVANCE VS IRRELEVANCE

- **Observation:** In system $\mathscr{R}$, $\lambda x.x$ (resp. $\lambda y.x$) can only be typed with a type of the form $[\tau] \to \tau$ (resp. $[\,] \to \tau$).
- System $\mathscr{R}$ is said to be **relevant**: *weakening* is not allowed. For instance, a type is used when it is assigned:

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

- If we replace `ax` by `axw`:

$$\frac{i_0 \in I}{\Gamma;\, x : [\sigma_i]_{i \in I} \vdash x : \sigma_{i_0}}\ \texttt{axw}$$

  ... we obtain an irrelevant system, called $\mathscr{R}_{\mathtt{w}}$.

- In $\mathscr{R}_{\mathtt{w}}$, we may derive:

$$\frac{\dfrac{}{x : [\tau, \tau_1, \tau_1] \vdash x : \tau}\ \texttt{axw}}{\vdash \lambda x.x : [\tau, \tau_1, \tau_2] \to \tau}\ \texttt{abs} \qquad \frac{\dfrac{}{x : [\tau], y : [\tau] \vdash x : \tau}\ \texttt{axw}}{x : [\tau] \vdash \lambda y.x : [\tau] \to \tau}\ \texttt{abs}$$

IRRELEVANCY AND COMPLETE UNSOUNDNESS

- We have met the type ρ satisfying $\rho = [\rho]_\omega \to \rho$.

IRRELEVANCY AND COMPLETE UNSOUNDNESS

- We have met the type ρ satisfying $\rho = [\rho]_\omega \to \rho$.
- Due to irrelevancy, every term is typable in $\mathscr{R}_{\mathtt{w}}$ (**complete unsoundness of $\mathscr{R}_{\mathtt{w}}$**).

IRRELEVANCY AND COMPLETE UNSOUNDNESS

- We have met the type ρ satisfying $\rho = [\rho]_\omega \to \rho$.
- Due to irrelevancy, every term is typable in $\mathscr{R}_{\mathtt{w}}$ (**complete unsoundness of $\mathscr{R}_{\mathtt{w}}$**).
- **Claim:** Let t be a term. If $\Gamma(x) = [\rho]_\omega$ for all free variable x of t, then $\Gamma \vdash t : \rho$ is derivable in $\mathscr{R}_{\mathtt{w}}$.

IRRELEVANCY AND COMPLETE UNSOUNDNESS

- We have met the type ρ satisfying $\rho = [\rho]_\omega \to \rho$.
- Due to irrelevancy, every term is typable in $\mathscr{R}_{\mathtt{w}}$ (**complete unsoundness of $\mathscr{R}_{\mathtt{w}}$**).
- **Claim:** Let t be a term. If $\Gamma(x) = [\rho]_\omega$ for all free variable x of t, then $\Gamma \vdash t : \rho$ is derivable in $\mathscr{R}_{\mathtt{w}}$.

  Proof.

$$\dfrac{\Gamma; x : [\rho]_\omega \vdash t : \rho}{\Gamma \vdash \lambda x.t : [\rho]_\omega \to \rho \ \ (= \rho)}\mathtt{abs}$$

$$\dfrac{\Gamma \vdash t : \rho \ \ (= [\rho]_\omega \to \rho) \qquad (\Gamma \vdash u : \rho)_\omega}{\Gamma \vdash t\, u : \rho}\mathtt{app}$$

RELEVANT COINDUCTIVE TYPES

- In $\mathscr{R}$, the typing rules constrain $[\,]$ to appear.
  Failure of the previous argument.

RELEVANT COINDUCTIVE TYPES

- In $\mathscr{R}$, the typing rules constrain $[\,]$ to appear. Failure of the previous argument.

- **Question:** what is the set of typable terms in $\mathscr{R}$?

RELEVANT COINDUCTIVE TYPES

- In $\mathscr{R}$, the typing rules constrain $[\,]$ to appear.
  Failure of the previous argument.

- **Question:** what is the set of typable terms in $\mathscr{R}$?

- Naively, when we meet the subterm $x\,u$ in a term t, we want to type x with an arrow whose domain is the type of u (thus, $x : [\tau_u] \rightarrow \tau$), and proceed by induction.

RELEVANT COINDUCTIVE TYPES

- In $\mathscr{R}$, the typing rules constrain $[\,]$ to appear.
  Failure of the previous argument.

- **Question:** what is the set of typable terms in $\mathscr{R}$?

- Naively, when we meet the subterm $x\,u$ in a term t, we want to type x with an arrow whose domain is the type of u (thus, $x : [\tau_u] \to \tau$), and proceed by induction.

- *Problem:* x may substituted at some point by $\lambda xy.x$ (or another constrained term).

RELEVANT COINDUCTIVE TYPES

- In $\mathscr{R}$, the typing rules constrain $[\,]$ to appear.
  Failure of the previous argument.

- **Question:** what is the set of typable terms in $\mathscr{R}$?

- Naively, when we meet the subterm $x\,u$ in a term t, we want to type x with an arrow whose domain is the type of u (thus, $x : [\tau_u] \rightarrow \tau$), and proceed by induction.
- *Problem:* x may substituted at some point by $\lambda xy.x$ (or another constrained term).
- In that case, the type of x must also be of the form $[\sigma'] \rightarrow [\,] \rightarrow \sigma'$.

RELEVANT COINDUCTIVE TYPES

- In $\mathscr{R}$, the typing rules constrain $[\,]$ to appear.
  Failure of the previous argument.

- **Question:** what is the set of typable terms in $\mathscr{R}$?

- Naively, when we meet the subterm $x\,u$ in a term t, we want to type x with an arrow whose domain is the type of u (thus, $x : [\tau_u] \to \tau$), and proceed by induction.
- *Problem:* x may substituted at some point by $\lambda xy.x$ (or another constrained term).
- In that case, the type of x must also be of the form $[\sigma'] \to [\,] \to \sigma'$.
- Difficulty to see the typing constraints on x.

Question: what is the set of typable terms in $\mathscr{R}$?

Question: what is the set of typable terms in $\mathscr{R}$?

- *In the finite case:* type Normal Forms and proceed by expansion.

Question: what is the set of typable terms in $\mathscr{R}$?

- *In the finite case:* type Normal Forms and proceed by expansion.
- *Problem for coinductive Types:* no form of normalization is granted (*e.g.* Ω typable in $\mathscr{R}$).

Question: what is the set of typable terms in $\mathscr{R}$?

- *In the finite case:* type Normal Forms and proceed by expansion.
- *Problem for coinductive Types:* no form of normalization is granted (*e.g.* Ω typable in $\mathscr{R}$).

We study then **typability** as a first order theory. For that, we will rather study typability in $\mathtt{S}$.

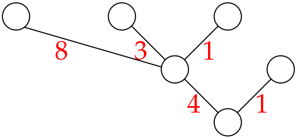
Question: what is the set of typable terms in $\mathscr{R}$?

- *In the finite case:* type Normal Forms and proceed by expansion.
- *Problem for coinductive Types:* no form of normalization is granted (*e.g.* Ω typable in $\mathscr{R}$).

We study then **typability** as a first order theory. For that, we will rather study typability in $\mathtt{S}$.
System $\mathtt{S}$ collapses on $\mathscr{R}$. Thus, if every term is typable in $\mathtt{S}$, then every term is typable in $\mathscr{R}$.

CANDIDATE SUPPORTS

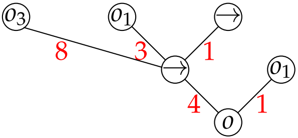
What is a correct type ?



Support:
$\{\varepsilon, 1, 4, 4 \cdot 1, 4 \cdot 3, 4 \cdot 8\}$

CANDIDATE SUPPORTS

What is a correct type ?

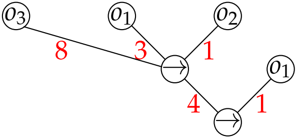


Wrong Labels

Support:
$\{\varepsilon, 1, 4, 4 \cdot 1, 4 \cdot 3, 4 \cdot 8\}$

CANDIDATE SUPPORTS

What is a correct type ?

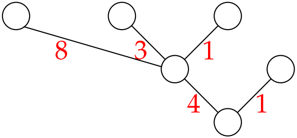


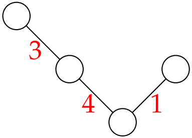
Correct Labels

Support:
$\{\varepsilon, 1, 4, 4 \cdot 1, 4 \cdot 3, 4 \cdot 8\}$

CANDIDATE SUPPORTS

What is a correct type ?



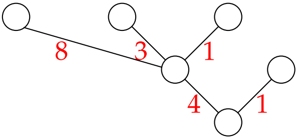


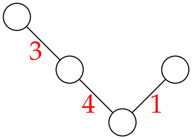
Support:
$\{\varepsilon, 1, 4, 4 \cdot 1, 4 \cdot 3, 4 \cdot 8\}$

Support:
$\{\varepsilon, 1, 4, 4 \cdot 3\}$

CANDIDATE SUPPORTS

What is a correct type ?





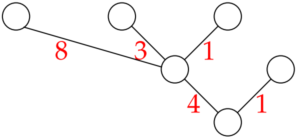
Wrong Support

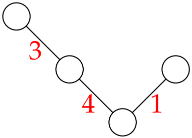
Support:
$\{\varepsilon, 1, 4, 4 \cdot 1, 4 \cdot 3, 4 \cdot 8\}$

Support:
$\{\varepsilon, 1, 4, 4 \cdot 3\}$

CANDIDATE SUPPORTS

What is a correct type ?





Support:
$\{\varepsilon, 1, 4, 4\cdot 1, 4\cdot 3, 4\cdot 8\}$

Support:
$\{\varepsilon, 1, 4, 4\cdot 3\}$

Candidate Support: a set of positions that is the support of a type

- $c \to_{\mathtt{t1}} c\cdot k$ (a candidate supp is a tree)
- $c\cdot 1 \to_{\mathtt{t2}} c\cdot k$ (if a node does not have a 1-son, it is a leaf)

CANDIDATE BISUPPORTS

- We want to show that every term t is typable in $\mathtt{S}$.

CANDIDATE BISUPPORTS

- We want to show that every term t is typable in $\mathtt{S}$.
- *Idea:* we try to capture the notion of **candidate bisupport**: a set of pointers that is the bisupport of a $\mathtt{S}$-derivation typing t.

CANDIDATE BISUPPORTS

- We want to show that every term t is typable in $\mathtt{S}$.
- *Idea:* we try to capture the notion of **candidate bisupport**: a set of pointers that is the bisupport of a $\mathtt{S}$-derivation typing t.
- We must find suitable stability conditions.

CANDIDATE BISUPPORTS

- We want to show that every term t is typable in $\mathtt{S}$.
- *Idea:* we try to capture the notion of **candidate bisupport**: a set of pointers that is the bisupport of a $\mathtt{S}$-derivation typing t.
- We must find suitable stability conditions.
- Then, we show that there is a *non-empty* set that satisfies them.

CANDIDATE BISUPPORTS

- $(a,\, c) \to_{\mathtt{asc}} (a \cdot 1, 1 \cdot c)$ if $t(a) = @$.
- $(a,\, 1 \cdot c) \to (a \cdot 0,\, c)$ if $t(a) = \lambda x$.
- $(a, k \cdot c) \to_{\mathtt{pi}} (pos(k), c)$ if $t(a) = \lambda x$ and $k \in \mathtt{Tr}_1(a)$.
- $(a, k \cdot c) \to_{\mathtt{pi}} \mathtt{b}_\perp$ if $t(\bar{a}) = \lambda x$ and $k \notin \mathtt{Tr}_1(a)$, $k \geqslant 2$.
- $(a \cdot 1, k \cdot c) \xrightarrow{a} (a \cdot k, c)$ if $t(a) = @$.
- $(a, c) \to_{\mathtt{t1}} (a, c \cdot k)$.
- $(a, c \cdot 1) \to_{\mathtt{t2}} (a, c \cdot k)$ for any $k \geqslant 2$.
- $(a, 1) \to_{\mathtt{rt}} (a, \varepsilon)$ if $t(a) = \lambda x$.
- $(a, \varepsilon) \to_{\mathtt{up}} \mathtt{b}_\perp$.
- $(a, \varepsilon) \to_{\mathtt{up}} (a', c)$ if $a \leqslant a'$

CANDIDATE BISUPPORTS

- $(a,\, c) \to_{\texttt{asc}} (a \cdot 1, 1 \cdot c)$ if $t(a) = @$.
- $(a,\, 1 \cdot c) \to (a \cdot 0,\, c)$ if $t(a) = \lambda x$.
- $(a, k \cdot c) \to_{\texttt{pi}} (pos(k), c)$ if $t(a) = \lambda x$ and $k \in \texttt{Tr}_1(a)$.
- $(a, k \cdot c) \to_{\texttt{pi}} \texttt{b}_\perp$ if $t(\bar{a}) = \lambda x$ and $k \notin \texttt{Tr}_1(a)$, $k \geqslant 2$.
- $(a \cdot 1, k \cdot c) \xrightarrow{a} (a \cdot k, c)$ if $t(a) = @$.
- $(a, c) \to_{\texttt{t1}} (a, c \cdot k)$.
- $(a, c \cdot 1) \to_{\texttt{t2}} (a, c \cdot k)$ for any $k \geqslant 2$.
- $(a, 1) \to_{\texttt{rt}} (a, \varepsilon)$ if $t(a) = \lambda x$.
- $(a, \varepsilon) \to_{\texttt{up}} \texttt{b}_\perp$.
- $(a, \varepsilon) \to_{\texttt{up}} (a', c)$ if $a \leqslant a'$

GUIDELINES OF THE PROOF

Goal: checking that the former conditions cannot prove that the type of t must be empty.
In that case, we can build a derivation whose bisupport is minimal.

GUIDELINES OF THE PROOF

Goal: checking that the former conditions cannot prove that the type of t must be empty.
In that case, we can build a derivation whose bisupport is minimal.

- *Ad absurbum*, we consider $\mathscr{P}$, a proof showing that the type of t is empty (a "**bad proof**").

GUIDELINES OF THE PROOF

Goal: checking that the former conditions cannot prove that the type of t must be empty.
In that case, we can build a derivation whose bisupport is minimal.

- *Ad absurbum*, we consider $\mathscr{P}$, a proof showing that the type of t is empty (a "**bad proof**").
- The presence of redex is still problematic. A finite reduction strategy (the **collapsing strategy**) allows us to reduce $\mathscr{P}$ to a proof $\mathscr{P}'$, in which redexes are not a problem.

GUIDELINES OF THE PROOF

Goal: checking that the former conditions cannot prove that the type of t must be empty.
In that case, we can build a derivation whose bisupport is minimal.

- *Ad absurbum*, we consider $\mathscr{P}$, a proof showing that the type of t is empty (a "**bad proof**").
- The presence of redex is still problematic. A finite reduction strategy (the **collapsing strategy**) allows us to reduce $\mathscr{P}$ to a proof $\mathscr{P}'$, in which redexes are not a problem.
- In $\mathscr{P}'$, commutations and nice interactions occur. Considering a minimal case, we show that $\mathscr{P}'$ *cannot* prove that t has an empty type. *Contradiction*.

GUIDELINES OF THE PROOF

Goal: checking that the former conditions cannot prove that the type of t must be empty.
In that case, we can build a derivation whose bisupport is minimal.

- *Ad absurbum*, we consider $\mathscr{P}$, a proof showing that the type of t is empty (a "**bad proof**").
- The presence of redex is still problematic. A finite reduction strategy (the **collapsing strategy**) allows us to reduce $\mathscr{P}$ to a proof $\mathscr{P}'$, in which redexes are not a problem.
- In $\mathscr{P}'$, commutations and nice interactions occur. Considering a minimal case, we show that $\mathscr{P}'$ *cannot* prove that t has an empty type. *Contradiction*.

This works for the infinitary λ-calculus.

ORDER

Theorem (complete unsoundness): in $\mathscr{R}$, every term is typable.

ORDER

Theorem (complete unsoundness): in $\mathscr{R}$, every term is typable.

Definition: The **order** of a λ-term t is the maximal $n \in \mathbb{N} \cup \{\infty\}$ s.t.
$t \to^* t' = \lambda x_1 \ldots \lambda x_n . t'_0$.
A **zero term** is a term of order 0.

ORDER

Theorem (complete unsoundness): in $\mathscr{R}$, every term is typable.

Definition: The **order** of a λ-term t is the maximal $n \in \mathbb{N} \cup \{\infty\}$ s.t. $t \to^* t' = \lambda x_1 \ldots \lambda x_n.t'_0$.
A **zero term** is a term of order 0.

Proposition: if t is a zero-term, then, t is typable with o.

ORDER

Theorem (complete unsoundness): in $\mathscr{R}$, every term is typable.

Definition: The **order** of a λ-term t is the maximal $n \in \mathbb{N} \cup \{\infty\}$ s.t. $t \to^* t' = \lambda x_1 \dots \lambda x_n . t'_0$.
A **zero term** is a term of order 0.

Proposition: if t is a zero-term, then, t is typable with o.

Definition (relational model): For all closed λ-term t, we set

$$[\![t]\!] = \{\tau \mid \vdash t : \tau \text{ is derivable}\}$$

ORDER

Theorem (complete unsoundness): in $\mathscr{R}$, every term is typable.

Definition: The **order** of a λ-term t is the maximal $n \in \mathbb{N} \cup \{\infty\}$ s.t. $t \rightarrow^* t' = \lambda x_1 \ldots \lambda x_n . t'_0$.
A **zero term** is a term of order 0.

Proposition: if t is a zero-term, then, t is typable with o.

Definition (relational model): For all closed λ-term t, we set

$$[\![t]\!] = \{\tau \mid \vdash t : \tau \text{ is derivable}\}$$

Theorem: This yields a non-sensible model that discriminates terms according to their order.

Plan

THE PROBLEM OF COLLAPSE

- **Question:** Any derivation of $\mathtt{S}$ collapses on a derivation of $\mathscr{R}$. Is this collapse surjective ? Is every derivation of $\mathscr{R}$ the collapse of a derivation of $\mathtt{S}$?

THE PROBLEM OF COLLAPSE

- **Question:** Any derivation of `S` collapses on a derivation of $\mathscr{R}$. Is this collapse surjective ? Is every derivation of $\mathscr{R}$ the collapse of a derivation of `S`?
- The `app`-rule can be restated as follows:

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

THE PROBLEM OF COLLAPSE

- **Question:** Any derivation of $\mathtt{S}$ collapses on a derivation of $\mathscr{R}$. Is this collapse surjective ? Is every derivation of $\mathscr{R}$ the collapse of a derivation of $\mathtt{S}$?
- The `app`-rule can be restated as follows:

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

- Thus, the choice of types in axiom rules must ensure that we have a **syntactic equality** for every `app`-rule.

The problem of collapse

- **Question:** Any derivation of `S` collapses on a derivation of $\mathscr{R}$. Is this collapse surjective ? Is every derivation of $\mathscr{R}$ the collapse of a derivation of `S`?
- The `app`-rule can be restated as follows:

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

- Thus, the choice of types in axiom rules must ensure that we have a **syntactic equality** for every `app`-rule.
- Moreover, we must avoid track conflict in the contexts.

HYBRID DERIVATIONS

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

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

where $(S_k)_{k\in K} \equiv (S'_k)_{k'\in K'}$ means that $(S_k)_{k\in K}$ and $(S'_k)_{k'\in K'}$ collapse on the same type of $\mathscr{R}$.

HYBRID DERIVATIONS

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

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

  where $(S_k)_{k \in K} \equiv (S'_k)_{k' \in K'}$ means that $(S_k)_{k \in K}$ and $(S'_k)_{k' \in K'}$ collapse on the same type of $\mathscr{R}$.

- Easy to show that every $\mathscr{R}$-derivation is the collapse of a $\mathtt{S_h}$-derivation.

Operable Derivation

- Let P be a hybrid derivation typing t.

OPERABLE DERIVATION

- Let P be a hybrid derivation typing t.
  - If a is the position of a judgment typing a redex $(\lambda x.r)s$ inside t, a **root isomorphism** $\rho_a : (S_k)_{k \in K}(a) \to (S'_k)_{k \in K'}(a)$ tells us how to perform subject reduction.

OPERABLE DERIVATION

- Let P be a hybrid derivation typing t.
  - If a is the position of a judgment typing a redex $(\lambda x.r)s$ inside t, a **root isomorphism** $\rho_a : (S_k)_{k \in K}(a) \to (S'_k)_{k \in K'}(a)$ 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 is the position of a judgment typing a redex $(\lambda x.r)s$ inside t, a **root isomorphism** $\rho_a : (S_k)_{k \in K}(a) \to (S'_k)_{k \in K'}(a)$ 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 is the position of a judgment typing a redex $(\lambda x.r)s$ inside t, a **root isomorphism** $\rho_a : (S_k)_{k\in K}(a) \to (S'_k)_{k\in K'}(a)$ 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 **sequence type isomorphism** $\phi_a : (S_k)_{k\in K}(a) \to (S'_k)_{k\in K'}(a)$

OPERABLE DERIVATION

- Let P be a hybrid derivation typing t.
  - If a is the position of a judgment typing a redex $(\lambda x.r)s$ inside t, a **root isomorphism** $\rho_a : (S_k)_{k \in K}(a) \to (S'_k)_{k \in K'}(a)$ 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 **sequence type isomorphism** $\phi_a : (S_k)_{k \in K}(a) \to (S'_k)_{k \in K'}(a)$
  - A **complete interface** is given by a family of full sequence type isomorphisms $\phi_a : (S_k)_{k \in K}(a) \to (S'_k)_{k \in K'}(a)$ when a ranges over the `app`-nodes of P.

OPERABLE DERIVATION

- Let P be a hybrid derivation typing t.
  - If a is the position of a judgment typing a redex $(\lambda x.r)s$ inside t, a **root isomorphism** $\rho_a : (S_k)_{k\in K}(a) \to (S'_k)_{k\in K'}(a)$ 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 **sequence type isomorphism** $\phi_a : (S_k)_{k\in K}(a) \to (S'_k)_{k\in K'}(a)$
  - A **complete interface** is given by a family of full sequence type isomorphisms $\phi_a : (S_k)_{k\in K}(a) \to (S'_k)_{k\in K'}(a)$ 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 a.

OPERABLE DERIVATION

- Let P be a hybrid derivation typing t.
  - If a is the position of a judgment typing a redex $(\lambda x.r)s$ inside t, a **root isomorphism** $\rho_a : (S_k)_{k \in K}(a) \to (S'_k)_{k \in K'}(a)$ 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 **sequence type isomorphism** $\phi_a : (S_k)_{k \in K}(a) \to (S'_k)_{k \in K'}(a)$
  - A **complete interface** is given by a family of full sequence type isomorphisms $\phi_a : (S_k)_{k \in K}(a) \to (S'_k)_{k \in K'}(a)$ 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 a.
- An **operable derivation** is a hybrid derivation endowed with a complete interface (for each `app`-rule).

REPRESENTATION LEMMA

Lemma

Let Π a $\mathscr{R}$-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 $\mathscr{R}$-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 $\mathscr{R}$-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 $t_0 \xrightarrow{b_0} t_1 \xrightarrow{b_1} t_2 \xrightarrow{b_2} \dots$.
- 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 $\mathscr{R}$-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 $\mathscr{R}$-derivation Π, there is a trivial $\mathtt{S}$-derivation P that collapses into Π.

RESTATEMENT

Theorem:
For all $\mathscr{R}$-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 $\mathscr{R}$-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 $\mathscr{R}$-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 $\mathtt{supp}(P_1)$ to $\mathtt{supp}(P_2)$.

RESTATEMENT

Theorem:
For all $\mathscr{R}$-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 $\mathtt{supp}(P_1)$ to $\mathtt{supp}(P_2)$.
- Between each associated axioms rules of P_1 and P_2, a type isomorphism (w.r.t. the former bijection).

RESTATEMENT

Theorem:
For all $\mathscr{R}$-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 $\mathtt{supp}(P_1)$ to $\mathtt{supp}(P_2)$.
- 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.

RELATED AND FUTURE WORK

- Quantitative types for $\lambda\mu$ (ongoing work with Delia Kesner) and an explicit classical calculus.
- Can infinitary Strong Normalization be characterized ?
- *Categorical Adaptation* of this framework (ongoing work with D. Mazza and L. Pellisier).
- Equational theory of the Model.
- Is the collapse of $\mathscr{R}$ onto $\mathscr{D}$ (idempotent intersection) also surjective ?

QUESTIONS

Thank you for your attention !

ASCENDANCE

Some bipositions can be intuitively identified in a derivation.

ASCENDANCE

Some bipositions can be intuitively identified in a derivation.

$$\frac{C \vdash t : (S_k)_{k \in K} \to T \quad \text{(pos. } a \cdot 1) \qquad (D_k \vdash u : S_k \text{ (pos. } a \cdot k))_{k \in K}}{C \cup_{k \in K} D_k \vdash tu : T \quad \text{(pos. } a)}$$

ASCENDANCE

Some bipositions can be intuitively identified in a derivation.

$$\frac{C \vdash t : (S_k)_{k\in K} \to T \quad \text{(pos. } a\cdot 1\text{)} \qquad (D_k \vdash u : S_k \text{ (pos. } a\cdot k\text{)})_{k\in K}}{C \cup_{k\in K} D_k \vdash tu : T \quad \text{(pos. } a\text{)}}$$

ASCENDANCE

Some bipositions can be intuitively identified in a derivation.

$$\dfrac{C \vdash t : (S_k)_{k\in K} \to T \quad \text{(pos. } a \cdot 1) \qquad (D_k \vdash u : S_k \text{ (pos. } a \cdot k))_{k \in K}}{C \cup_{k\in K} D_k \vdash tu : T \quad \text{(pos. } a)}$$

Two occurrences of the same type

ASCENDANCE

Some bipositions can be intuitively identified in a derivation.

$$\frac{C \vdash t : (S_k)_{k\in K} \to T \quad \text{(pos. } a \cdot 1) \qquad (D_k \vdash u : S_k \text{ (pos. } a \cdot k))_{k\in K}}{C \cup_{k\in K} D_k \vdash tu : T \quad \text{(pos. } a)}$$

ASCENDANCE

Some bipositions can be intuitively identified in a derivation.

$$\frac{C \vdash t : (S_k)_{k \in K} \to T \quad (\text{pos. } a \cdot 1) \qquad (D_k \vdash u : S_k \ (\text{pos. } a \cdot k)\ )_{k \in K}}{C \cup_{k \in K} D_k \vdash tu : T \quad (\text{pos. } a)}$$

Nested position c here
corresponds to...

ASCENDANCE

Some bipositions can be intuitively identified in a derivation.

nested position $1 \cdot c$ there.

$$\frac{C \vdash t : (S_k)_{k \in K} \to T \quad \text{(pos. } a \cdot 1) \qquad (D_k \vdash u : S_k \text{ (pos. } a \cdot k) \,)_{k \in K}}{C \cup_{k \in K} D_k \vdash tu : T \quad \text{(pos. } a)}$$

Nested position c here
corresponds to…

ASCENDANCE

Some bipositions can be intuitively identified in a derivation.

nested position $1 \cdot c$ there.

$$\frac{C \vdash t : (S_k)_{k \in K} \to T \quad \text{(pos. } a \cdot 1\text{)} \qquad (D_k \vdash u : S_k \text{ (pos. } a \cdot k\text{)})_{k \in K}}{C \cup_{k \in K} D_k \vdash tu : T \quad \text{(pos. } a\text{)}}$$

Nested position c here corresponds to…

We then set:
$(a, c) \to_{\mathtt{asc}} (a \cdot 1, 1 \cdot c)$ when $t(a) = @$

ASCENDANCE

Some bipositions can be intuitively identified in a derivation.

ASCENDANCE

Some bipositions can be intuitively identified in a derivation.

$$\frac{C;\, x : (S_k)_{k \in K} \vdash t : T \qquad \text{(pos. } a \cdot 0\text{)}}{C \vdash \lambda x.t : (S_k)_{k \in K} \to T \qquad \text{(pos. } a\text{)}}$$

ASCENDANCE

Some bipositions can be intuitively identified in a derivation.

$$\frac{C;\, x : (S_k)_{k \in K} \vdash t : T \qquad (\text{pos. } a \cdot 0)}{C \vdash \lambda x.t : (S_k)_{k \in K} \to T \qquad (\text{pos. } a)}$$

We then set:
$(a,\, 1 \cdot c) \to_{\texttt{asc}} (a \cdot 0, 1 \cdot c)$ when $t(a) = \lambda x$

POLAR INVERSION

Let us remind rules `ax` and `abs`:

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

POLAR INVERSION

Let us remind rules `ax` and `abs`:

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

In a derivation:

$$\frac{C; x : (S_k)_{k \in K} \vdash t : T \quad \text{(pos. } a \cdot 0\text{)}}{C \vdash \lambda x.t : (S_k)_{k \in K} \to T \quad \text{(pos. } a\text{)}}$$

POLAR INVERSION

Let us remind rules ax and abs:

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

Let $k \geqslant 2$. We have two cases :

$$\frac{C; x : (S_k)_{k \in K} \vdash t : T \qquad \text{(pos. } a \cdot 0\text{)}}{C \vdash \lambda x.t : (S_k)_{k \in K} \to T \qquad \text{(pos. } a\text{)}}$$

Look at S_7
inside this seq. type.

POLAR INVERSION

Let us remind rules `ax` and `abs`:

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

Let $k \geqslant 2$. We have two cases :

- First case :

$$\frac{}{x:\ 7\cdot S_7\vdash x:\ S_7\ \text{(pos. } a'\text{)}}$$

$$\frac{C;\ x:(S_k)_{k\in K}\vdash t:T \qquad \text{(pos. } a\cdot 0\text{)}}{C\vdash \lambda x.t:(S_k)_{k\in K}\to T \qquad \text{(pos. } a\text{)}}$$

Look at S_7
inside this seq. type.

POLAR INVERSION

Let us remind rules `ax` and `abs`:

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

Let $k \geqslant 2$. We have two cases :

- First case :

$$\frac{}{x : 7 \cdot S_7 \vdash x : S_7 \text{ (pos. } a'\text{)}}$$

$$\frac{C; x : (S_k)_{k \in K} \vdash t : T \quad \text{(pos. } a \cdot 0\text{)}}{C \vdash \lambda x.t : (S_k)_{k \in K} \to T \quad \text{(pos. } a\text{)}}$$

Look at S_7
inside this seq. type.

We then set: $(a, 7 \cdot c) \to_{\texttt{pi}} (a', c)$ when $t(a) = \lambda x$

POLAR INVERSION

Let us remind rules `ax` and `abs`:

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

Let $k \geqslant 2$. We have two cases :

Second case :

$$\frac{C; x : (S_k)_{k \in K} \vdash t : T \qquad (\text{pos. } a \cdot 0)}{C \vdash \lambda x.t : (S_k)_{k \in K} \to T \qquad (\text{pos. } a)}$$

Look at S_7
inside this seq. type.

POLAR INVERSION

Let us remind rules `ax` and `abs`:

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

Let $k \geqslant 2$. We have two cases :

Second case :

No `ax`-rule typing x with track 7.

$$\frac{C;\ x:(S_k)_{k\in K}\vdash t:T \qquad \text{(pos. } a\cdot 0)}{C\vdash \lambda x.t:(S_k)_{k\in K}\to T \qquad \text{(pos. } a)}$$

Look at S_7
inside this seq. type.

POLAR INVERSION

Let us remind rules `ax` and `abs`:

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

Let $k \geqslant 2$. We have two cases :

No `ax`-rule typing x with track 7.

$$\frac{C; x : (S_k)_{k \in K} \vdash t : T \qquad \text{(pos. } a \cdot 0\text{)}}{C \vdash \lambda x.t : (S_k)_{k \in K} \to T \qquad \text{(pos. } a\text{)}}$$

Look at S_7
inside this seq. type.

We then set: $(a, 7 \cdot c) \to_{\texttt{pi}} \texttt{b}_\perp$ when $t(a) = \lambda x$