Tracking Redexes in the lambda-calculus

jean-jacques.levy@inria.fr
Irif

24 oct 2024

http://jeanjacqueslevy.net/talks/24track-rev/track.pdf

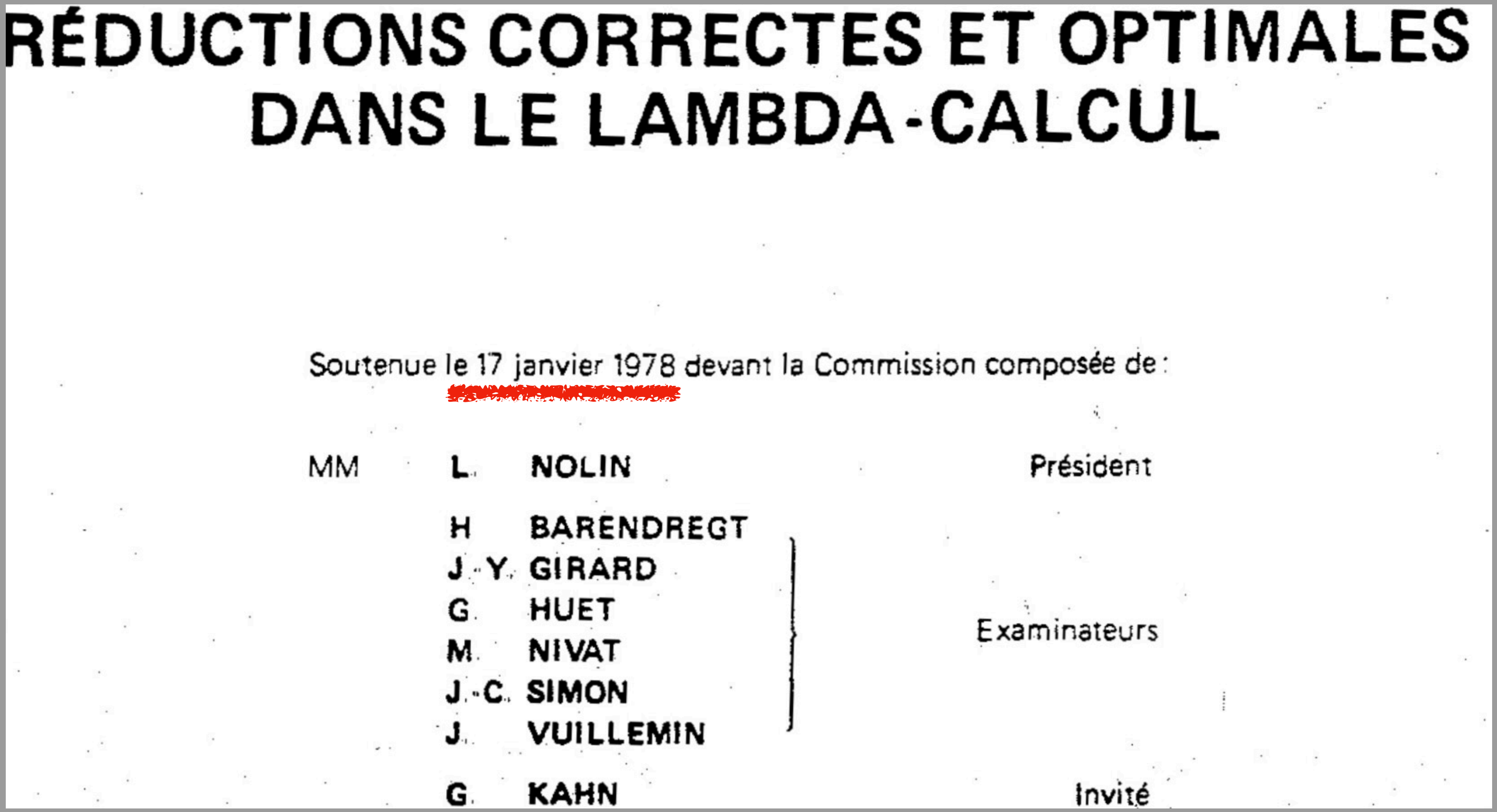
The lambda-calculus

- for logicians: important tool for proof theory
- for computer scientists: kernel of functional programming

```
Tracking redexes in the lambda-calculus
"The French School of Programming"
ed. Bertrand Meyer, Springer, 2024.
```

The lambda-calculus

- for logicians: important tool for proof theory
- for computer scientists: kernel of functional programming



RÉDUCTIONS CORRECTES ET OPTIMALES DANS LE LAMBDA-CALCUL

Soutenue le 17 janvier 1978 devant la Commission composée de :

MM **L. NOLIN** Président

H BARENDREGT
J.-Y. GIRARD
G. HUET
M. NIVAT Examinateurs
J.-C. SIMON
J. VUILLEMIN

G. KAHN Invité

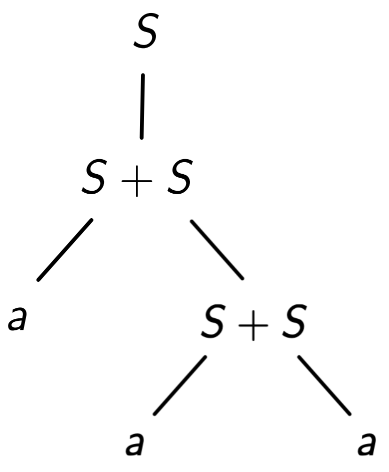
```
Tracking redexes in the lambda-calculus
"The French School of Programming"
ed. Bertrand Meyer, Springer, 2024.
```

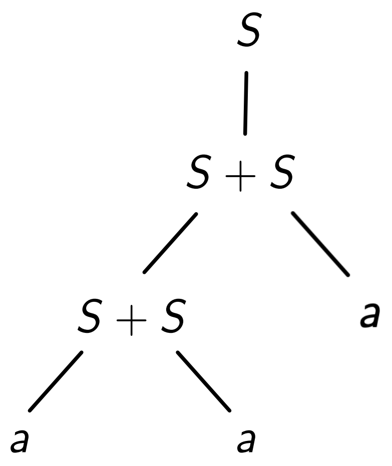
Context-free languages

- ambiguous grammar

$$S \rightarrow S + S$$
$$S \rightarrow a$$

- 2 distinct parse trees for $a + a + a$





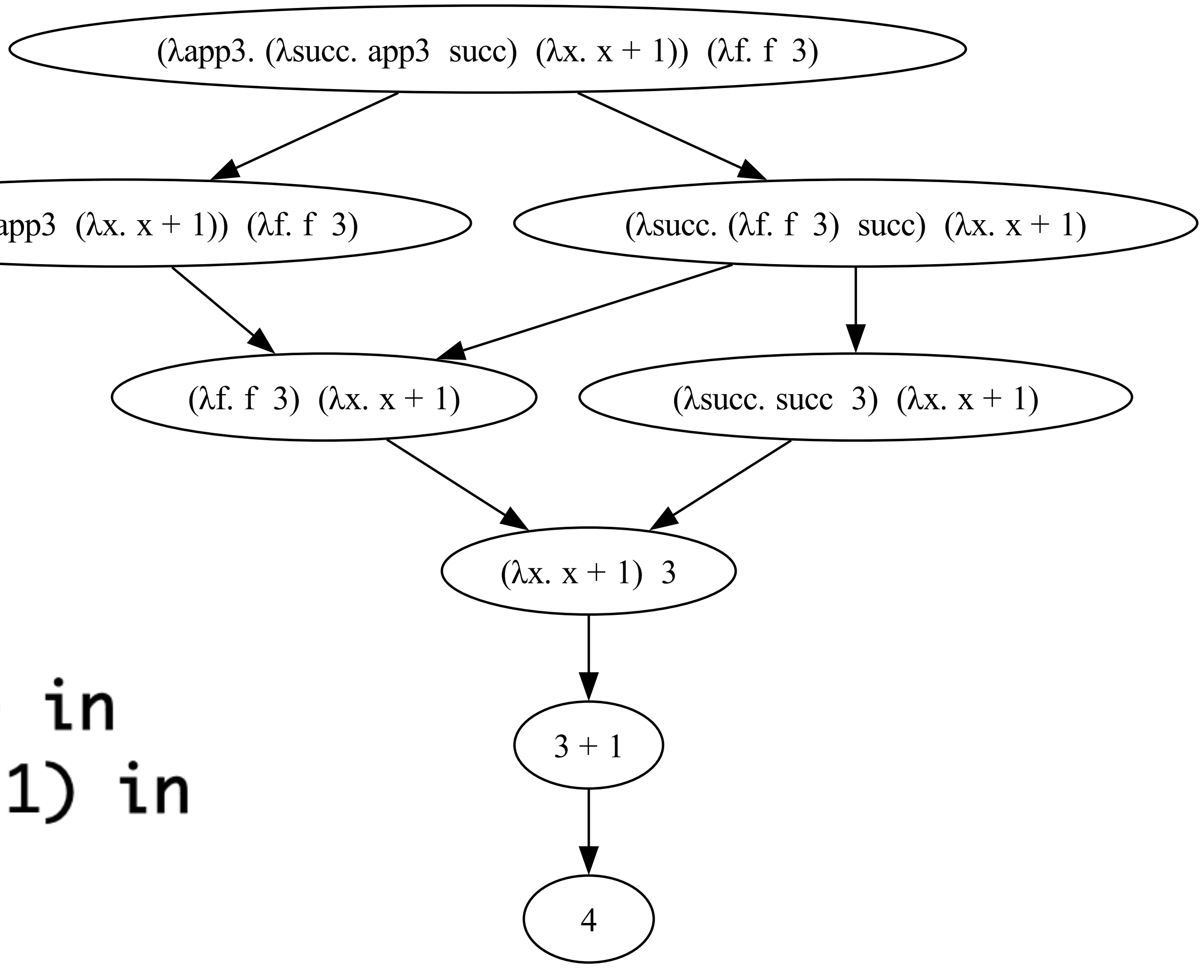
- a parse tree represents a **permutation equivalence** class w.r.t. to derivations

$$S \rightarrow S + S \rightarrow a + S \rightarrow a + S + S \rightarrow a + a + S \rightarrow a + a + a$$
$$S \rightarrow S + S \rightarrow a + S \rightarrow a + S + S \rightarrow a + S + a \rightarrow a + a + a$$
$$S \rightarrow S + S \rightarrow S + S + S \rightarrow a + S + S \rightarrow a + a + S \rightarrow a + a + a$$
$$S \rightarrow S + S \rightarrow S + S + S \rightarrow S + a + S \rightarrow a + a + S \rightarrow a + a + a$$
$\ldots$

- only 1 leftmost derivation per parse tree

The lambda-calculus

- calculus of functions with nested redexes and bound variables

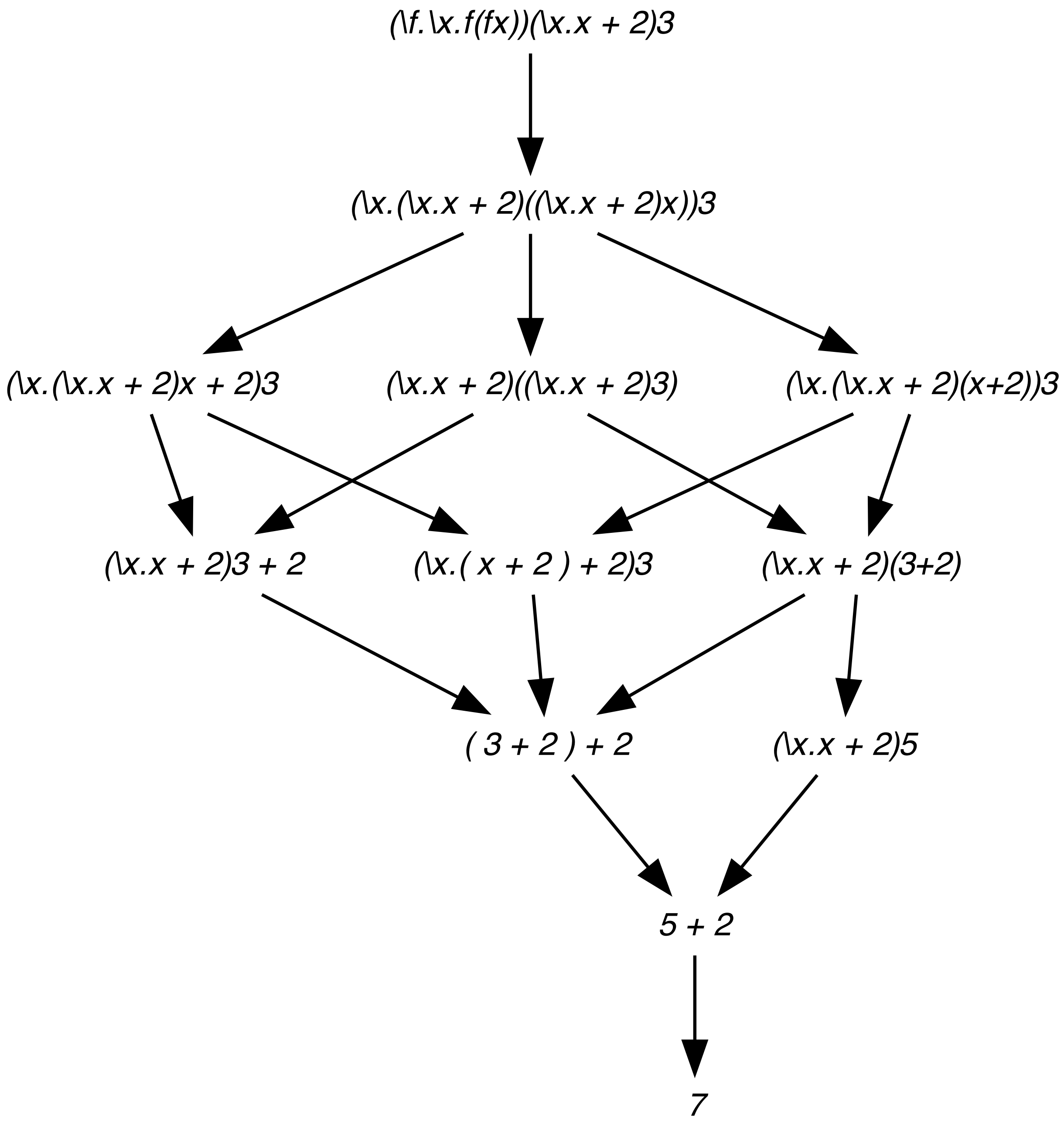


```
let app3 = (fun f => f 3) in
let succ = (fun x => x + 1) in
app3 succ
```

The lambda-calculus

- many permutations of redex contractions

$$(\lambda f.\lambda x.f(f\,x))(\lambda x.x+2)3 \longrightarrow \ldots$$



The lambda-calculus

- local confluence

$$\begin{tikzcd} & {} \arrow[dl, "R"'] \arrow[dr, "S"] & \\ {} \arrow[dr, dashed, "S/R"'] & & {} \arrow[dl, dashed, "R/S"] \\ & {} & \end{tikzcd}$$

- let *R* and *S* be two occurences of redexes in a given term.

Then the **residuals** of *R* by *S* is the set R/S of occurences of (disjoint) redexes which remain of *R* when *S* is contracted.

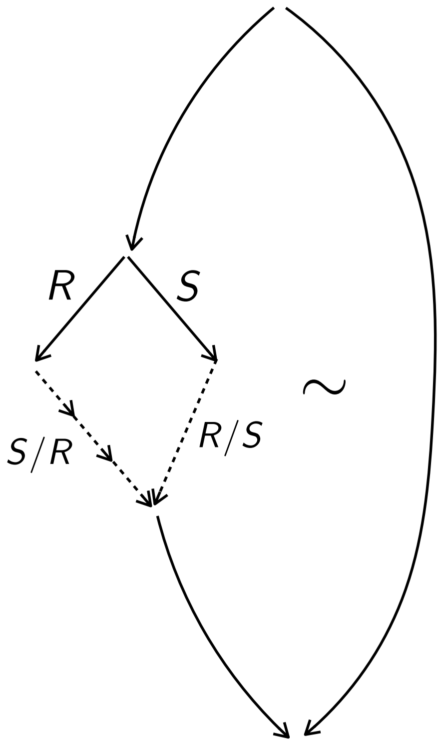
- residuals of (underlined) redexes

$$\begin{array}{ll} \underline{(I\,x)}\,(I\,x) \;\rightarrow\; \underline{(I\,x)}\,x & \qquad I\,x\,\underline{(I\,x)} \;\rightarrow\; x\,\underline{(I\,x)} \\ \Delta\,\underline{(I\,x)} \;\rightarrow\; \underline{(I\,x)}\,\underline{(I\,x)} & \qquad (\lambda x.I\,x\,\underline{(I\,x)})\,y \;\rightarrow\; I\,y\,\underline{(I\,y)} \\ \underline{\Delta\,(I\,x)} \;\rightarrow\; \underline{(\Delta\,x)} & \qquad \underline{(I\,x)} \;\rightarrow\; x \end{array}$$

where $\Delta = \lambda x.x\,x, \quad I = \lambda x.x$

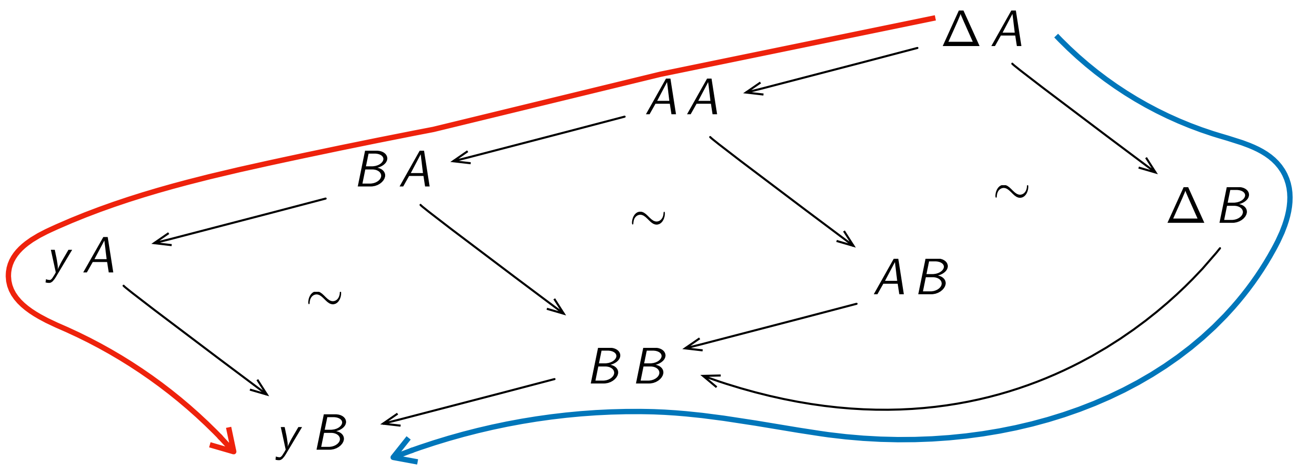
Equivalence by permutations

- 2 reductions are permutation equivalent by iterating the local confluence diagram



Equivalence by permutations

- example



$\Delta = \lambda x.x\,x$

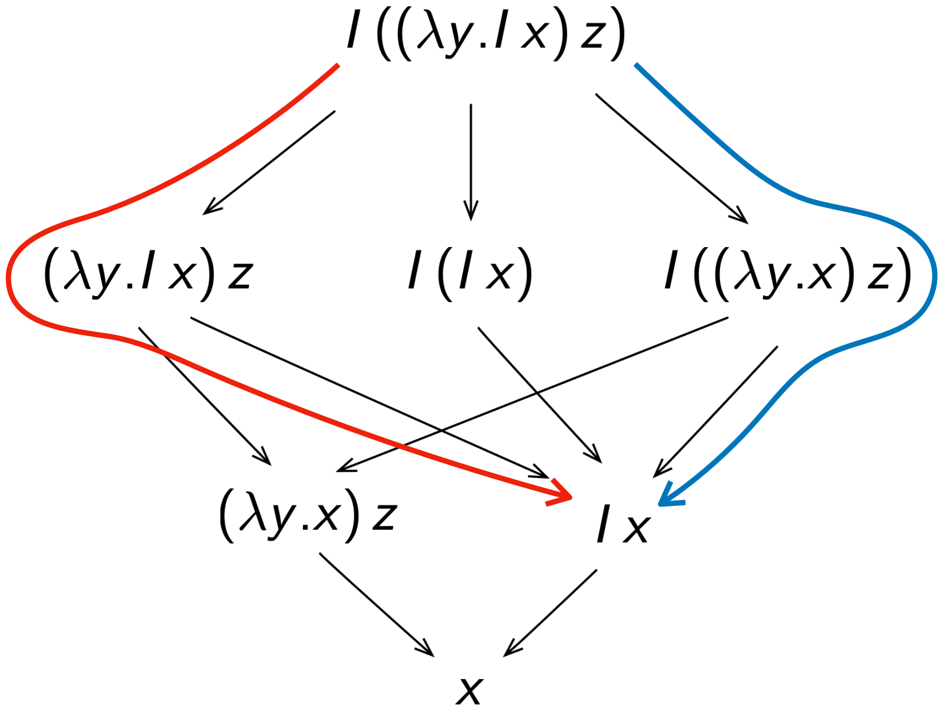
$F = \lambda f.f\,y$

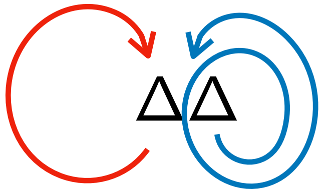
$I = \lambda x.x$

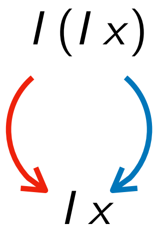
$A = F\,I$

$B = I\,y$

- counter-examples



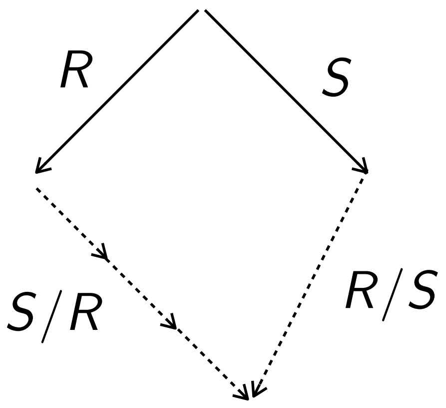




permutation equivalence

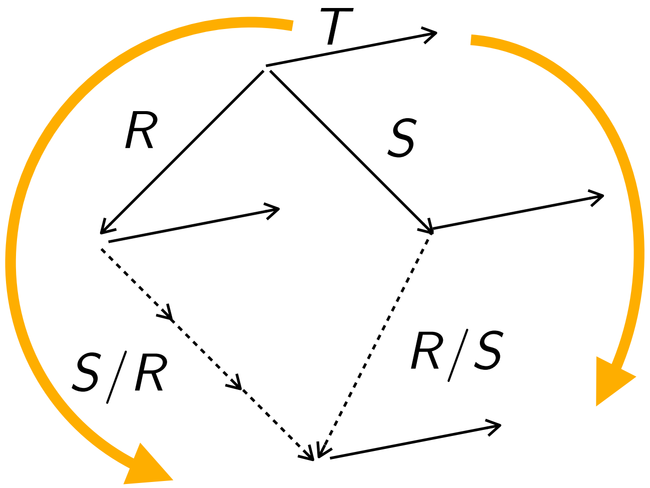
Permutation equivalence

- single-redex reduction steps $M \xrightarrow{R} N$
- residuals S/R of another redex S in M are **disjoint** redexes
  - let $\mathcal{F}$ be a set of disjoint redexes
  - write $\rho : \mathcal{F}$ for any single-redex reduction ρ of redexes of $\mathcal{F}$ in any order
  - these reductions are all cofinal (end on a same term)
- single-redex reductions are locally confluent



Permutation equivalence

- moreover, let T be a another redex in M
- residuals of T on both sides of the permutation are the **same**



$$T/(R \sqcup S) = T/(S \sqcup R)$$

the minicube lemma

$$T/(R\,;(S/R)) = T/(S;(R/S))$$

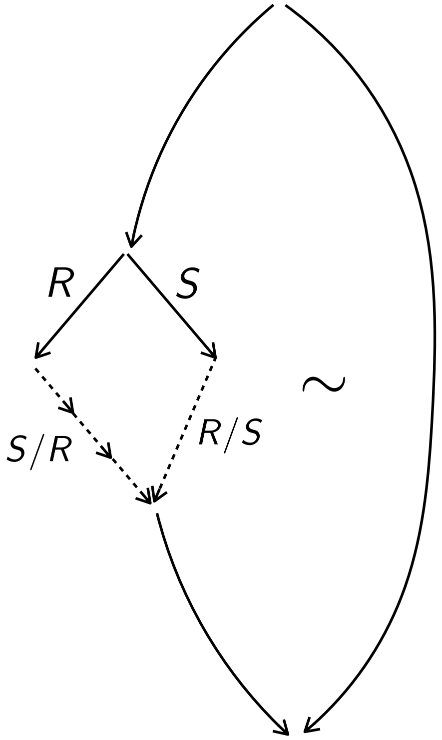
Permutation equivalence

- definition with permutations

$\sim$ is the smallest equivalence relation such that:

$$(i) \quad R \sqcup S \;\sim\; S \sqcup R$$

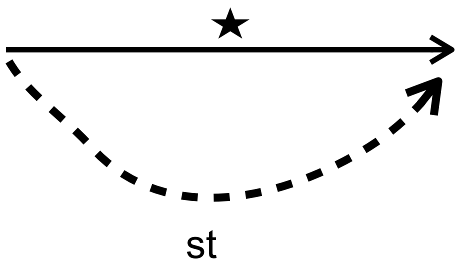
$$(ii) \quad \rho \sim \sigma \implies \tau\,;\rho\,;\upsilon \sim \tau\,;\sigma\,;\upsilon$$



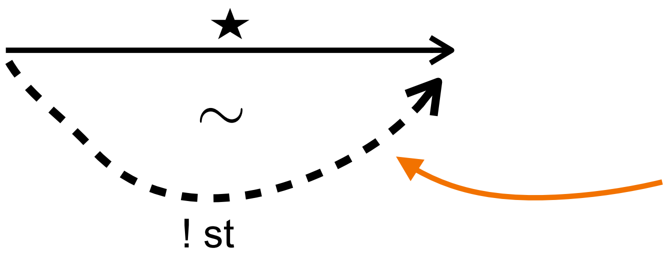
aka parse trees for contex-free languages

Permutation equivalence

- a **standard** reduction is an outside-in left-to-right reduction strategy
- any reduction is equivalent by permutations to a unique standard reduction



λ-calculus



unique

λ-calculus with permutations

- standard reductions are canonical representatives in equivalence classes

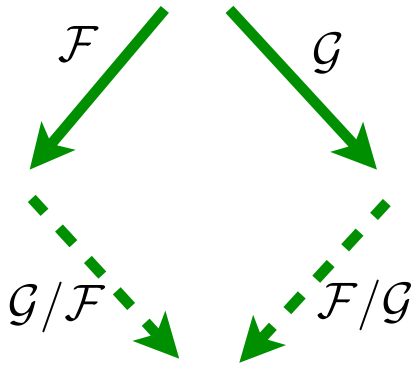
Finite developments

- parallel reduction steps $M \xrightarrow{\mathcal{F}} N$ where $\mathcal{F}$ is a set of redexes in a given term

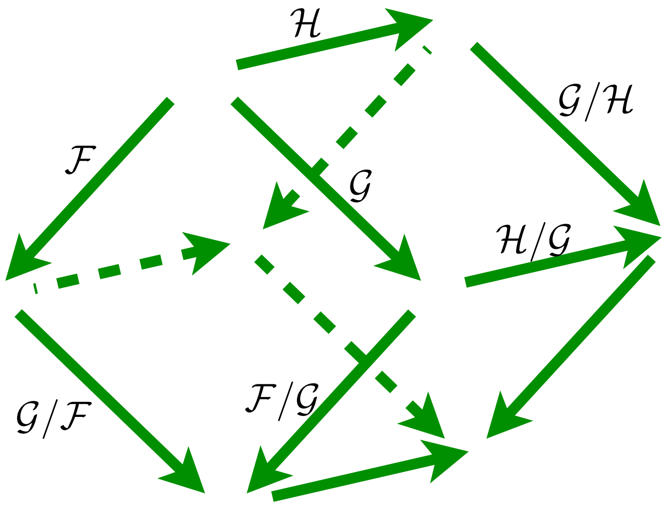
finite development thm `[Curry]` let $\mathcal{F}$ be a set of redexes in a term M

- all reductions contracting only residuals of $\mathcal{F}$ have finite length
- these reductions are all cofinal (end on a same term N)
- these maximal reductions are named developments of $\mathcal{F}$
- write $\rho : M \xrightarrow{\mathcal{F}} N$ or simply $\rho : \mathcal{F}$ for any development ρ of $\mathcal{F}$
- residuals of any redex R in initial term are the same for all developments of $\mathcal{F}$

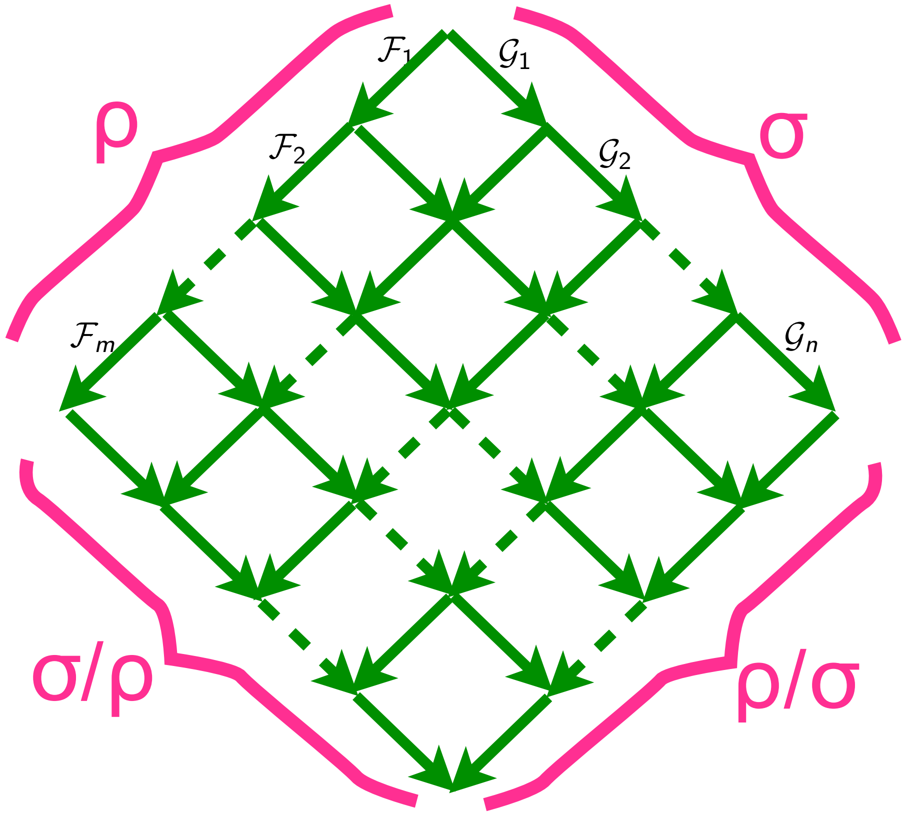
corollary: parallel moves `[Curry]`

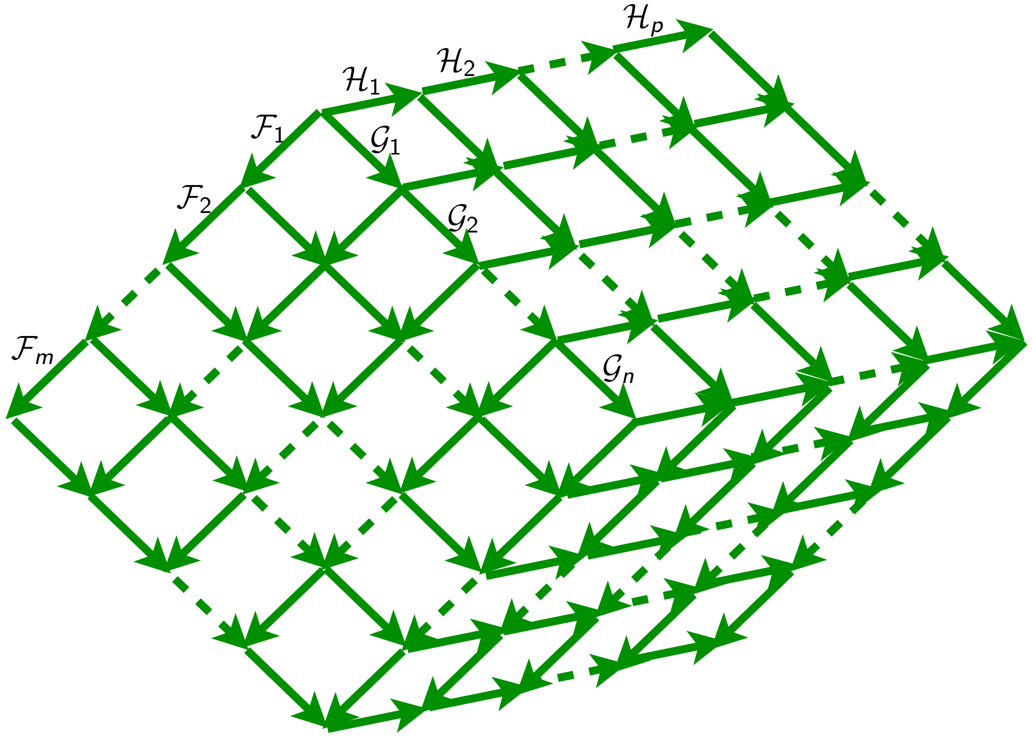


cube lemma



Residual of reductions





parallel moves +

$\rho \sqcup \sigma$ and $\sigma \sqcup \rho$ are cofinal

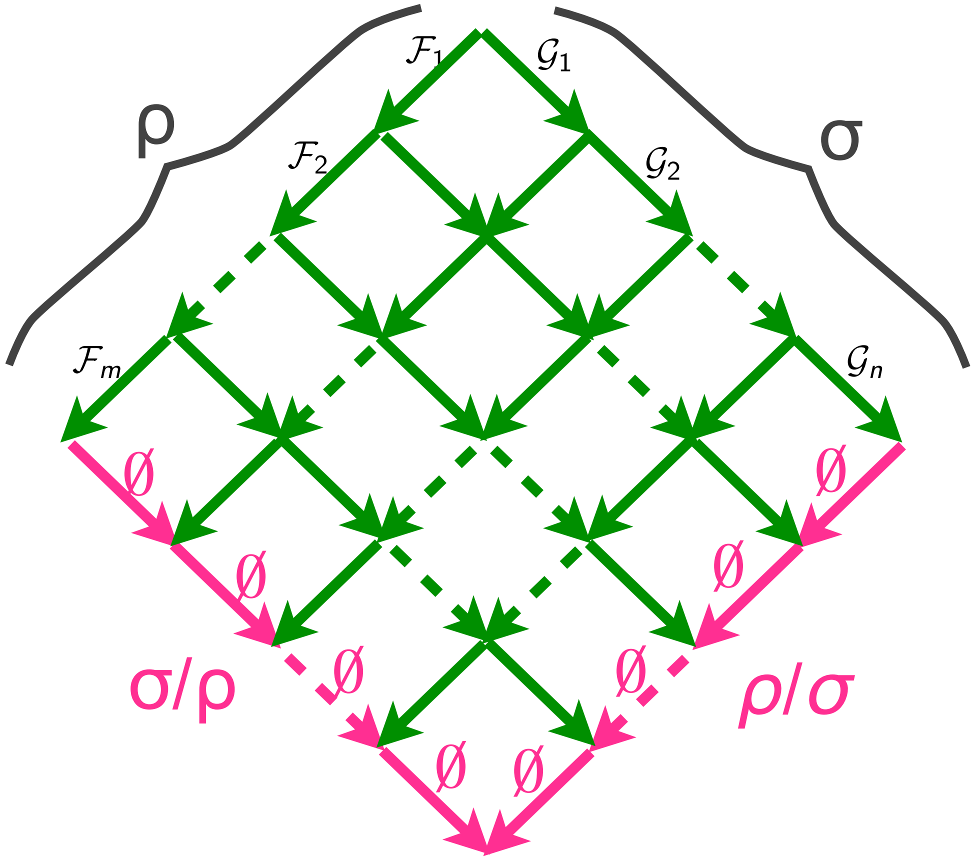
cube lemma +

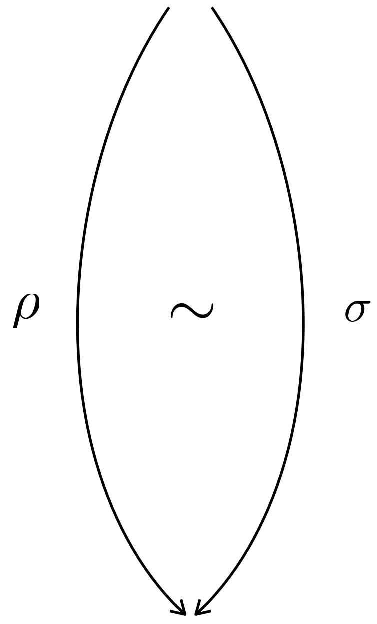
$\tau/(\rho \sqcup \sigma) = \tau/(\sigma \sqcup \rho)$

Permutation equivalence

permutation equivalence (revisited) Let ρ and σ be coinitial reductions.

Then $\rho \sim \sigma$ iff $\rho/\sigma = \emptyset^m$ and $\sigma/\rho = \emptyset^n$



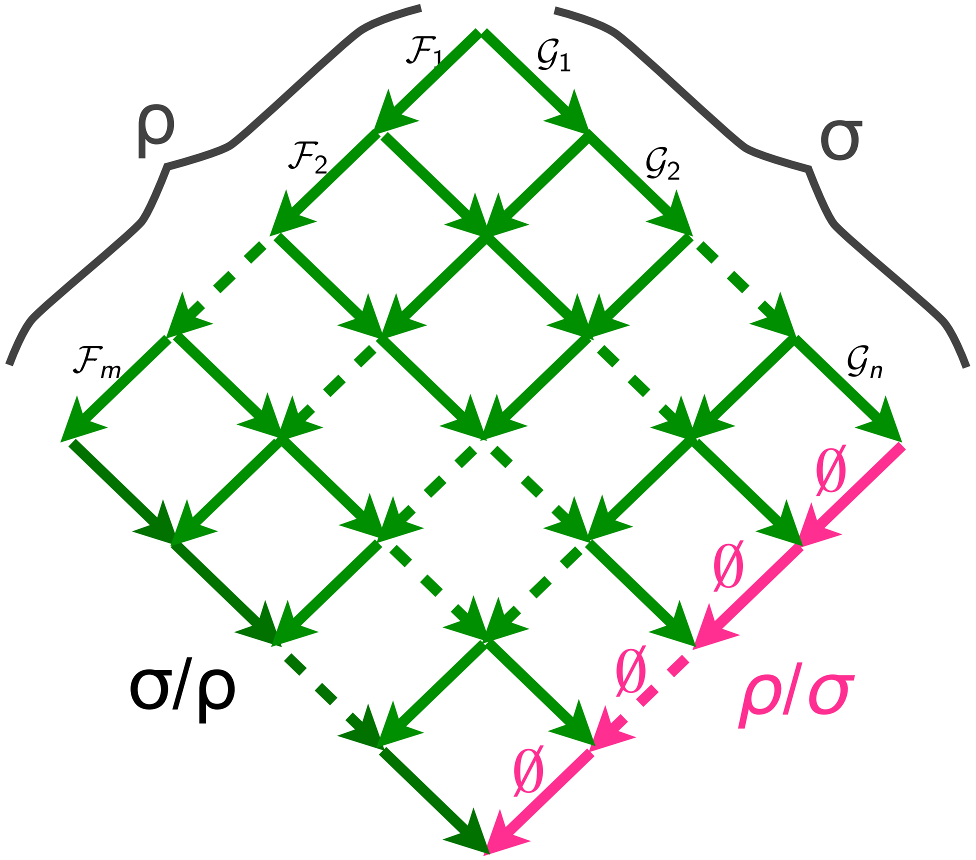


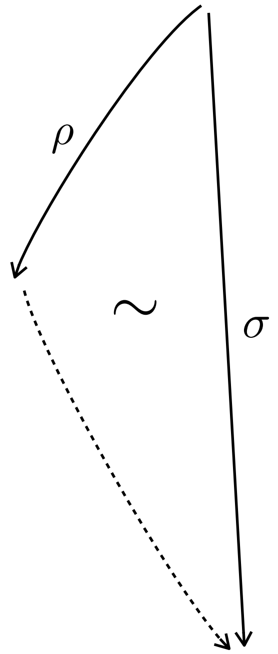
exercise Test previous example and counterexamples for permutation equivalence

Prefix modulo permutations

prefix modulo permutations Let ρ and σ be coinitial reductions.

Then $\rho \leq \sigma$ iff $\rho/\sigma = \emptyset^m$





exercise Prove $\rho \leq \sigma$ iff $\exists \tau,\ \rho\,;\tau \sim \sigma$

Prefix modulo permutations

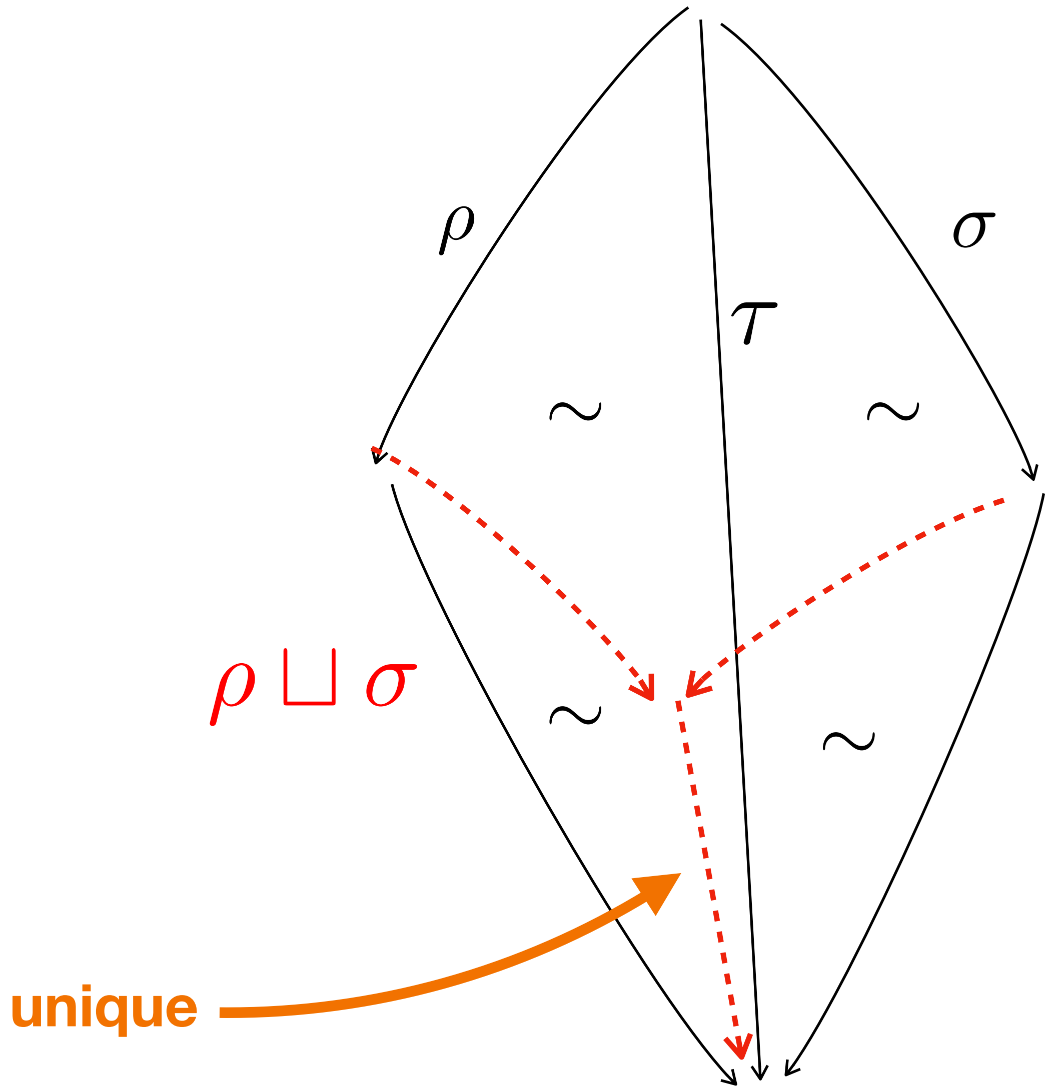
lattice of prefix modulo permutations

$$\begin{tikzcd}
& & I\,((\lambda y.I\,x)\,z) \arrow[dll] \arrow[d] \arrow[drr] & & \\
(\lambda y.I\,x)\,z \arrow[dr] \arrow[drrr] & & I\,(I\,x) \arrow[dr] & & I\,((\lambda y.x)\,z) \arrow[dlll] \arrow[dl] \\
& (\lambda y.x)\,z \arrow[dr] & & I\,x \arrow[dl] & \\
& & x & &
\end{tikzcd}$$

$$\begin{tikzcd}
& I^a((\lambda y.I^b x)z) \arrow[dl] \arrow[d] \arrow[dr] & \\
(\lambda y.I^b x)z \arrow[d] \arrow[dr] & I^a(I^b x) \arrow[d] \arrow[dr] & I^a((\lambda y.x)z) \arrow[dll] \arrow[d] \\
(\lambda y.x)z \arrow[dr] & (I^b x) \arrow[d] & I^a x' \arrow[dl] \\
& x' &
\end{tikzcd}$$

Prefix modulo permutations

lattice of prefix modulo permutations



Easy lemmas

properties of permutations

$$(i) \quad \rho \sim \sigma \iff \forall \tau \,.\, \tau/\rho = \tau/\sigma$$

$$(ii) \quad \rho \sim \sigma \iff \rho/\tau \sim \sigma/\tau$$

$$(iii) \quad \rho \sim \sigma \iff \tau\,;\rho \sim \tau\,;\sigma$$

$$(iv) \quad \rho \sim \sigma \iff \rho\,;\tau \sim \sigma\,;\tau$$

$$(v) \quad \rho \sqcup \sigma \sim \sigma \sqcup \rho$$

proof

(i) $\rho \sim \sigma \implies \sigma/\rho = \emptyset^n$ and $\rho/\sigma = \emptyset^m$

Thus $\tau/(\rho \sqcup \sigma) = \tau/(\rho;(\sigma/\rho)) = \tau/\rho/(\sigma/\rho) = \tau/\rho/\emptyset^n = \tau/$

Similarly $\tau/(\sigma \sqcup \tau) = \tau/\sigma$

By cube lemma $\tau/\rho = \tau/\sigma$

Conversely, take $\tau = \rho$ and $\tau = \sigma$

Easy lemmas

properties of prefixes modulo permutations

$$(i) \quad \rho \leq \sigma \leq \rho \iff \rho \sim \sigma$$

$$(ii) \quad \rho \leq \sigma \leq \tau \implies \rho \leq \tau$$

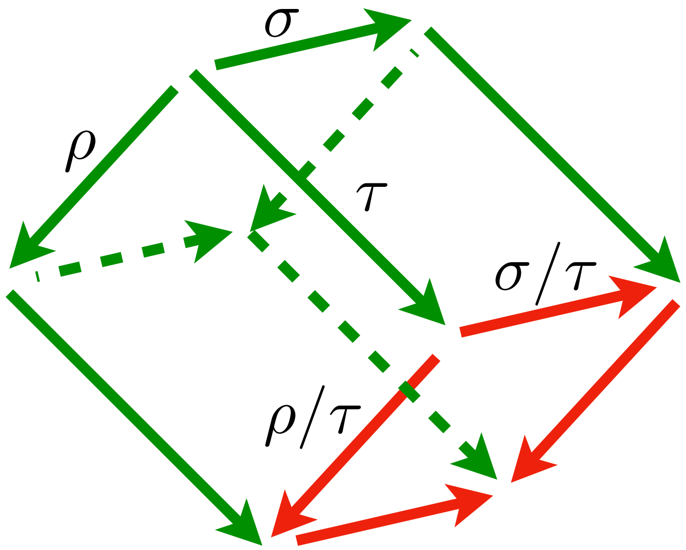
$$(iii) \quad \rho \leq \rho \sqcup \sigma \text{ and } \sigma \leq \rho \sqcup \sigma$$

$$(iv) \quad \rho \leq \tau \text{ and } \sigma \leq \tau \implies \rho \sqcup \sigma \leq \tau$$

$$(v) \quad \rho \leq \sigma \iff \tau ; \rho \leq \tau ; \sigma$$

proof

$$\rho \leq \tau \text{ and } \sigma \leq \tau \implies \rho \sqcup \sigma \leq \tau$$



history: the terminology "equivalence by permutations" is due to Gérard Berry]
[my initial definition was with residual of reductions]

optimal reductions

Initial motivation

- what is call-by-need in the **λ**-calculus ?
  - recursive program schemes with dag implementation [Vuillemin 1973]
  - attempts (non optimal) for the **λ**-calculus [Wadsworth 1972]
  - dag with 1st-order term-rewriting systems [Maranget 1992]
  - using explicit substitutions ? [Ariola et al 1995]

- problem with sharing of functions

- no single-redex reduction strategy can be optimal

example $F_m\,(\lambda y.\,\Delta_n(y\,z))$ where $F_m = \lambda x.\,x\,I\,x\,x\,\ldots\,x$ and $\Delta_n = \lambda x.\,x\,x\,\ldots\,x$

Initial motivation

- no single-redex reduction strategy can be optimal

example

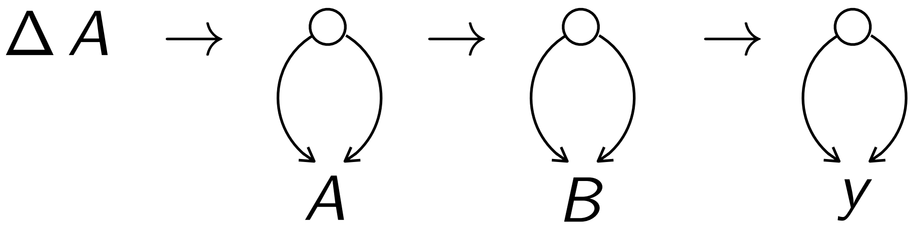
$$\begin{tikzcd}
F_m\,(\lambda y.\,\Delta_n(y\,z)) \text{ where } F_m = \lambda x.\,x\,I\,x\,x\dots x \text{ and } \Delta_n = \lambda x.\,x\,x\dots x \arrow[d] \arrow[dr] & \\
(\lambda y.\Delta_n(y\,z))\,I\,(\lambda y.\Delta_n(y\,z))^{m-1} \arrow[d] & F_m(\lambda y.\,(y\,z)^n) \arrow[d] \\
\Delta_n(I\,z)\,(\lambda y.\Delta_n(y\,z))^{m-1} \arrow[d] & (\lambda y.\,(y\,z)^n)I(\lambda y.\,(y\,z)^n)^{m-1} \arrow[d] \\
\Delta_n\,z\,(\lambda y.\Delta_n(y\,z))^{m-1} \arrow[d] & (I\,z)^n\,(\lambda y.\,(y\,z)^n)^{m-1} \arrow[d, dashed, "n"] \\
z^n\,(\lambda y.\Delta_n(y\,z))^{m-1} \arrow[d, dashed, "m-1"] & z^n\,(\lambda y.\,(y\,z)^n)^{m-1} \\
z^n\,(\lambda y.\,(y\,z)^n)^{m-1} &
\end{tikzcd}$$

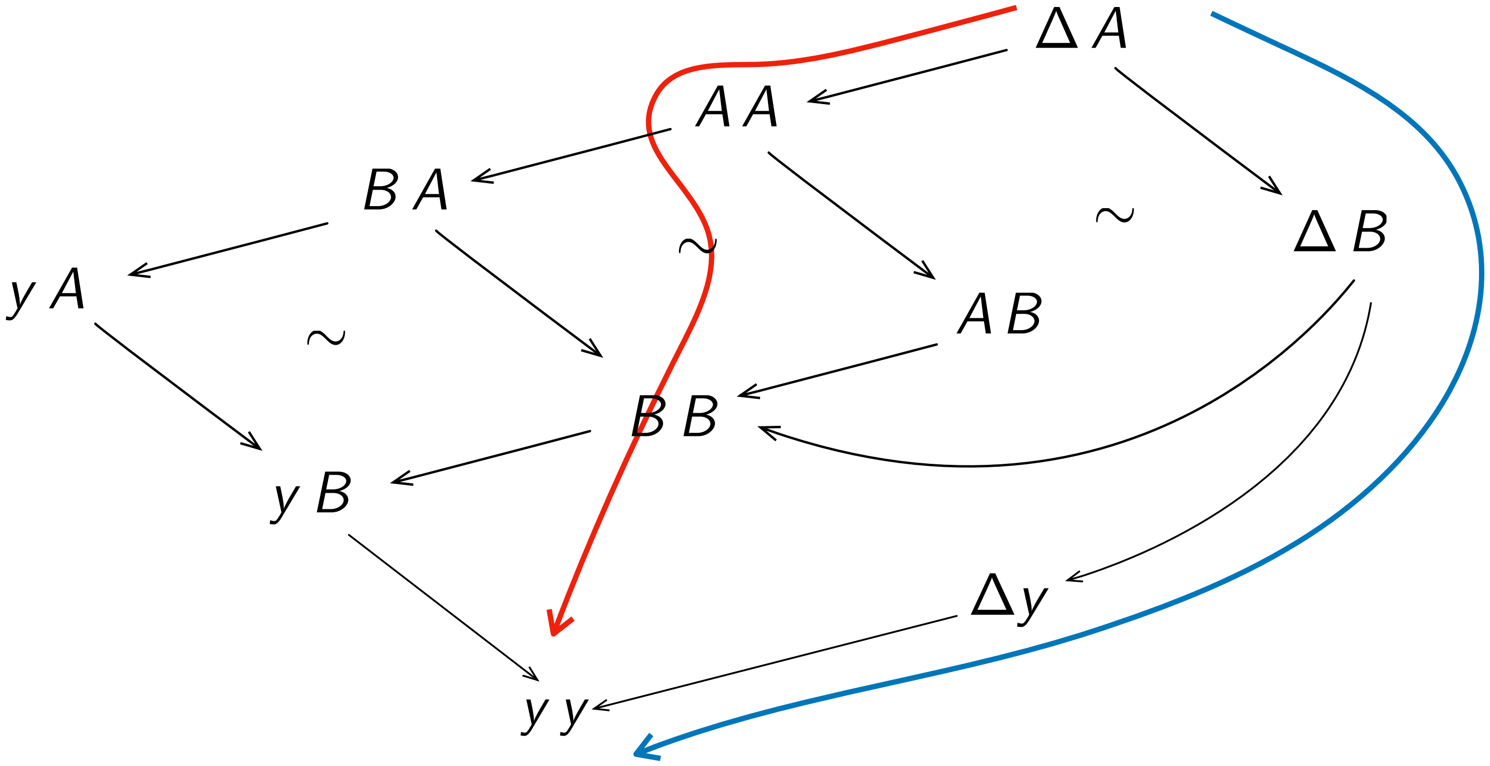
Initial motivation

- call-by-need is call-by-name with sharing to avoid duplications
- what is duplication ?
- duplication along reductions already performed
- so duplication should take care of history of reductions

Duplication of redexes

- call-by-need is call-by-name with sharing to avoid duplications





$\Delta = \lambda x.x\,x$

$F = \lambda f.f\ y$

$I = \lambda x.x$

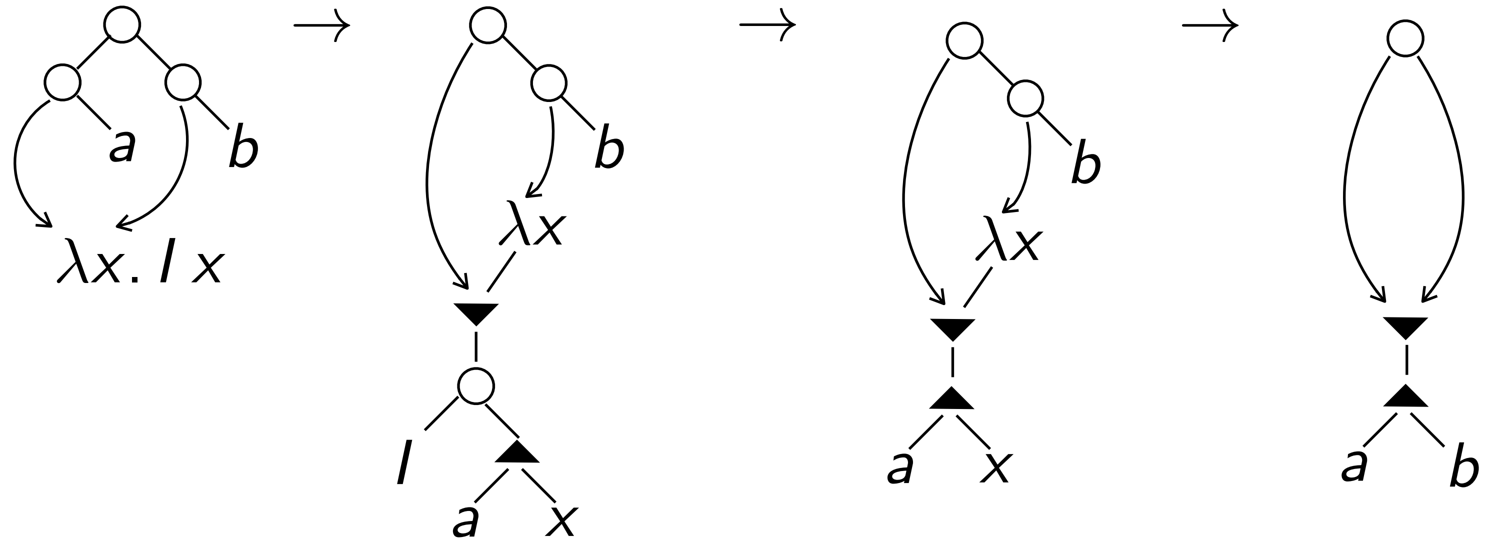
$A = F\ I$

$B = I\ y$

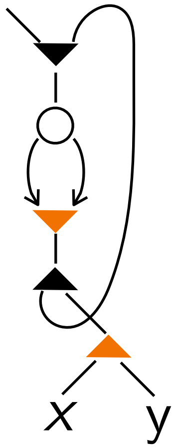
Duplication of redexes

- sharing of **λ**-abstractions ? $I = \lambda x.\, x$

$(\lambda f.\, f\, a\, (f\, b))(\lambda x.\, I\, x) \;\rightarrow$



- possible sharing !!



(x y)(x y)
ou
(x x)(y y)

??

- sharing of contexts ? with boxes and tags ??

- back to easier theory …

Redex & history

h-redex (definition) $\langle \rho, R \rangle$ when R is a redex in final term of ρ

residual of h-redex $\langle \rho, R \rangle \lesssim \langle \sigma, S \rangle$ if there is τ such that $S \in R/\tau$

$$\begin{tikzcd}
 & {} \arrow[dl, bend right, "\rho"'] \arrow[dd, "\sigma"] \\
R \arrow[dr, dashed, "\tau"'] & \sim \\
 & S \in R/\tau
\end{tikzcd}$$

Residuals modulo permutations

- properties of residuals of h-redexes

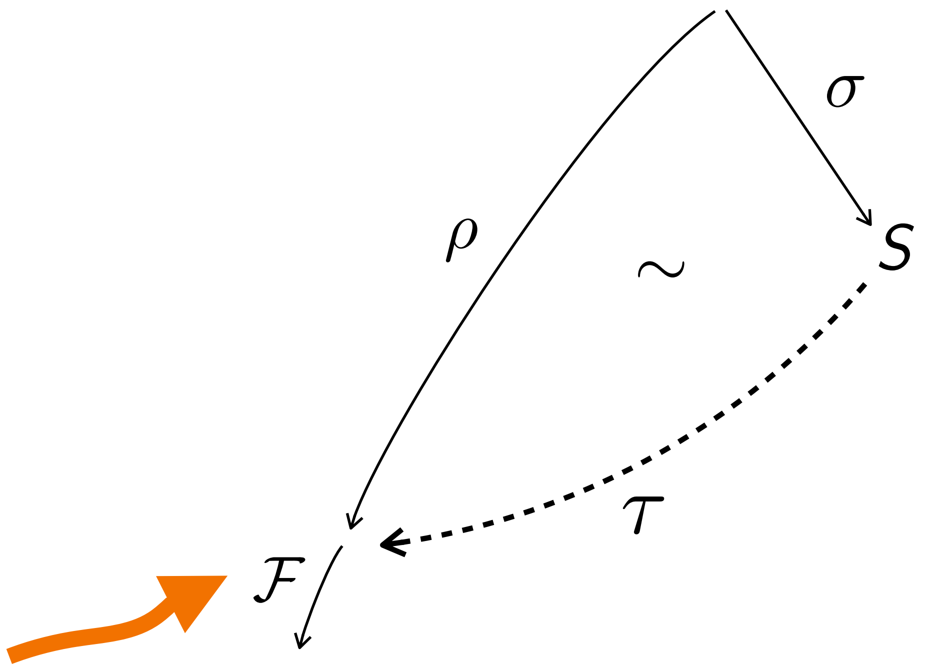
$$
\begin{array}{rl}
(i) & \langle \rho, R\rangle \lesssim \langle \sigma, S\rangle \iff \rho \le \sigma \ \wedge \ S \in R/(\sigma/\rho) \\
(ii) & \langle \rho, R\rangle \lesssim \langle \rho, R\rangle \\
(iii) & \langle \rho, R\rangle \lesssim \langle \sigma, S\rangle \lesssim \langle \rho, R\rangle \iff \rho \sim \sigma \ \wedge \ R = S \\
(iv) & \langle \rho, R\rangle \lesssim \langle \sigma, S\rangle \lesssim \langle \tau, T\rangle \implies \langle \rho, R\rangle \lesssim \langle \tau, T\rangle \\
(v) & \langle \rho, R\rangle \lesssim \langle \tau, T\rangle \ \wedge \ \rho \le \sigma \le \tau \implies \exists! S,\ \langle \rho, R\rangle \lesssim \langle \sigma, S\rangle \lesssim \langle \tau, T\rangle \\
(vi) & \langle \rho, R\rangle \lesssim \langle \sigma, S\rangle \iff \langle \tau;\rho,\ R\rangle \lesssim \langle \tau;\sigma,\ S\rangle
\end{array}
$$

interpolation

- residuals of h-redexes are consistent with permutation equivalence

Duplication complete reductions

- a reduction step $\xrightarrow{\mathcal{F}}$ is duplication complete if $\mathcal{F}$ is a maximal set of h-redexes residuals of a single h-redex



residuals of S by τ

$\mathcal{F} = S/\tau$

Goal **[optimality thm]** leftmost-outermost d-complete reductions are optimal in their number of reduction steps

- but how to find the $\langle \sigma, S \rangle$ h-redex ?

labeled calculus

The labeled λ-calculus

- labels over alphabet $\mathcal{A} = \{a, b, c, \cdots\}$

$$\alpha, \beta \quad ::= \quad a \mid \alpha\beta \mid \lceil\alpha\rceil \mid \lfloor\alpha\rfloor$$

atomic

compound

new atomic names (overlined, underlined)

- labeled **λ**-calculus

$$M, N, \ldots ::= x \mid MN \mid \lambda x.M \mid M^{\alpha}$$

$$(\lambda x.M)^{\alpha} N \rightarrow M\{x := N^{\lfloor\alpha\rfloor}\}^{\lceil\alpha\rceil}$$

$$M^{\alpha}\{x := N\} = M\{x := N\}^{\alpha}$$

$$(M^{\alpha})^{\beta} = M^{\alpha\beta}$$

The labeled **λ**-calculus

- again an alphabet of atomic labels $\mathcal{A} = \{a, b, c, \cdots\}$

- their labeled **λ**-calculus [Asperti-Laneve]

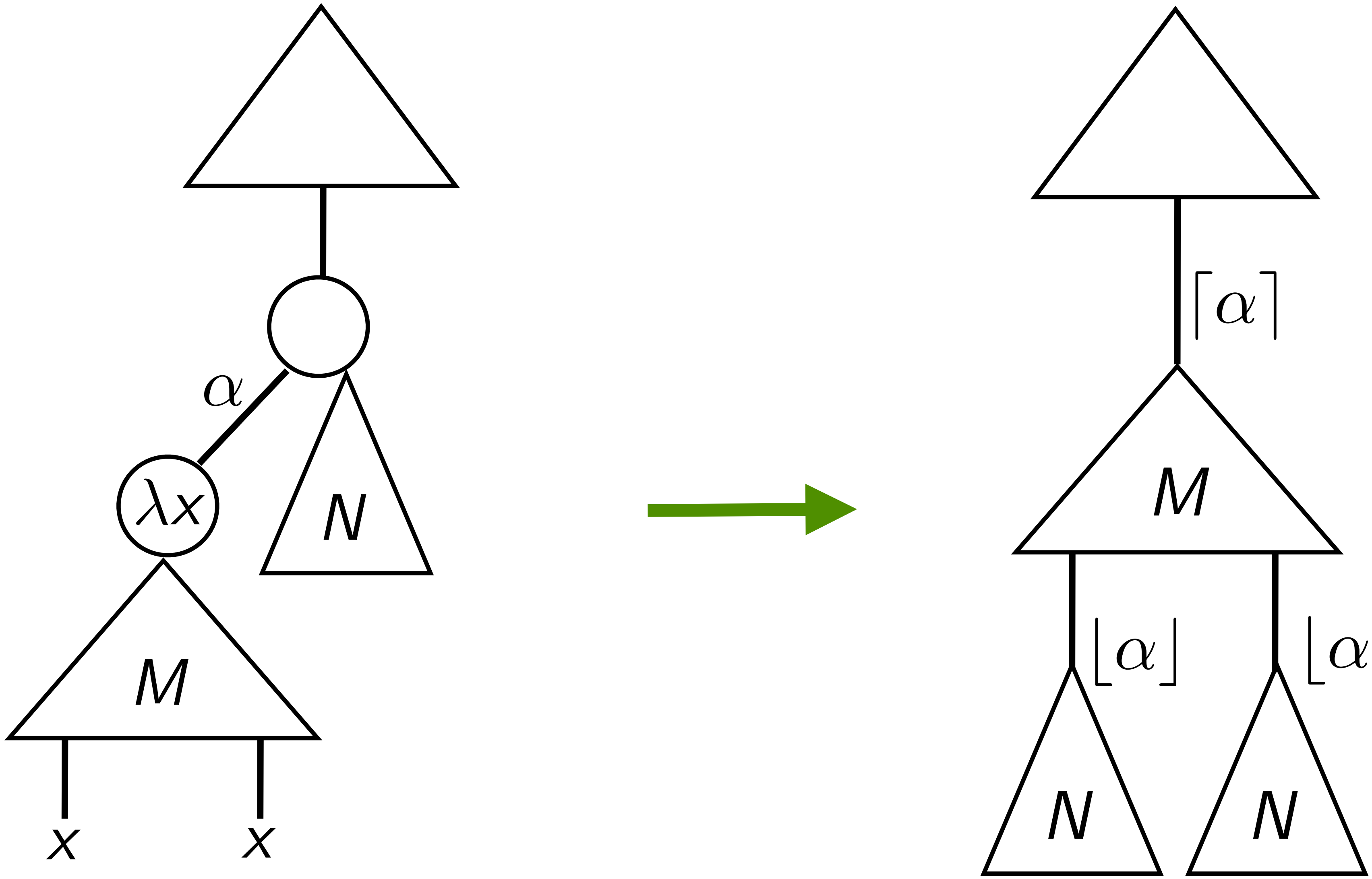
$$M, N, \ldots ::= x \mid MN \mid \lambda x.M \mid a : M$$

$$(a_1 : a_2 : \cdots a_n : \lambda x.M)N \to a_1 : a_2 : \cdots a_n : M\{x := a_n : a_{n-1} : \cdots a_1 : N\}$$

$$(a : M)\{x := N\} = a : M\{x := N\}$$

- correspondence with paths in initial term

The labeled λ-calculus



The labeled λ-calculus

- Let $\Delta = \lambda x.xx,\ \gamma_1 = \lfloor a \rfloor,\ \gamma_2 = \gamma_1 \lfloor \gamma_1 \rfloor$

$$\Delta^a\,\Delta \rightarrow (\Delta^{\gamma_1}\Delta^{\gamma_1})^{\lceil a \rceil} \rightarrow (\Delta^{\gamma_2}\Delta^{\gamma_2})^{\lceil \gamma_1 \rceil \lceil a \rceil} \rightarrow \cdots$$

- the name of a redex be the label of its function part

$$\mathsf{name}(\,(\lambda x.M)^\alpha N\,) = \alpha$$

- the name of a redex gives its **origin**

- residuals of a redex keep their names

- created new redexes strictly contain the names of their creators

The labeled λ-calculus

- An example: $\underline{\Delta} = \lambda x.(x^c\, x^d)^b, \;\; \Delta = \lambda x.(x^g\, x^h)^f$

$$
\begin{array}{lll}
\Omega = \underline{\Delta}^a\, \Delta^e & & \\
\quad \to \Omega_1 = (\Delta^{\gamma_1}\, \Delta^{\delta_1})^{b\lceil a \rceil} & \gamma_1 = e\lfloor a \rfloor c & \delta_1 = e\lfloor a \rfloor d \\
\quad \to \Omega_2 = (\Delta^{\gamma_2}\, \Delta^{\delta_2})^{f\lceil \gamma_1 \rceil b\lceil a \rceil} & \gamma_2 = \delta_1\lfloor \gamma_1 \rfloor g & \delta_2 = \delta_1\lfloor \gamma_1 \rfloor h \\
\quad \to \Omega_3 = (\Delta^{\gamma_3}\, \Delta^{\delta_3})^{f\lceil \gamma_2 \rceil f\lceil \gamma_1 \rceil b\lceil a \rceil} & \gamma_3 = \delta_2\lfloor \gamma_2 \rfloor g & \delta_3 = \delta_2\lfloor \gamma_2 \rfloor h \\
\quad \to \cdots & &
\end{array}
$$

- or simpler with partial labels: $\Delta = \lambda x.x\, x$

$$
\begin{array}{ll}
\Omega = \Delta^a\, \Delta & \\
\quad \to \Omega_1 = (\Delta^{\gamma_1}\, \Delta^{\gamma_1})^{\lceil a \rceil} & \gamma_1 = \lfloor a \rfloor \\
\quad \to \Omega_2 = (\Delta^{\gamma_2}\, \Delta^{\gamma_2})^{\lceil \gamma_1 \rceil \lceil a \rceil} & \gamma_2 = \gamma_1\lfloor \gamma_1 \rfloor \\
\quad \to \Omega_3 = (\Delta^{\gamma_3}\, \Delta^{\gamma_3})^{\lceil \gamma_2 \rceil \lceil \gamma_1 \rceil \lceil a \rceil} & \gamma_3 = \gamma_2\lfloor \gamma_2 \rfloor \\
\quad \to \cdots &
\end{array}
$$

The labeled λ-calculus

- the labeled calculus is confluent

- the labeled calculus is strongly normalizable when reduction is restricted to a finite set of redex names

- unique normal form when exists

  `[ Generalized Finite Developments thm ]`

- the standard **λ**-calculus can be seen as an infinite limit of finite labeled-calculi

The labeled λ-calculus

a

$\Delta = \lambda x.(x^c x^d)^b$

$F = \lambda f.(f^k\, y^\ell)^j$

$I = \lambda x.x^v$

$A = (F^i\, I^u)^q$

$B = (I^\gamma\, y^\ell)^q$

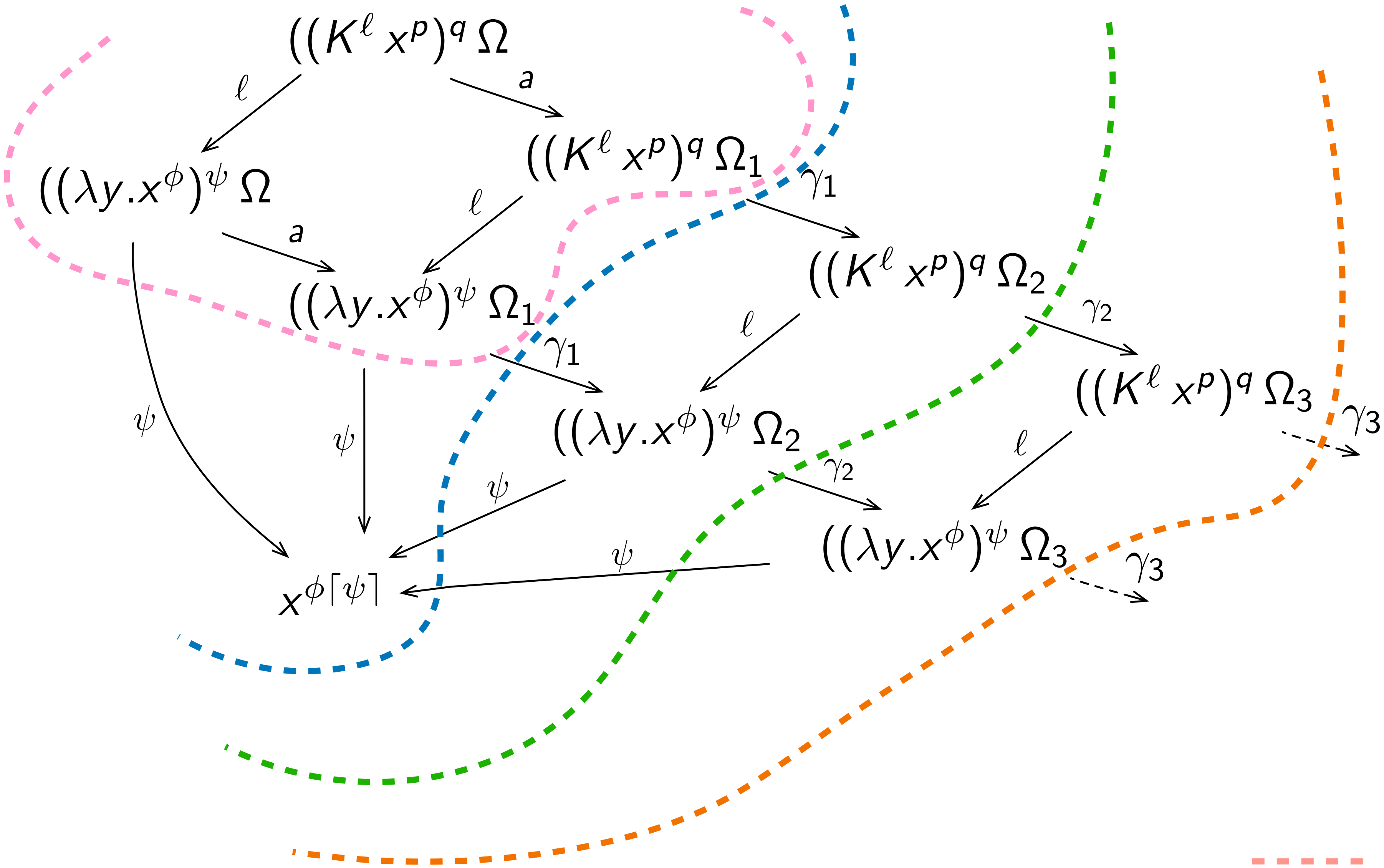
$C = y^{\ell\lfloor\gamma\rfloor v\lceil\gamma\rceil q}$

$\gamma = u\lfloor i\rfloor k$

$$\begin{tikzcd}
 & & & \Delta^a\, A \arrow[dl, "a"'] \arrow[dr, "i"] & & & \\
 & & (A^{\lfloor a\rfloor c}\, A^{\lfloor a\rfloor d})^{b\lceil a\rceil} \arrow[dl, "i"'] \arrow[dr, "i"] & & \Delta^a\, B \arrow[dr, "\gamma"] \arrow[ddll, bend left, "a"] & & \\
 & (B^{\lfloor a\rfloor c}\, A^{\lfloor a\rfloor d})^{b\lceil a\rceil} \arrow[dl, "\gamma"'] \arrow[dr, "i"] & & (A^{\lfloor a\rfloor c}\, B^{\lfloor a\rfloor d})^{b\lceil a\rceil} \arrow[dl, "i"'] \arrow[dr, "\gamma"] & & \Delta^a\, C \arrow[dddlll, bend left, "a"] & \\
(C^{\lfloor a\rfloor c}\, A^{\lfloor a\rfloor d})^{b\lceil a\rceil} \arrow[dr, "i"] & & (B^{\lfloor a\rfloor c}\, B^{\lfloor a\rfloor d})^{b\lceil a\rceil} \arrow[dl, "\gamma"'] \arrow[dr, "\gamma"] & & (A^{\lfloor a\rfloor c}\, C^{\lfloor a\rfloor d})^{b\lceil a\rceil} \arrow[dl, "i"'] & & \\
 & (C^{\lfloor a\rfloor c}\, B^{\lfloor a\rfloor d})^{b\lceil a\rceil} \arrow[dr, "\gamma"] & & (B^{\lfloor a\rfloor c}\, C^{\lfloor a\rfloor d})^{b\lceil a\rceil} \arrow[dl, "\gamma"'] & & & \\
 & & (C^{\lfloor a\rfloor c}\, C^{\lfloor a\rfloor d})^{b\lceil a\rceil} & & & &
\end{tikzcd}$$

confluence

The labeled λ-calculus



$$\begin{tikzcd}
 & ((K^{\ell}\, x^{p})^{q}\, \Omega \arrow[dl, "\ell"'] \arrow[dr, "a"] & & & & \\
((\lambda y.x^{\phi})^{\psi}\, \Omega \arrow[dr, "a"] \arrow[ddd, "\psi"'] & & ((K^{\ell}\, x^{p})^{q}\, \Omega_1 \arrow[dl, "\ell"'] \arrow[dr, "\gamma_1"] & & & \\
 & ((\lambda y.x^{\phi})^{\psi}\, \Omega_1 \arrow[dr, "\gamma_1"] \arrow[ddl, "\psi"] & & ((K^{\ell}\, x^{p})^{q}\, \Omega_2 \arrow[dl, "\ell"'] \arrow[dr, "\gamma_2"] & & \\
 & & ((\lambda y.x^{\phi})^{\psi}\, \Omega_2 \arrow[dr, "\gamma_2"] \arrow[dll, "\psi"'] & & ((K^{\ell}\, x^{p})^{q}\, \Omega_3 \arrow[dl, "\ell"'] \arrow[dr, dashed, "\gamma_3"] & \\
x^{\phi\lceil\psi\rceil} & & & ((\lambda y.x^{\phi})^{\psi}\, \Omega_3 \arrow[lll, "\psi"'] \arrow[dr, dashed, "\gamma_3"] & & \\
 & & & & &
\end{tikzcd}$$

$K = \lambda x.(\lambda y.x^{n})^{m}$

$\phi = p\lfloor\ell\rfloor n$

$\psi = m\lceil\ell\rceil q$

strong normalization

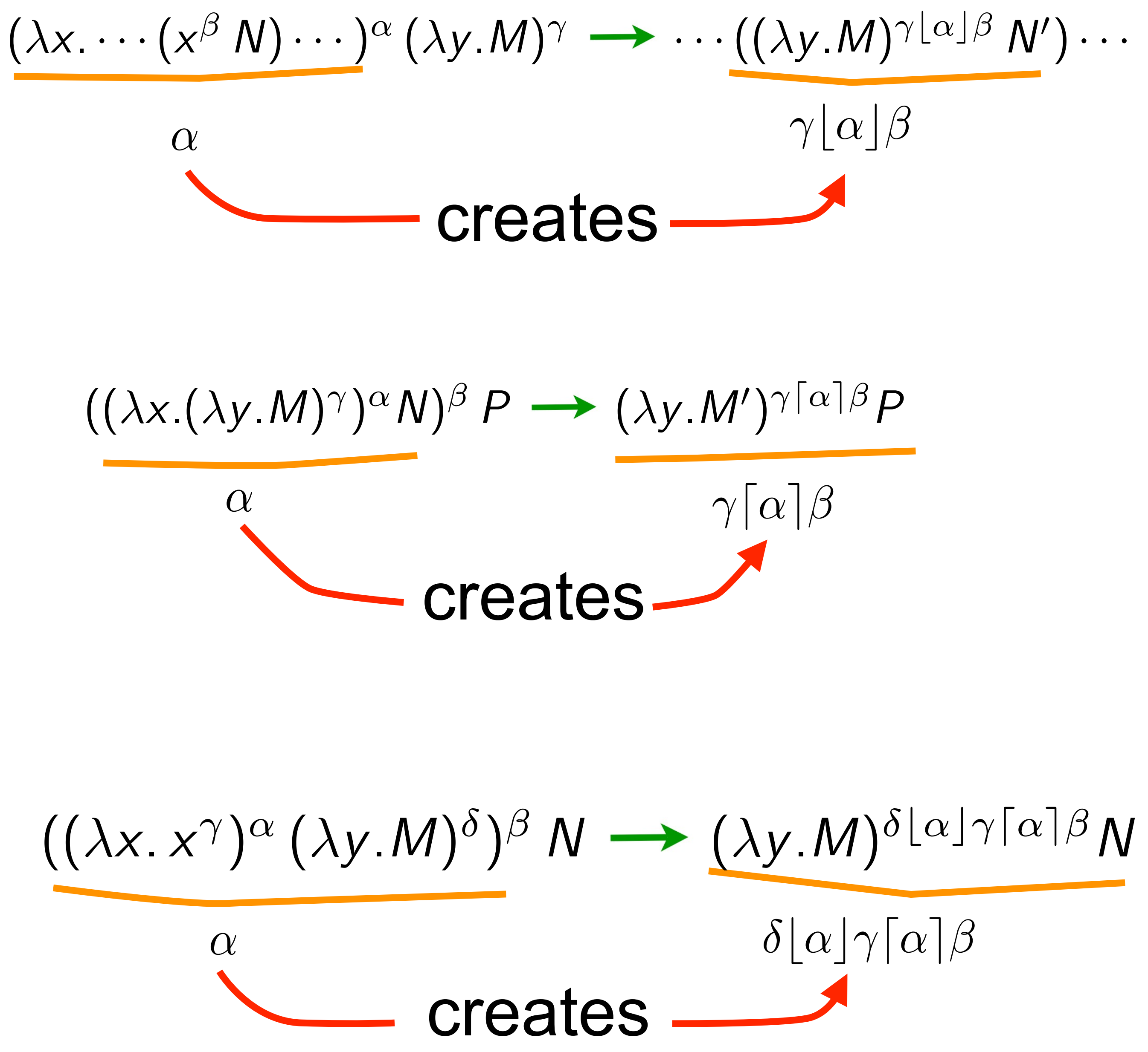
╌╌ $\{\ell, a\}$

╌╌ $\{\ell, a, \psi\}$

╌╌ $\{\ell, a, \psi, \gamma_1\}$

╌╌ $\{\ell, a, \psi, \gamma_1, \gamma_2\}$

The labeled λ-calculus



$$\underbrace{(\lambda x.\cdots(x^{\beta}\,N)\cdots)^{\alpha}}_{\alpha}(\lambda y.M)^{\gamma} \to \cdots\underbrace{((\lambda y.M)^{\gamma\lfloor\alpha\rfloor\beta}\,N')}_{\gamma\lfloor\alpha\rfloor\beta}\cdots$$

creates

$$(\underbrace{(\lambda x.(\lambda y.M)^{\gamma})^{\alpha}}_{\alpha}N)^{\beta}\,P \to \underbrace{(\lambda y.M')^{\gamma\lceil\alpha\rceil\beta}P}_{\gamma\lceil\alpha\rceil\beta}$$

creates

$$\underbrace{((\lambda x.\,x^{\gamma})^{\alpha}\,(\lambda y.M)^{\delta})^{\beta}}_{\alpha}\,N \to \underbrace{(\lambda y.M)^{\delta\lfloor\alpha\rfloor\gamma\lceil\alpha\rceil\beta}N}_{\delta\lfloor\alpha\rfloor\gamma\lceil\alpha\rceil\beta}$$

creates

origin

The labeled λ-calculus

$$\begin{tikzcd}
 & & I\,((\lambda y.I\,x)\,z) \arrow[dll] \arrow[d] \arrow[drr] & & \\
(\lambda y.I\,x)\,z \arrow[dr] \arrow[drrr] & & I\,(I\,x) \arrow[dr] & & I\,((\lambda y.x)\,z) \arrow[dlll] \arrow[dl] \\
 & (\lambda y.x)\,z \arrow[dr] & & I\,x \arrow[dl] & \\
 & & x & &
\end{tikzcd}$$

$I = \lambda x.x$

$\alpha = \lfloor a \rfloor \lceil a \rceil$

$\beta = \lfloor b \rfloor \lceil b \rceil$

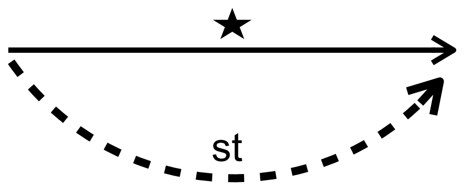
$\gamma = \lceil c \rceil$

$$\begin{tikzcd}
 & I^a((\lambda y.I^b x)^c z) \arrow[dl] \arrow[d] \arrow[dr] & \\
((\lambda y.I^b x)^c z)^\alpha \arrow[d] \arrow[dr] & I^a(I^b x)^\gamma \arrow[d] \arrow[dr] & I^a((\lambda y.x^\beta)^c z) \arrow[dll] \arrow[d] \\
((\lambda y.x^\beta)^c z)^\alpha \arrow[dr] & (I^b x)^{\gamma\alpha} \arrow[d] & I^a x^{\beta\gamma} \arrow[dl] \\
 & x^{\beta\gamma\alpha} &
\end{tikzcd}$$

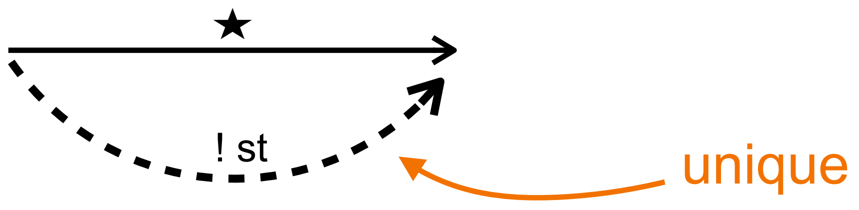
lattice

The labeled λ-calculus

- a **standard** reduction is an outside-in left-to-right reduction strategy
- any reduction can be reordered in a standard reduction `[Curry 1958]`



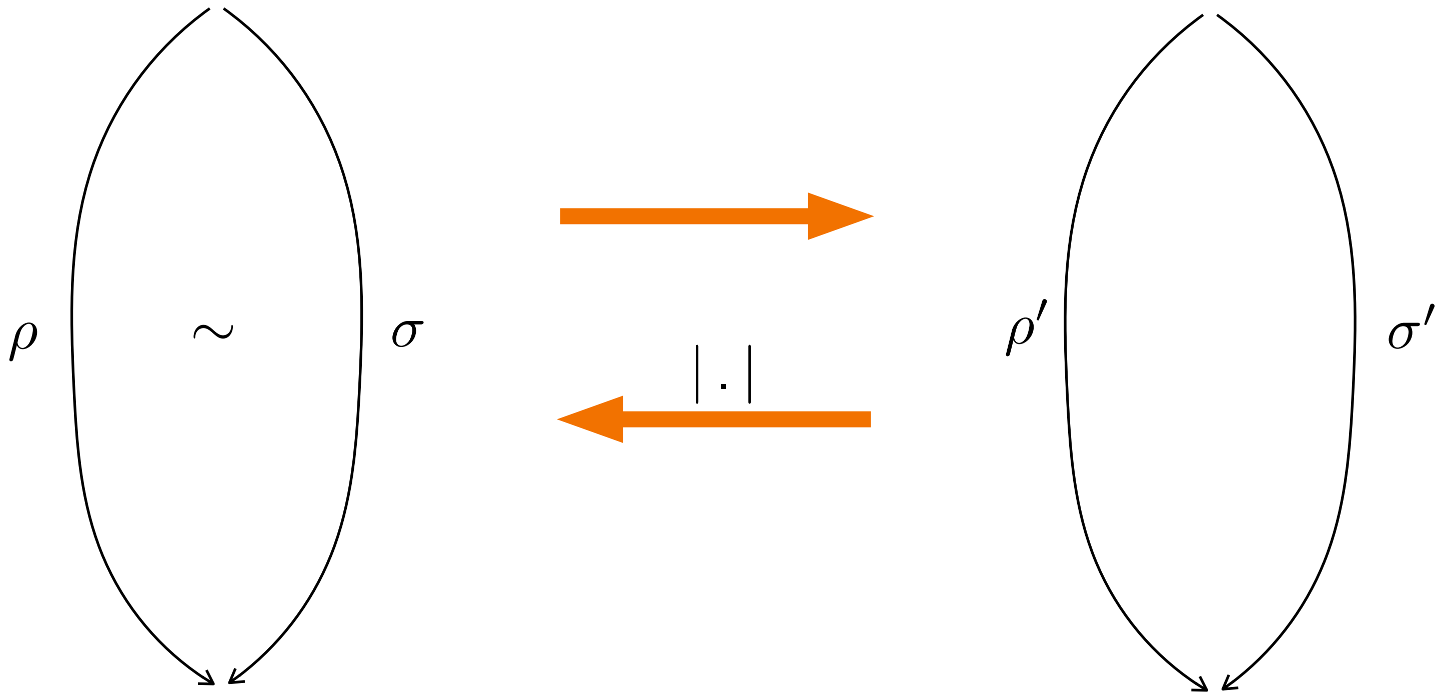
λ-calculus



labeled λ-calculus

Permutation equivalence

- $\sim$ corresponds to the coinitial / cofinal reductions of the labeled **λ**-calculus



λ-calculus

$\rho \sim \sigma$

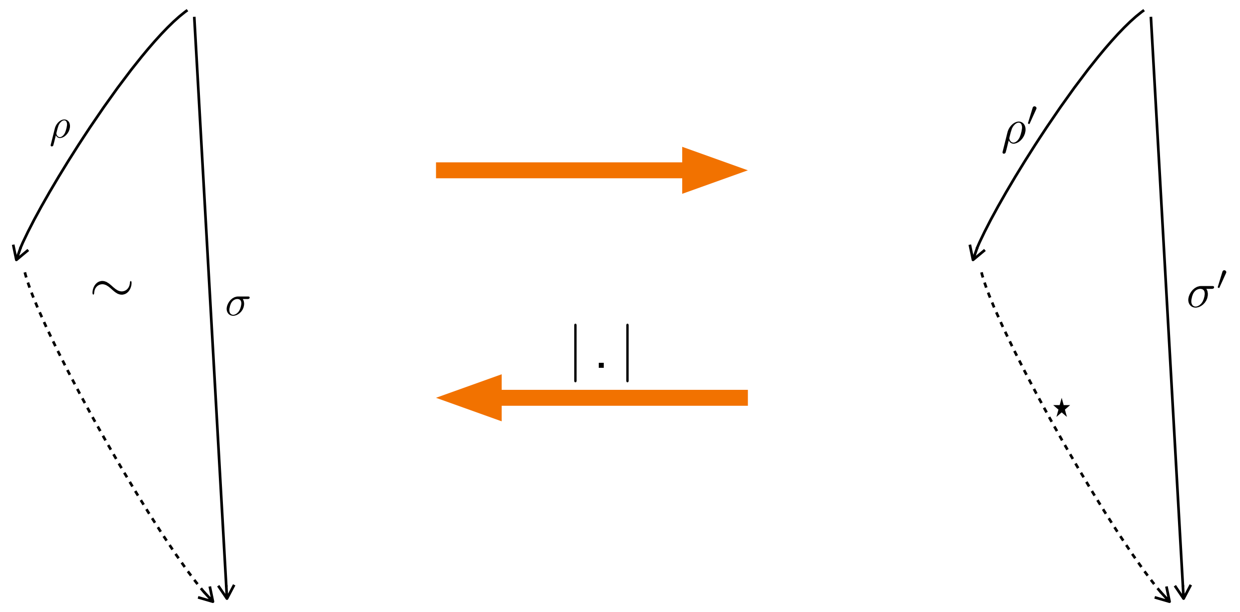
labeled **λ**-calculus

$\mathsf{org}(\rho') = \mathsf{org}(\sigma')$

$\mathsf{end}(\rho') = \mathsf{end}(\sigma')$

Prefix modulo permutations

- $\leq$ corresponds to reductions of the labeled **λ**-calculus



λ-calculus

$\rho \leq \sigma$

labeled **λ**-calculus

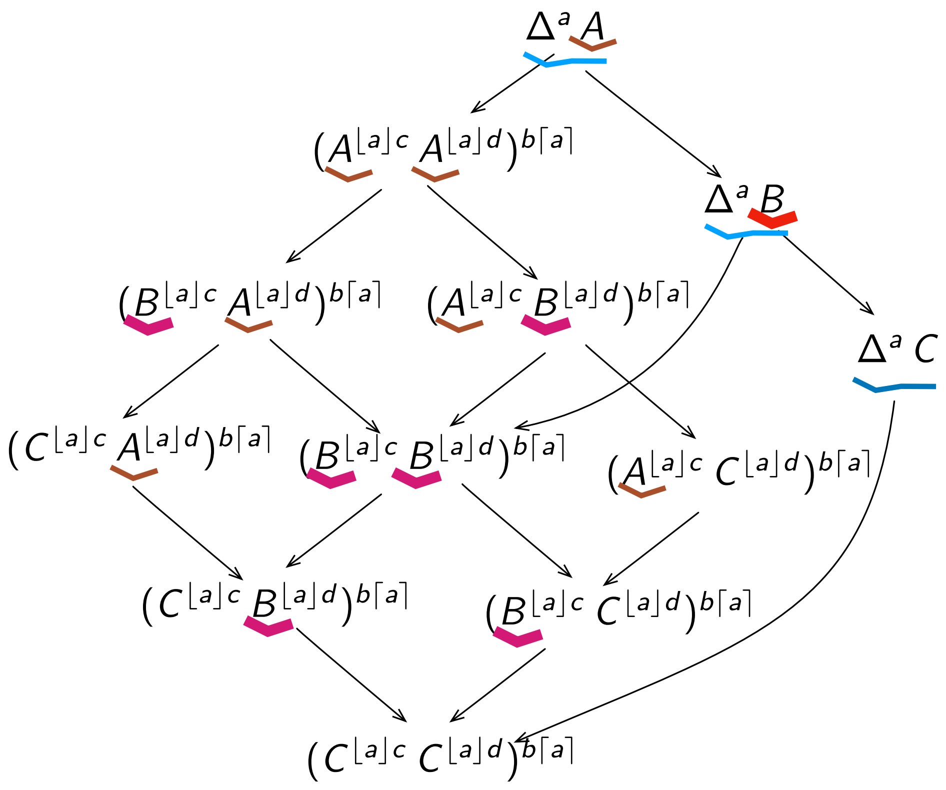
$\mathsf{org}(\rho') = \mathsf{org}(\sigma')$

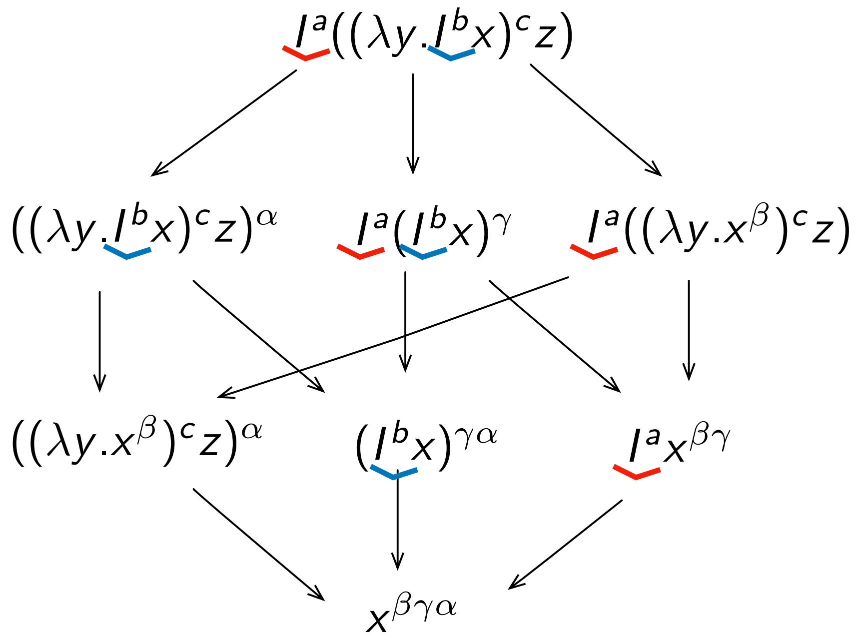
$\mathsf{end}(\rho') \twoheadrightarrow \mathsf{end}(\sigma')$

redex families

Residuals modulo permutations

- residuals of h-redexes correspond to names of redexes in :





$\Delta = \lambda x.(x^c x^d)^b$

$F = \lambda f.(f^k\, y^\ell)^j$

$I = \lambda x.x^v$

$A = (F^i\, I^u)^q$

$B = (I^\gamma\, y^\ell)^q$

$C = y^{\ell \lfloor \gamma \rfloor v \lceil \gamma \rceil q}$

$\alpha = \lceil a \rceil$

$\beta = \lceil b \rceil$

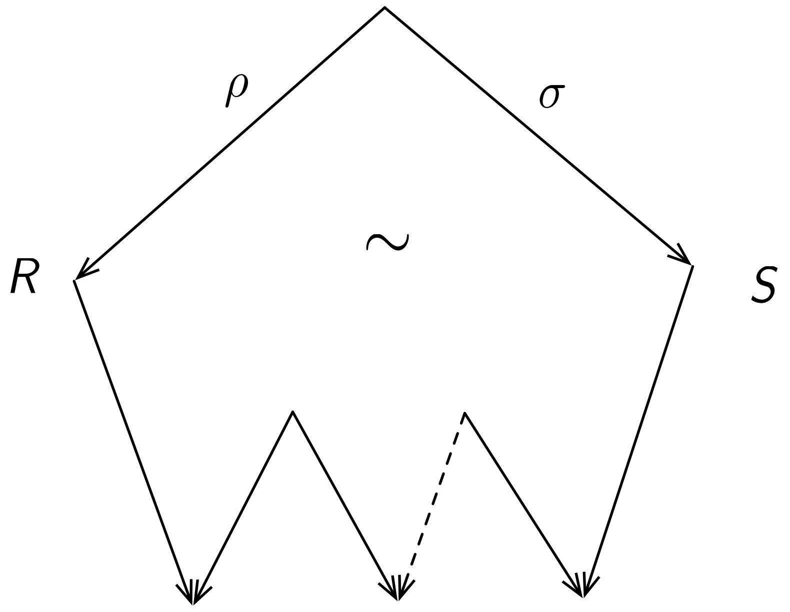
$\gamma = \lceil c \rceil$

Redex families

family relation $\simeq$ between h-redexes is defined by :

$$(i) \quad \langle \rho, R \rangle \lesssim \langle \sigma, S \rangle \implies \langle \rho, R \rangle \simeq \langle \sigma, S \rangle \simeq \langle \rho, R \rangle$$

$$(ii) \quad \langle \rho, R \rangle \simeq \langle \sigma, S \rangle \simeq \langle \tau, T \rangle \implies \langle \rho, R \rangle \simeq \langle \tau, T \rangle$$



- symmetric + transitive closure of residuals modulo permutations

Redex families

- from now on, we only consider standard reductions
- then the extraction relation $\triangleleft$ on h-redexes is defined as follows

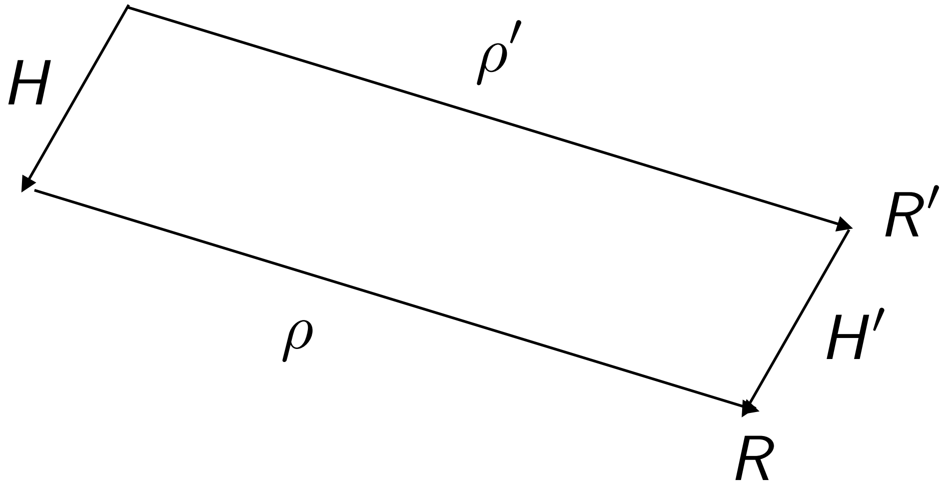
$$(i) \quad \langle o, R \rangle \triangleleft \langle o, R \rangle$$

$$(ii) \quad \langle \rho, R \rangle \triangleleft \langle \sigma, S \rangle \implies \langle \rho', R' \rangle \triangleleft \langle H; \sigma,\ S \rangle$$

where $\langle \rho', R' \rangle$ is defined by cases analysis on ρ w.r.t. H

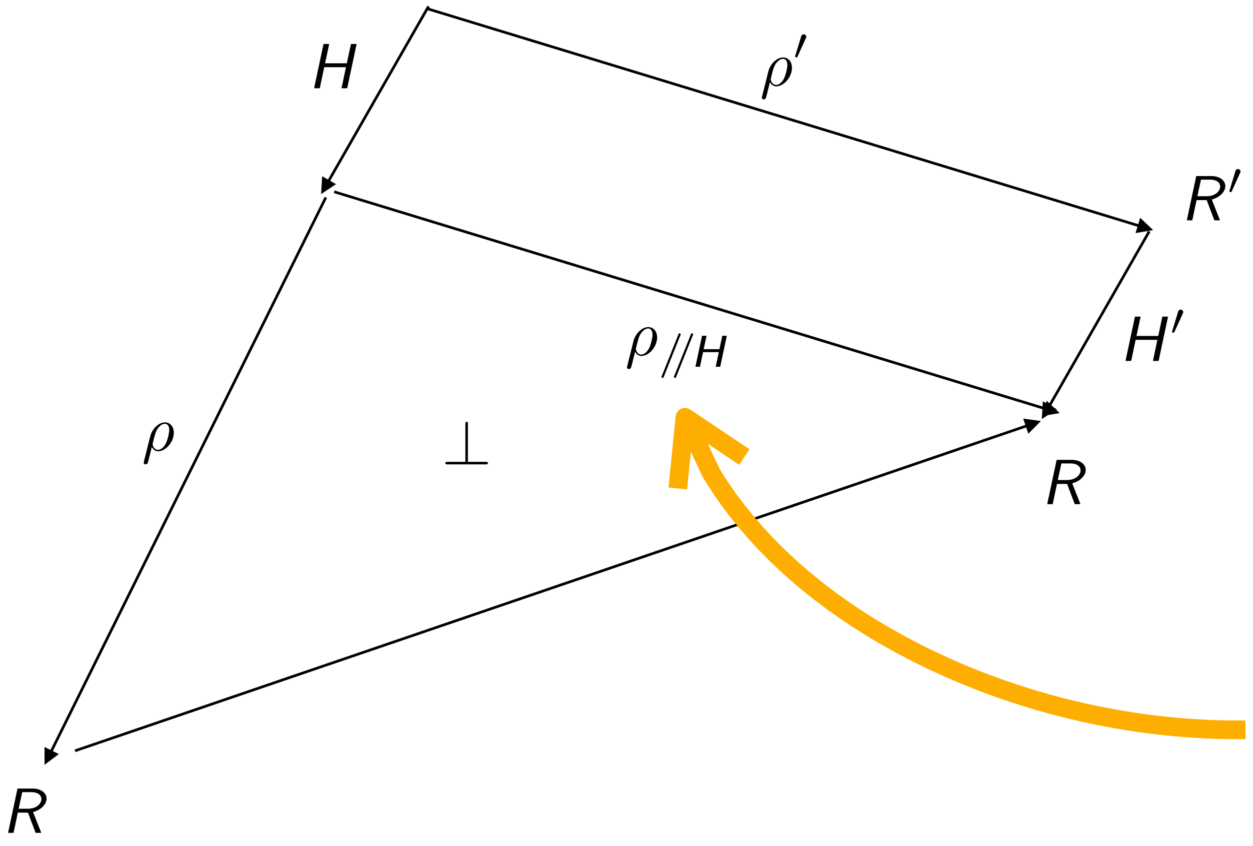
Redex families

- **Case 1:** ρ is in body of *H* or disjoint to the right of the contractum of *H* then ρ' is isomorphic to ρ



Redex families

- **Case 2:** ρ is internal to an instance of a copy of the argument of H then ρ' is isomorphic to ρ in the argument of H

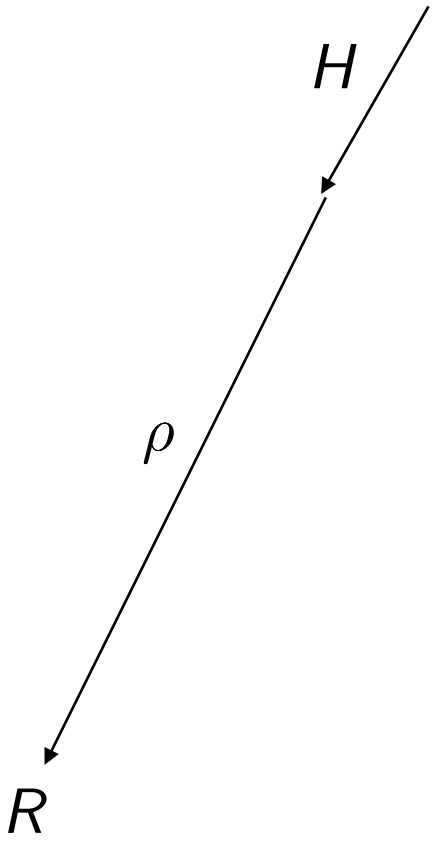


ρ parellized by H

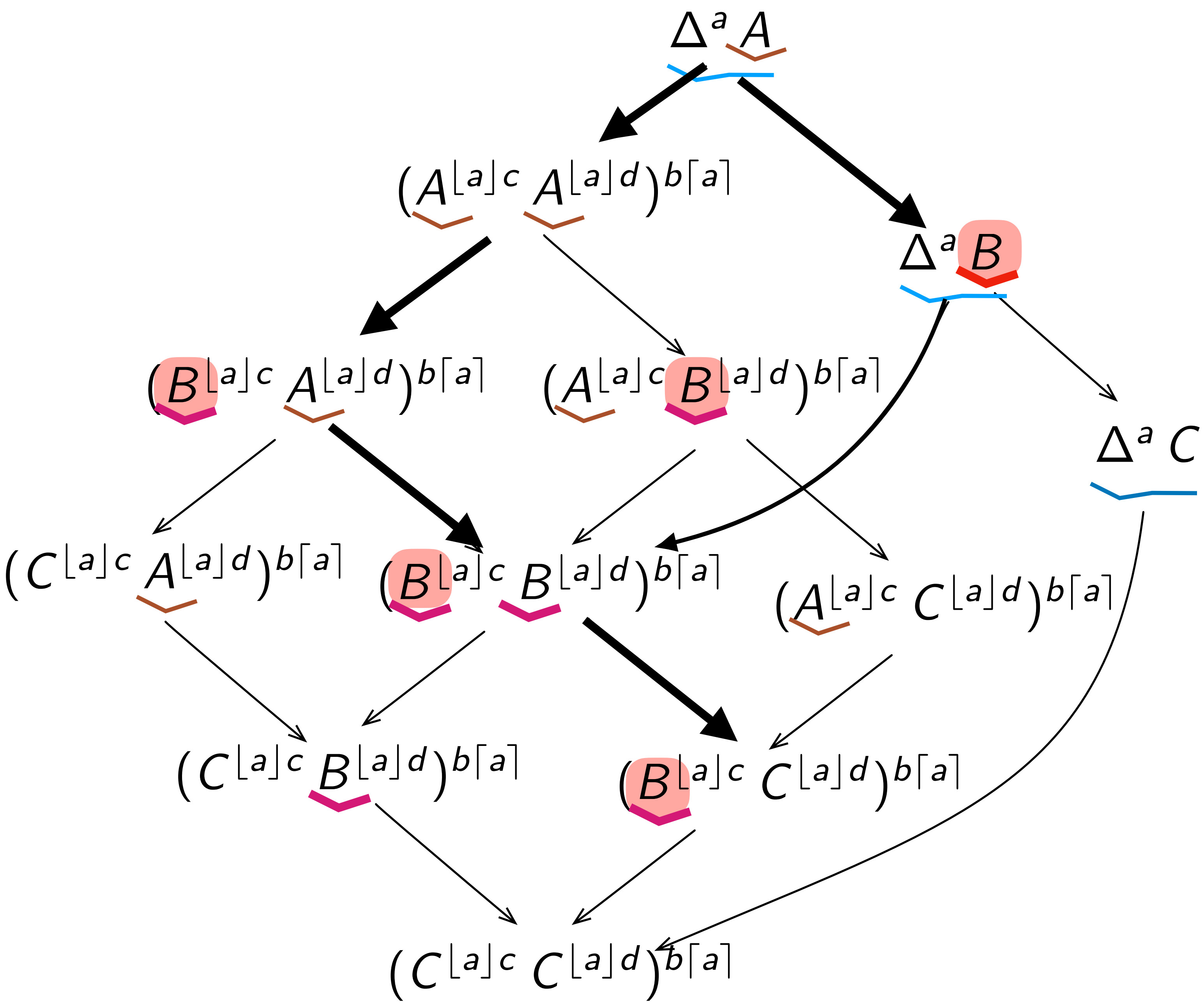
Redex families

- **Otherwise** (H necessary for R)

$$\rho' = H; \rho \quad \wedge \quad R' = R$$



Redex families



Redex families

- the family relation $\simeq$ can be decided by extraction

$$\langle \rho, R \rangle \simeq \langle \sigma, S \rangle \iff \langle \tau, T \rangle \lhd \langle \rho, R \rangle \ \wedge \ \langle \tau, T \rangle \lhd \langle \sigma, S \rangle \quad \text{for some } \langle \tau, T \rangle$$

- in fact $\langle \tau, T \rangle$ is unique and is the canonical representative of its family

- $\langle \tau, T \rangle$ is unique in family with minimum length of (standard) reduction τ

redexes are stable in the **λ**-calculus

↓

sequentiality

Redex families

- when $\langle \tau, T \rangle$ is a canonical representative

$$\langle \rho, R \rangle \simeq \langle \tau, T \rangle \;\wedge\; \tau \leq \rho \quad \Longrightarrow \quad \langle \tau, T \rangle \lesssim \langle \rho, R \rangle$$

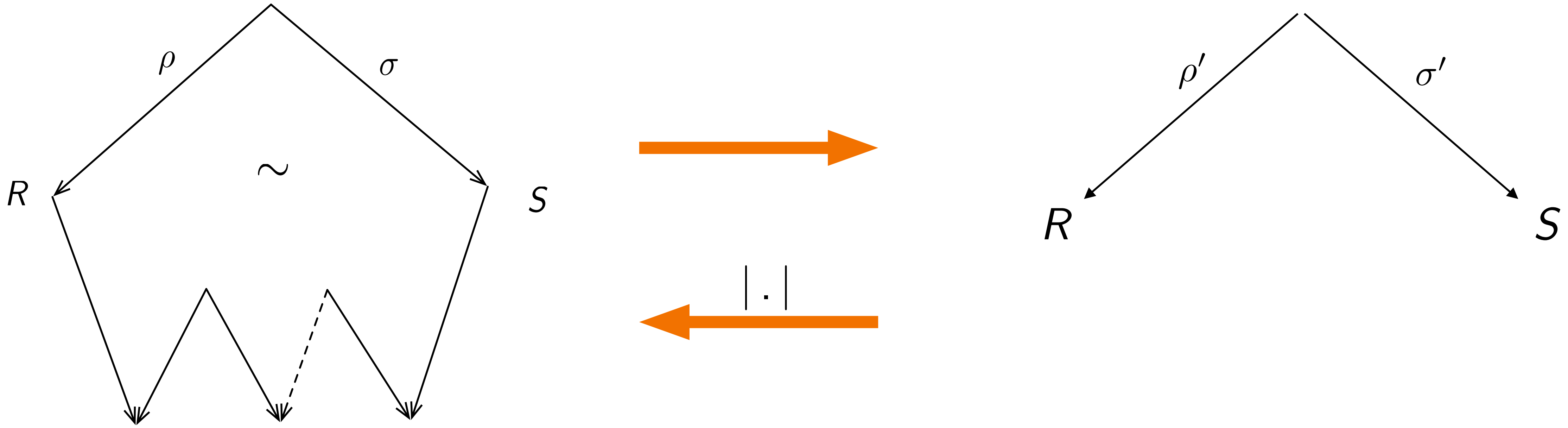
- family complete reductions are duplication complete reductions

- sublattice of family complete reductions

[optimality thm] leftmost-outermost d-complete reductions are optimal in their number of reduction steps

Redex families

- the family relation $\simeq$ corresponds to names in the labeled calculus



λ-calculus

$$\langle \rho, R\rangle \simeq \langle \sigma, S\rangle$$

labeled **λ**-calculus

$$\mathsf{name}(R) = \mathsf{name}(S)$$

when initial term is **labeled with distinct letters**

- family complete reductions are labeled complete reductions

The sublattice of optimal reductions

- optimal family complete reductions

example $F_m\,(\lambda y.\,\Delta_n(y\,z))$ where $F_m = \lambda x.\, x\, I\, x\, x \dots x$ and $\Delta_n = \lambda x.\, x\, x \dots x$

$$\begin{tikzcd}
F_m\,(\lambda y.\,\Delta_n(y\,z)) \arrow[d] \arrow[dr] & \\
(\lambda y.\Delta_n(y\,z))\; I\; (\lambda y.\Delta_n(y\,z))^{m-1} \arrow[d] \arrow[dr] & F_m(\lambda y.\,(y\,z)^n) \arrow[d] \\
\Delta_n(I\,z)\,(\lambda y.\Delta_n(y\,z))^{m-1} \arrow[d] \arrow[dr, orange, thick] & (\lambda y.(y\,z)^n)\; I\; (\lambda y.(y\,z)^n)^{m-1} \arrow[d] \\
\Delta_n\, z\,(\lambda y.\Delta_n(y\,z))^{m-1} \arrow[dr] & (I\,z)^n\,(\lambda y.\,(y\,z)^n)^{m-1} \arrow[d, orange, thick] \\
 & z^n\,(\lambda y.\,(y\,z)^n)^{m-1}
\end{tikzcd}$$

Generalized finite developments

GFD++ thm [JJL] let $\mathcal{F}$ be a finite set of redex families

- all reductions contracting only h-redexes in $\mathcal{F}$ have finite length
- these maximal reductions are permutation equivalent

corollary a lambda-term is strongly normalizable iff it only contracts a finite set of redex families.

strong normalization ↔ finiteness of redex families

Extra properties

- algebraic laws with parallel reductions of redexes
- residuals of parallel reductions
- optimality of family complete reductions
- reductions with ultra sharing `[Lamping]`
- linear logic without boxes `[Gonthier et al]`
- generalization to other systems (interaction systems, …)

Conclusion

- real implementations of sharing (more than call-by-need) ?

  [non exponential implementations]

- subsets where possible manageable sharing (weak calculi, others?)

- intuitive proofs of strong normalization $(\lfloor \alpha \rightarrow \beta \rfloor = \alpha,\ \lceil \alpha \rightarrow \beta \rceil = \beta,\ \alpha\beta = \alpha)$

- simplification of the extraction process

- history-based information flow

- incremental computations (makefiles [Vesta], neural networks)