Tracking Redexes in the lambda-calculus

(rev. 2:1)

jean-jacques.levy@inria.fr

Nanjing NJU

2024-11-22

http://jeanjacqueslevy.net/talks/24track-rev2/track.pdf

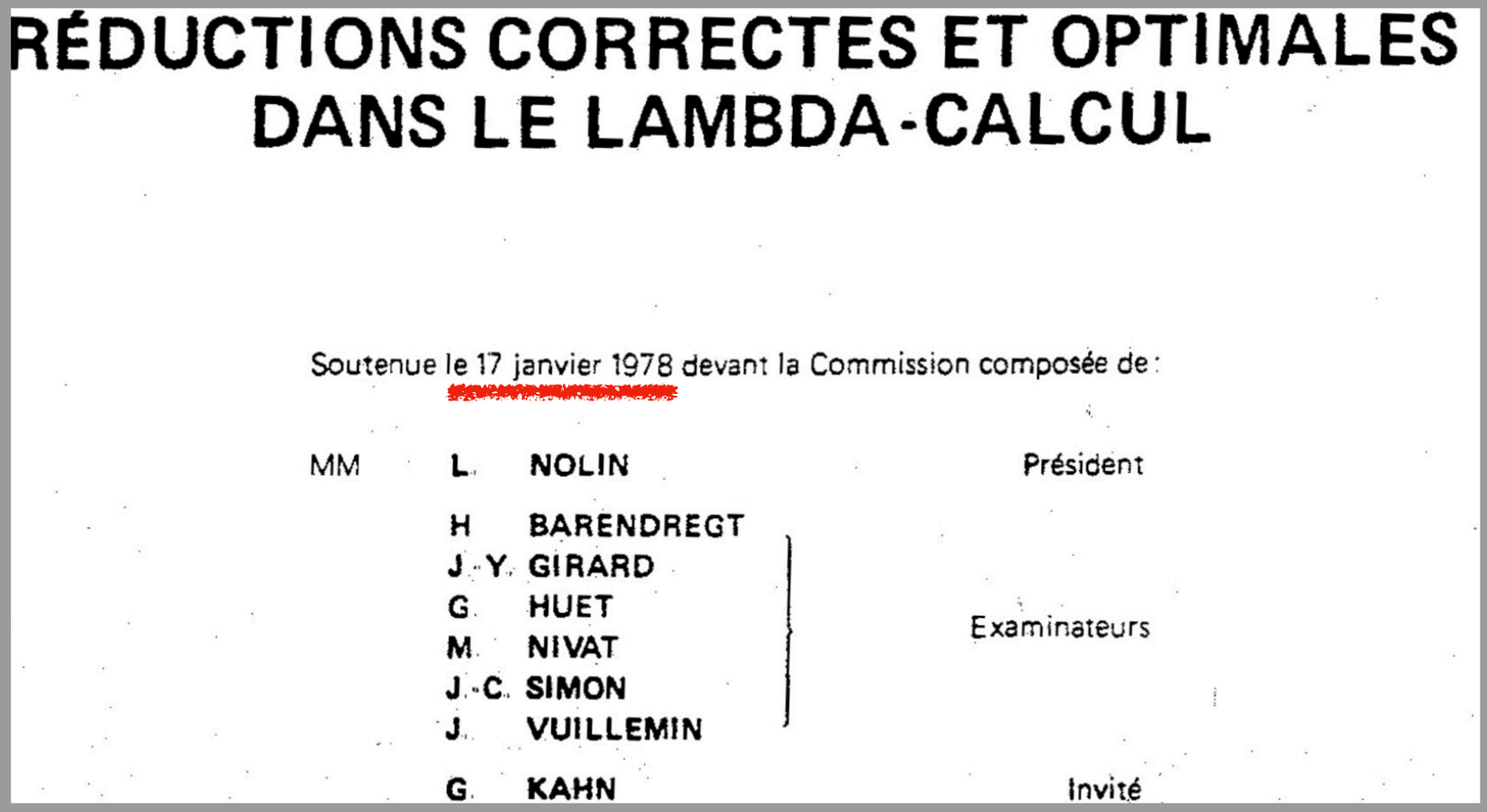
The lambda-calculus

- 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 :

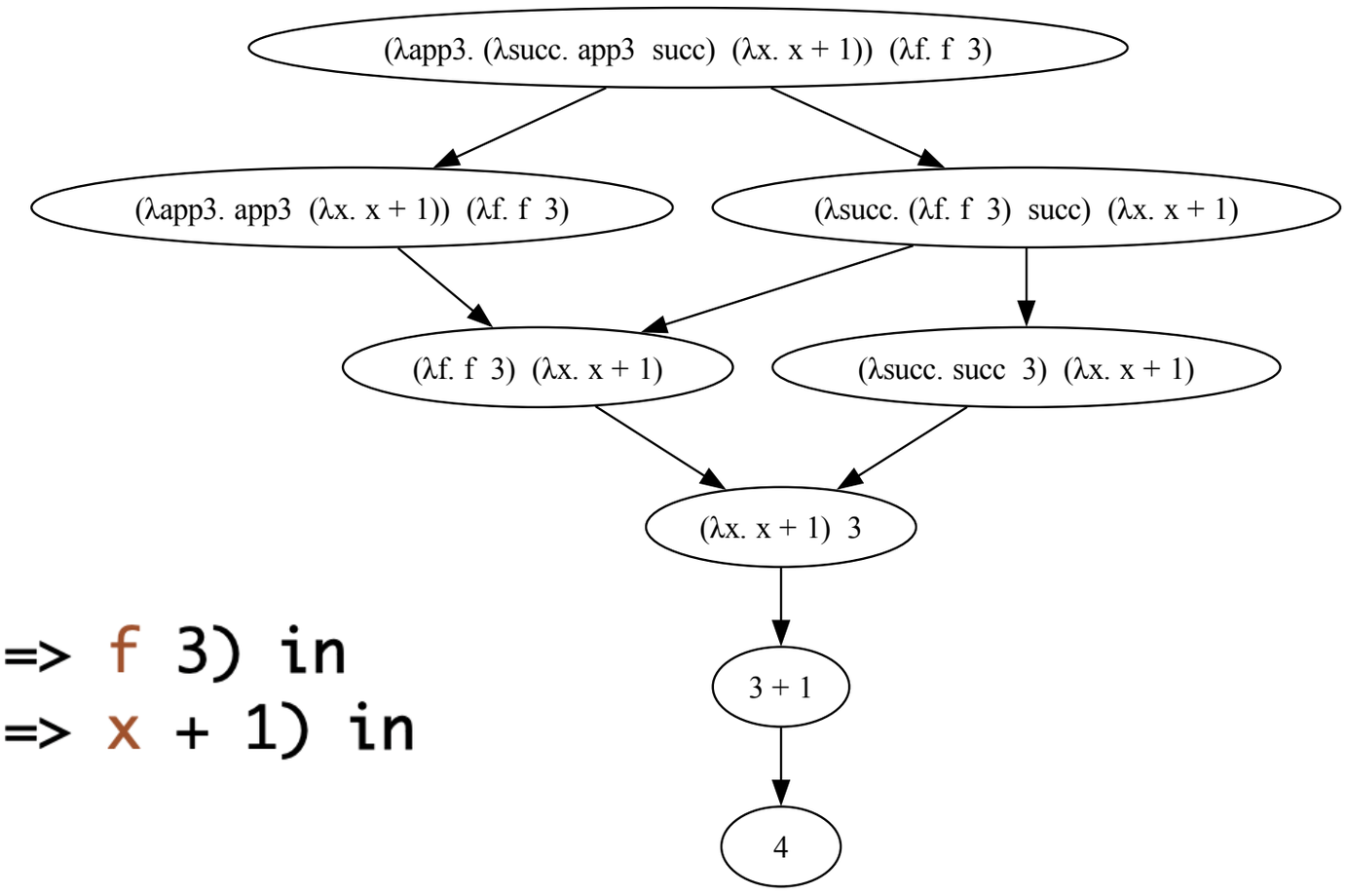
MM	L.	NOLIN	Président
	H	BARENDREGT	Examinateurs
	J.-Y.	GIRARD	
	G.	HUET	
	M.	NIVAT	
	J.-C.	SIMON	
	J.	VUILLEMIN	
	G.	KAHN	Invité

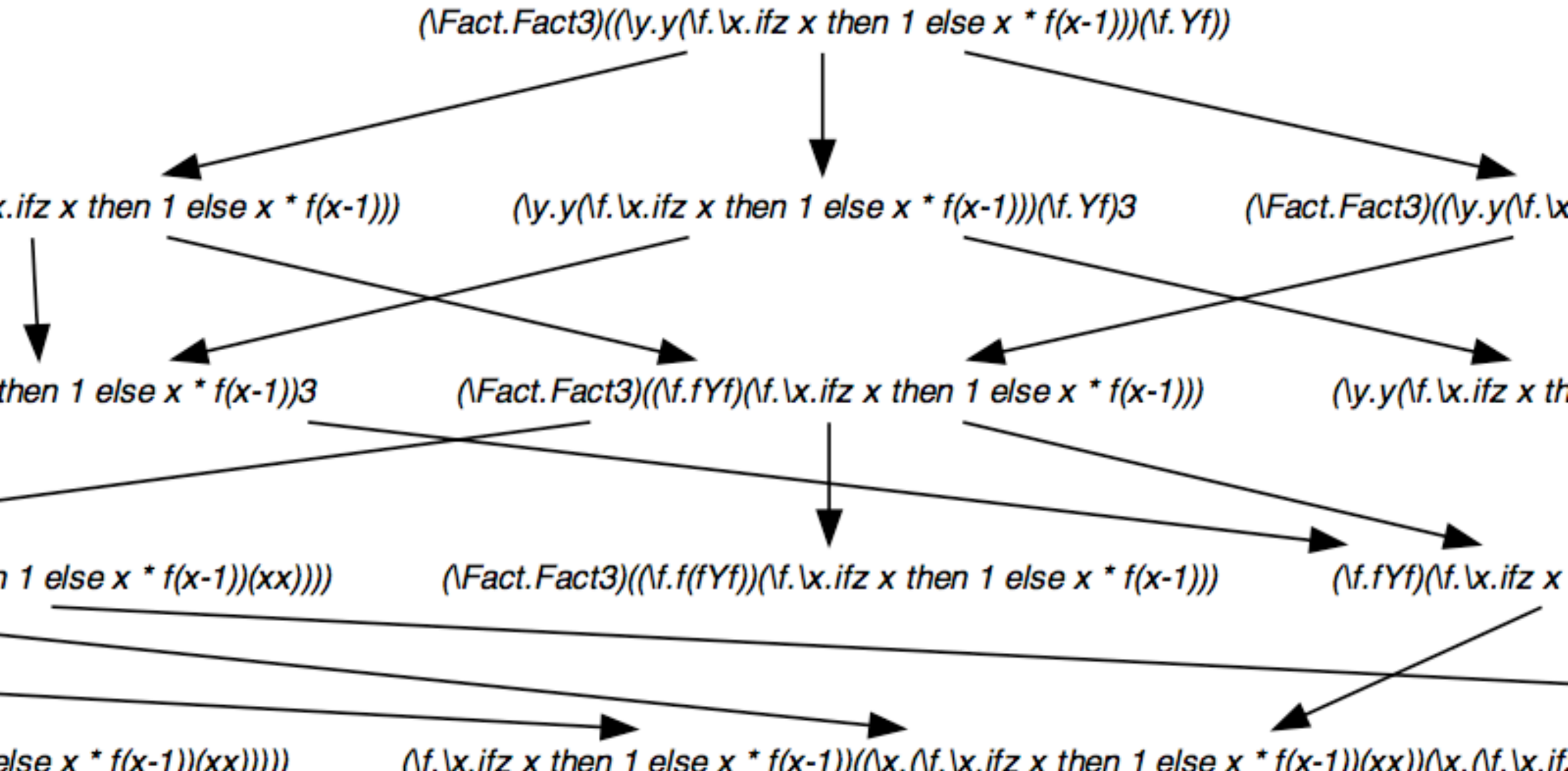
Tracking redexes in the lambda-calculus
"The French School of Programming"
ed. Bertrand Meyer, Springer, 2024.

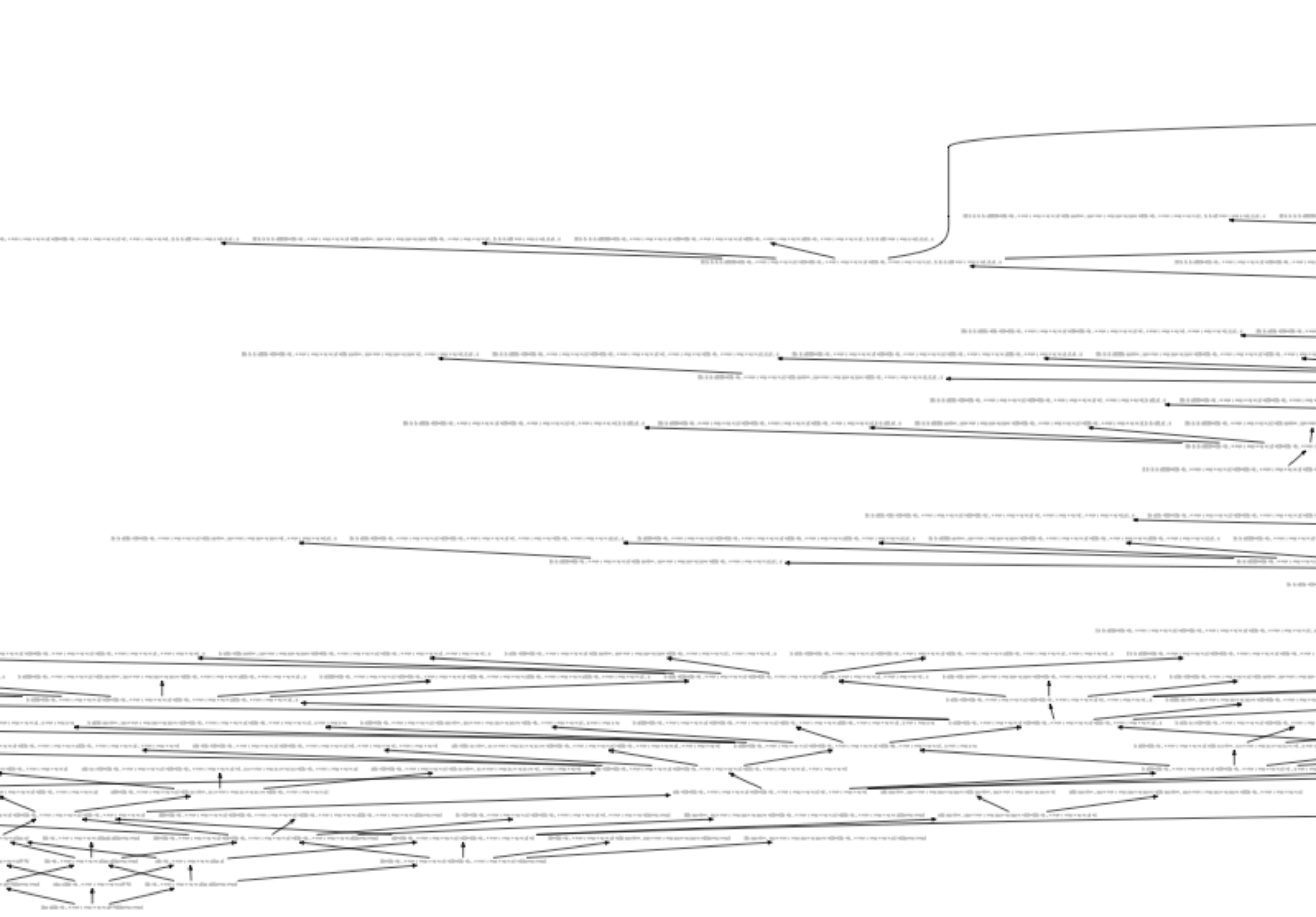
The lambda-calculus

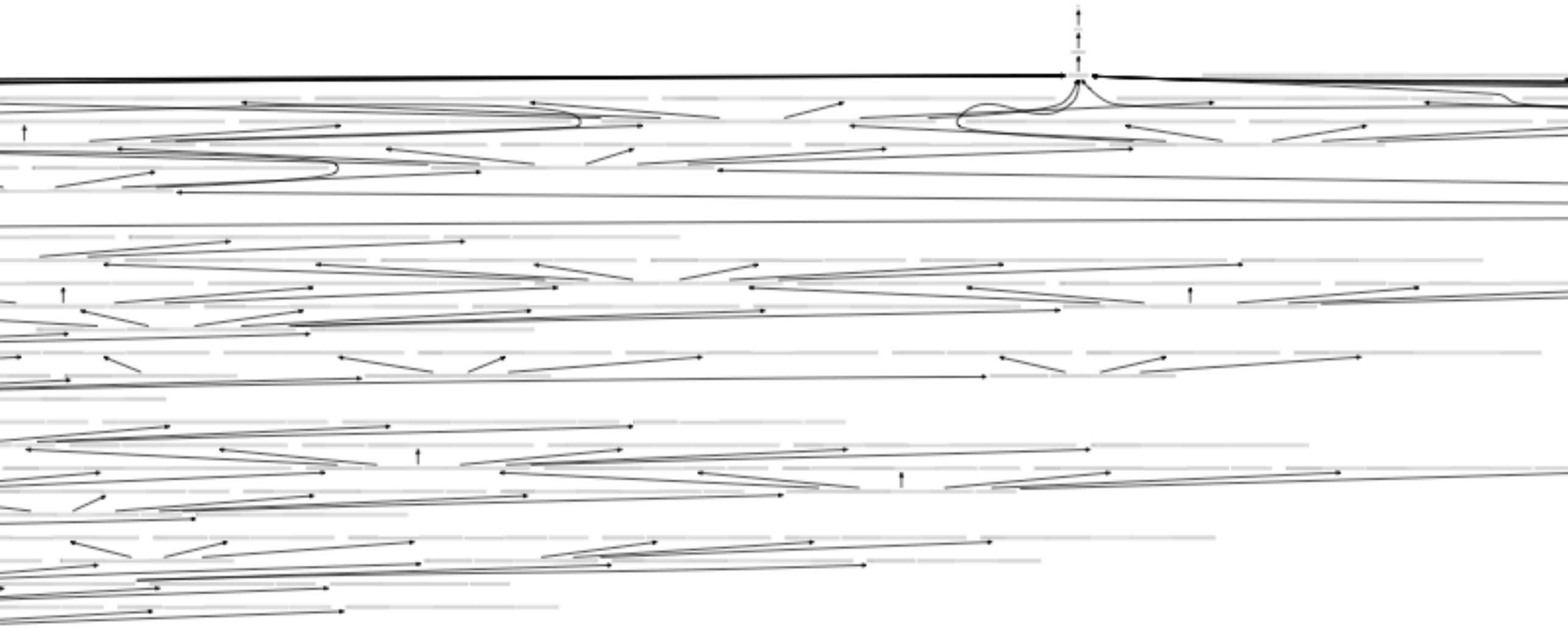
- calculus of functions with nested redexes and bound variables

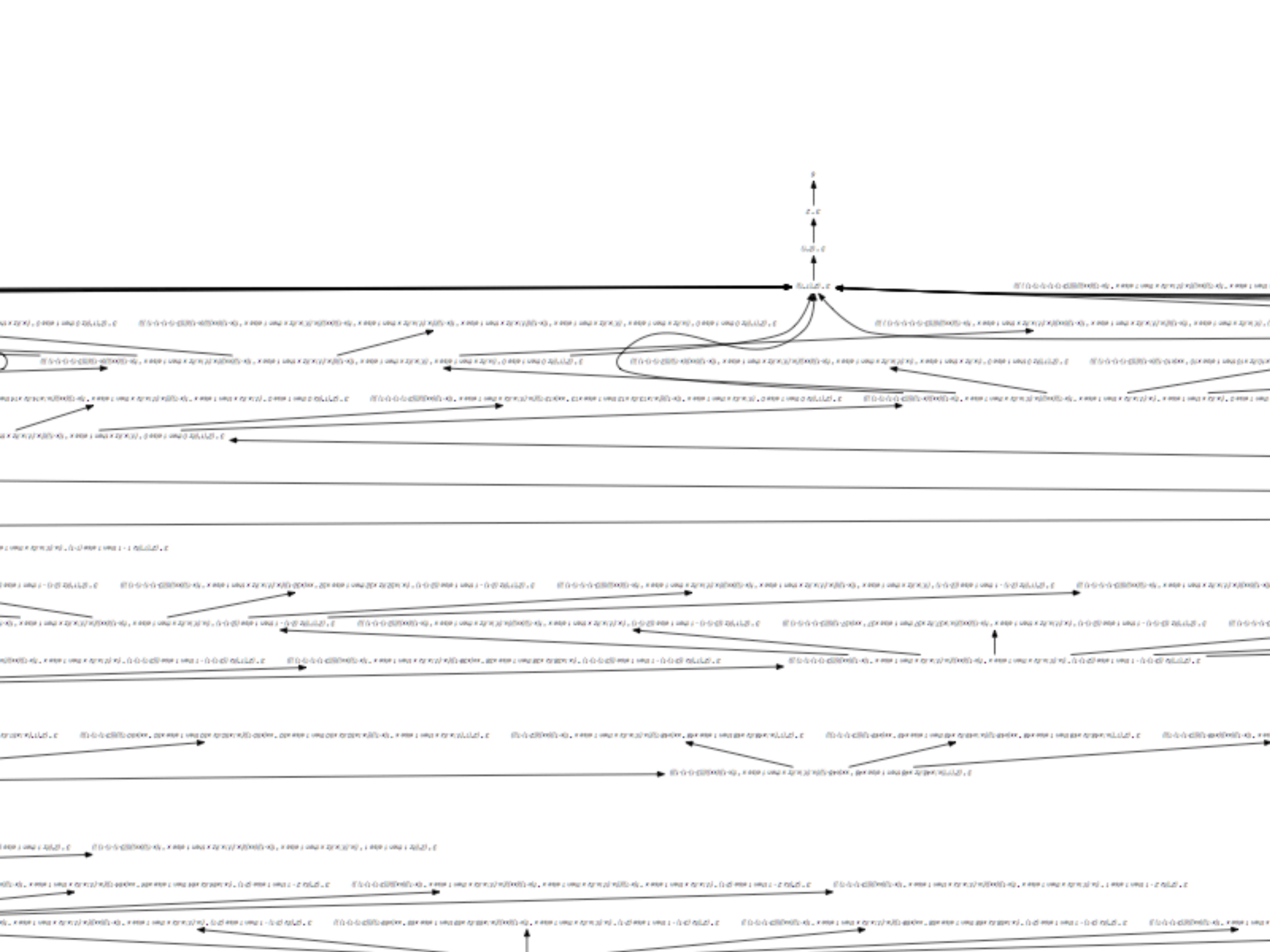
```
let app3 = (fun f => f 3) in
let succ = (fun x => x + 1) in
app3 succ
```

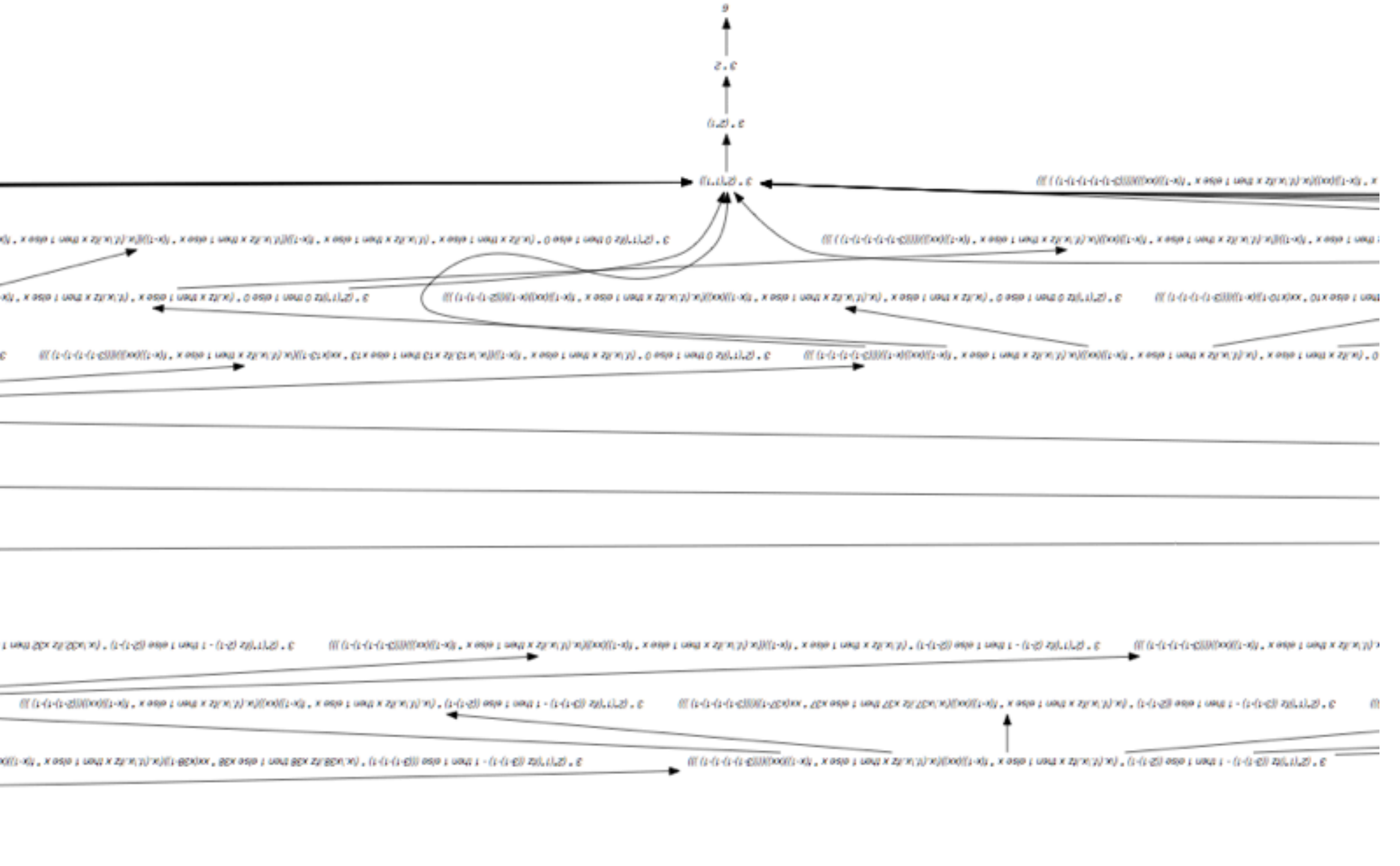


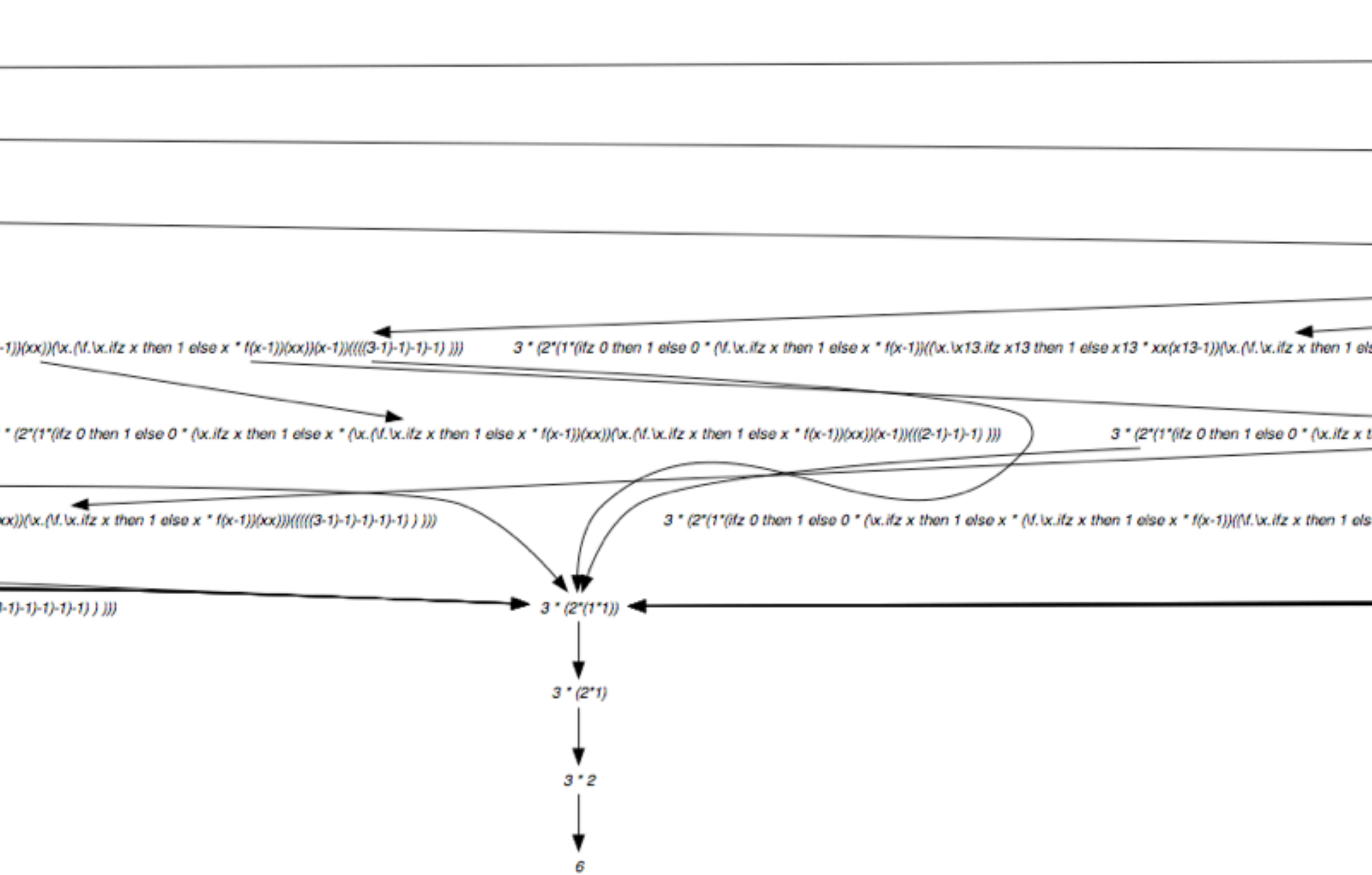

(\Fact.Fact3)((\y.y(\f.\x.ifz x then 1 else x * f(x-1)))(\f.Yf))
.ifz x then 1 else x * f(x-1)))
(\y.y(\f.\x.ifz x then 1 else x * f(x-1)))(\f.Yf)3
(\Fact.Fact3)((\y.y(\f.\x
then 1 else x * f(x-1))3
(\Fact.Fact3)((\f.fYf)(\f.\x.ifz x then 1 else x * f(x-1)))
(\y.y(\f.\x.ifz x th
n 1 else x * f(x-1))(xx))))
(\Fact.Fact3)((\f.f(fYf))(\f.\x.ifz x then 1 else x * f(x-1)))
(\f.fYf)(\f.\x.ifz x
else x * f(x-1))(xx)))))
(\f.\x.ifz x then 1 else x * f(x-1))((\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx))(\x.(\f.\x.if

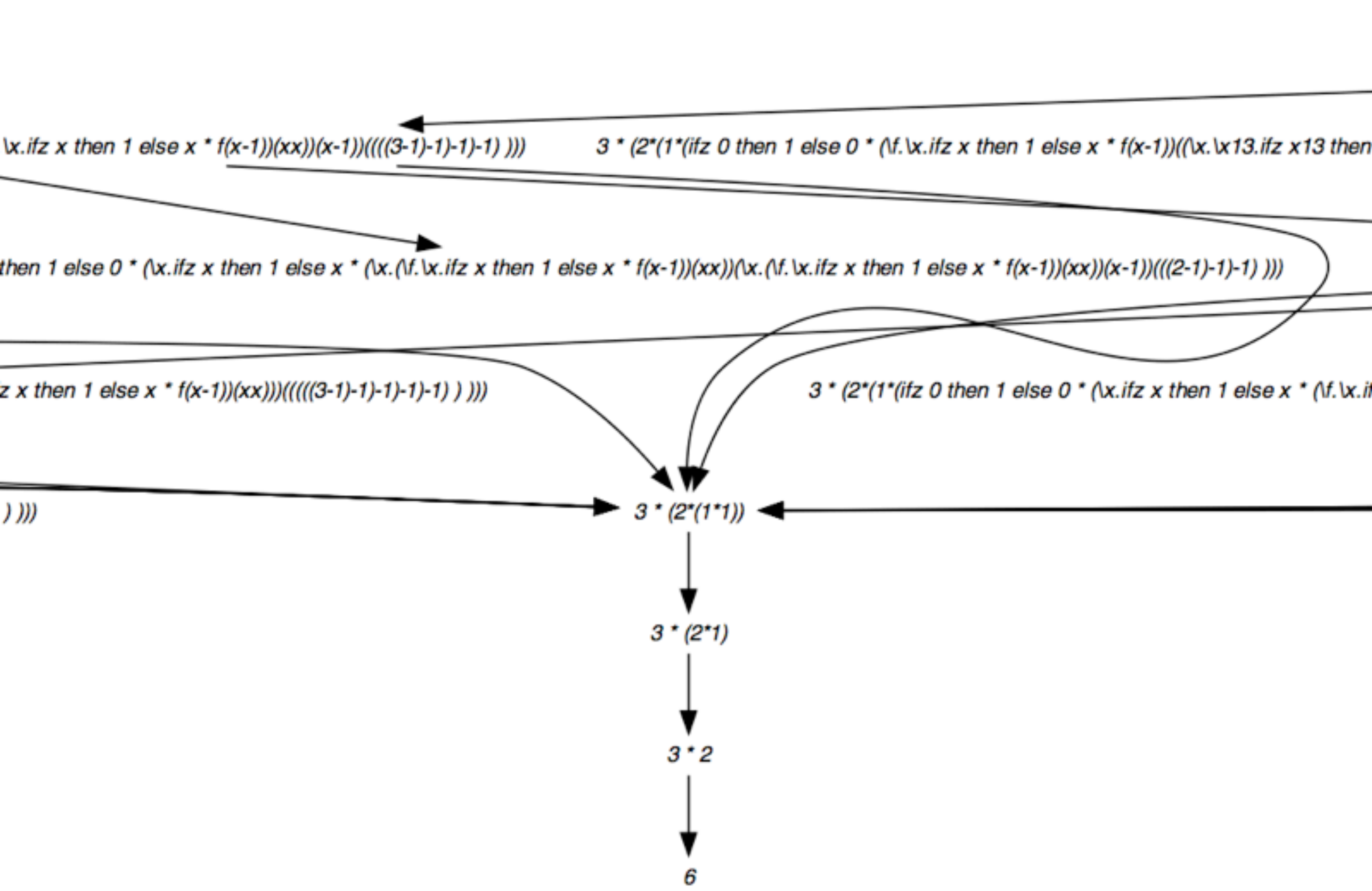







6
3 . 2
3 . (2^1)
3 . (2^1^1)


-1))(xx))(\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx))(x-1))((((3-1)-1)-1)-1)))) 3 * (2*(1*(ifz 0 then 1 else 0 * (\f.\x.ifz x then 1 else x * f(x-1))((\x.\x13.ifz x13 then 1 else x13 * xx(x13-1))(\x.(\f.\x.ifz x then 1 els
* (2*(1*(ifz 0 then 1 else 0 * (\x.ifz x then 1 else x * (\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx))(\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx))(x-1))(((2-1)-1)-1)))) 3 * (2*(1*(ifz 0 then 1 else 0 * (\x.ifz x t
xx))(\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx)))((((((3-1)-1)-1)-1)-1))))) 3 * (2*(1*(ifz 0 then 1 else 0 * (\x.ifz x then 1 else x * (\f.\x.ifz x then 1 else x * f(x-1))((\f.\x.ifz x then 1 els
-1)-1)-1)-1)-1)))))
3 * (2*(1*1))
3 * (2*1)
3 * 2
6


\x.ifz x then 1 else x * f(x-1))(xx))(x-1))((((3-1)-1)-1)-1))))
3 * (2*(1*(ifz 0 then 1 else 0 * (\f.\x.ifz x then 1 else x * f(x-1))((\x.\x13.ifz x13 then
then 1 else 0 * (\x.ifz x then 1 else x * (\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx))(\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx))(x-1))(((2-1)-1)-1))))
z x then 1 else x * f(x-1))(xx)))(((((3-1)-1)-1)-1)-1)))))
3 * (2*(1*(ifz 0 then 1 else 0 * (\x.ifz x then 1 else x * (\f.\x.it
))))
3 * (2*(1*1))
3 * (2*1)
3 * 2
6

Tracking redexes

- continuity of Bohm trees
- reversible computations
- flow analysis
- causality (event structures)
- incremental calculations (*makefile*)
- termination (strong normalization)
- reduction strategies (correctness, cost)
- and …

call-by-need

Initial motivation

- **call-by-value** : evaluate arguments of functions before application `[Ocaml]`
- **call-by-name** : apply arguments to functions without evaluating them `[Haskell]`
- **call-by-need** : 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

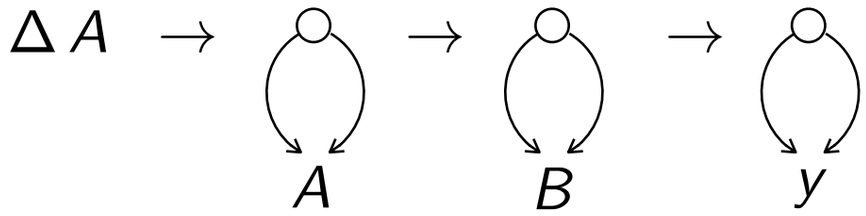
Initial motivation

- 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

Duplication of redexes

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



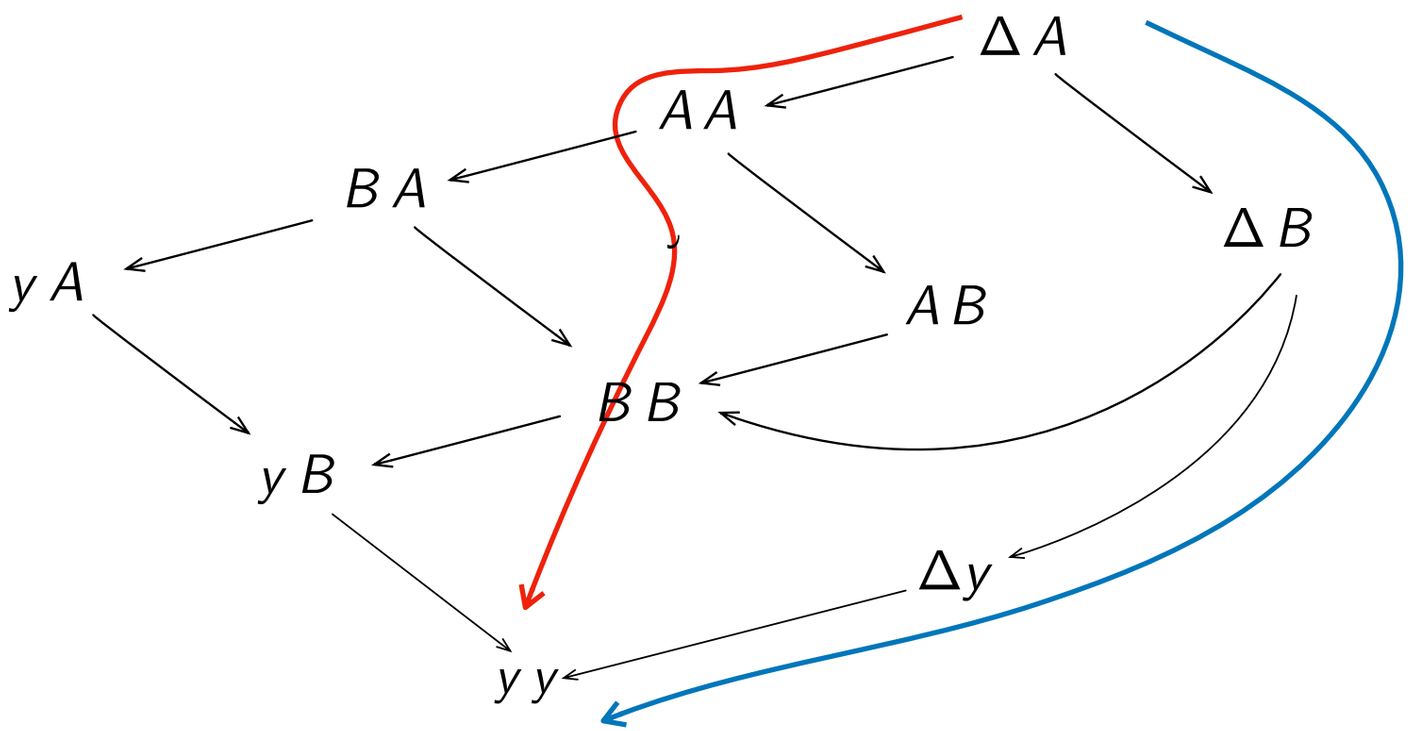
$\Delta = \lambda x.x\,x$

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

$I = \lambda x.x$

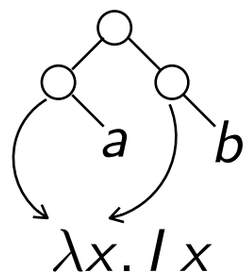
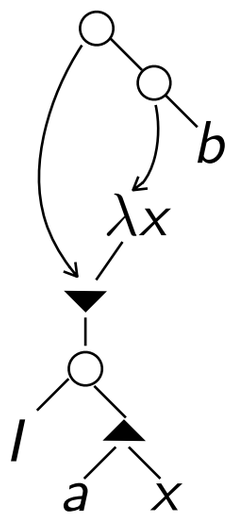
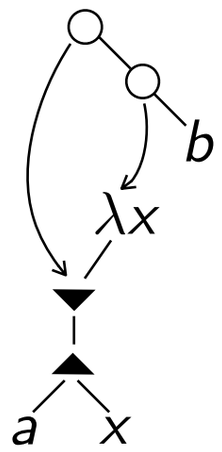
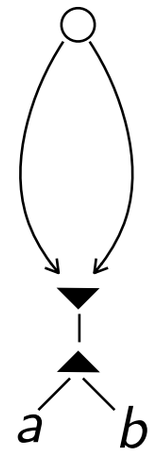
$A = F\,I$

$B = I\,y$

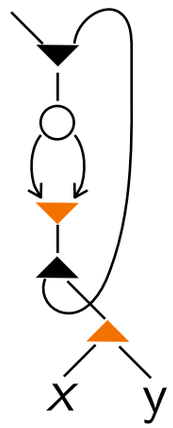


Duplication of redexes

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

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

- possible sharing !!



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

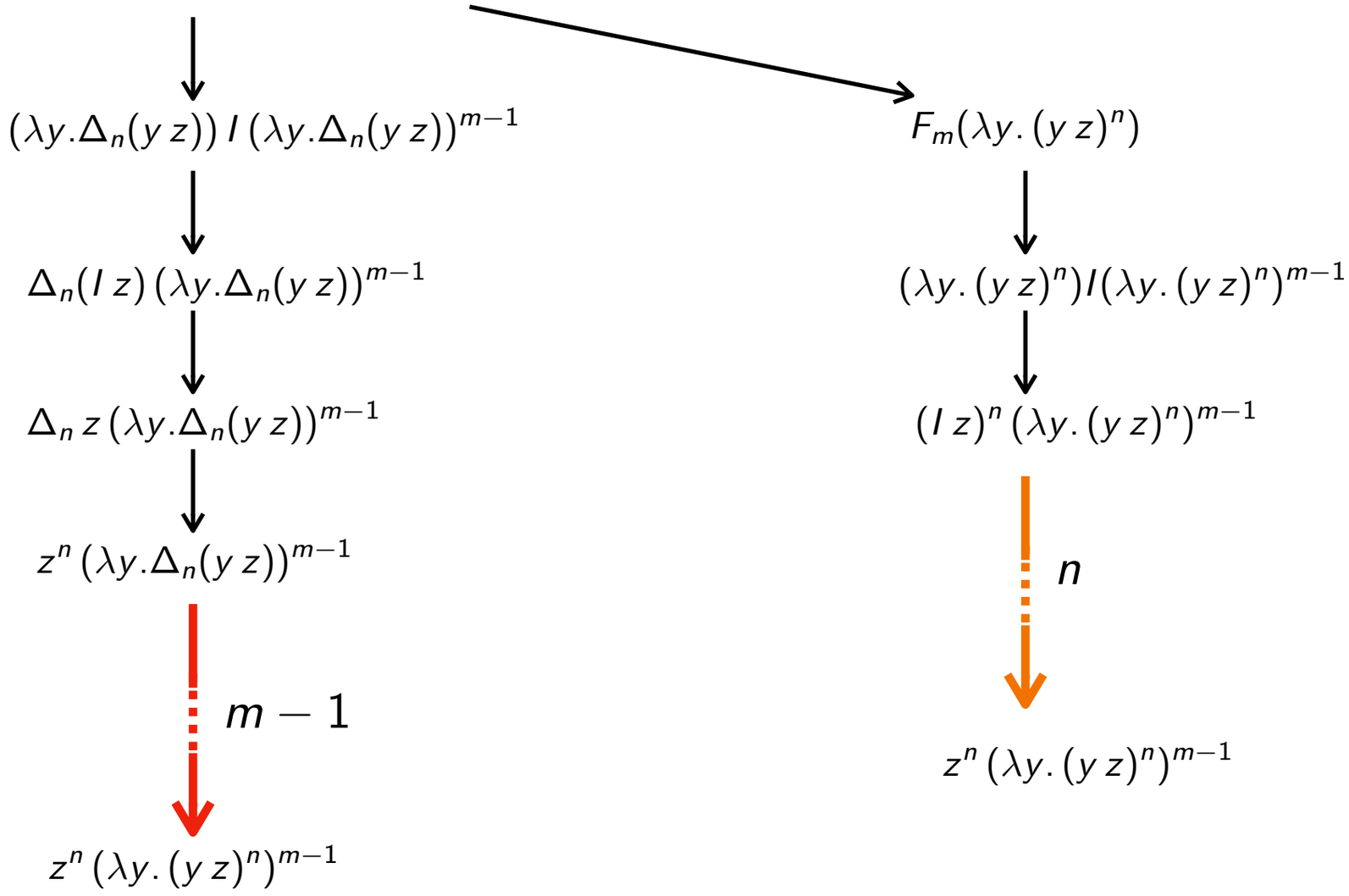
- sharing of contexts ? with boxes and tags ??

- back to easier theory …

Initial motivation (side remark)

- 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 $\Delta_n = \lambda x.\,x\,x\,\ldots\,x$



reduction steps and residuals

Back to context-free languages

- ambiguous grammar

$$S \to S + S$$
$$S \to a$$

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

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

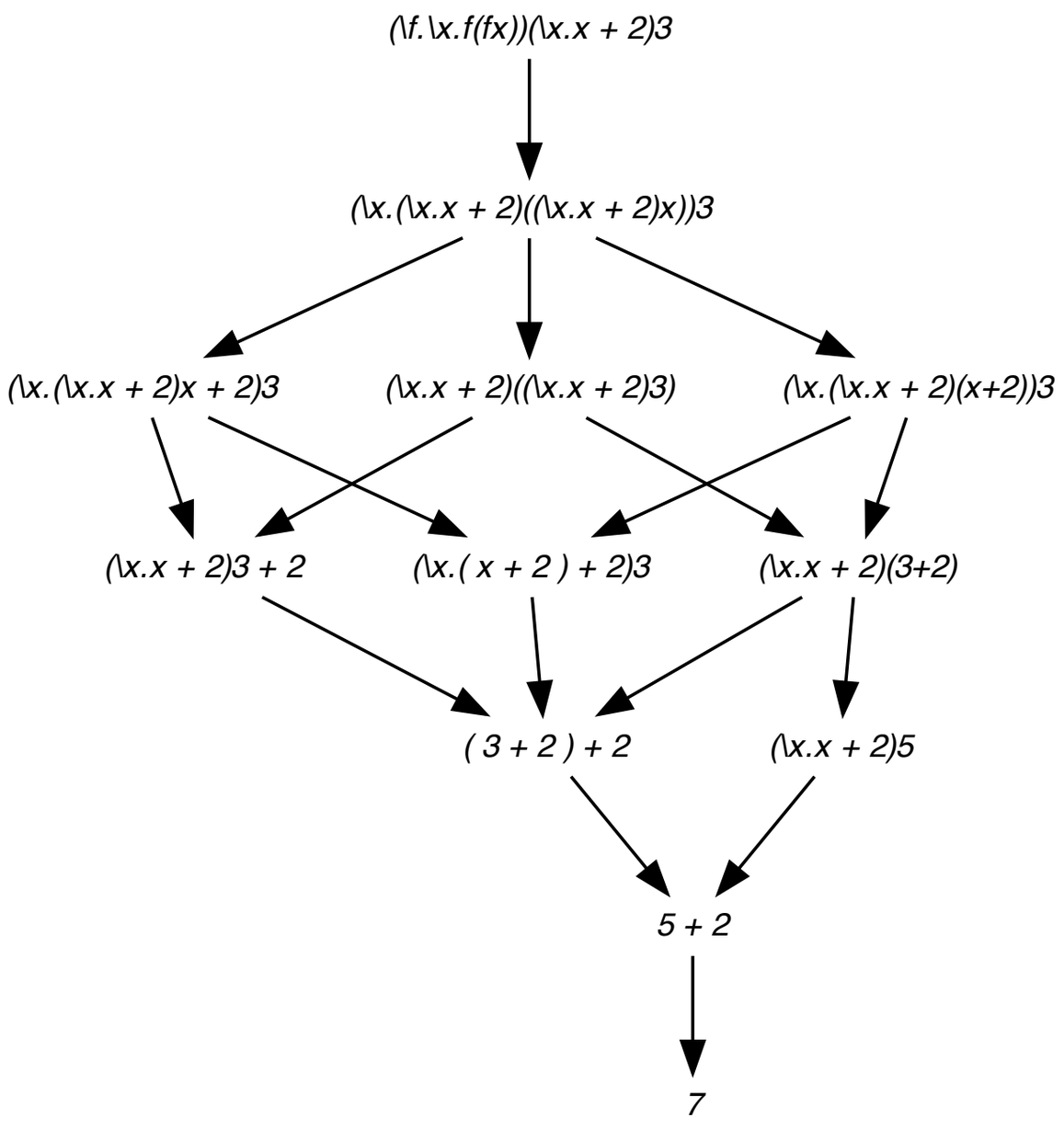
$$S \to S + S \to a + S \to a + S + S \to a + a + S \to a + a + a$$
$$S \to S + S \to a + S \to a + S + S \to a + S + a \to a + a + a$$
$$S \to S + S \to S + S + S \to a + S + S \to a + a + S \to a + a + a$$
$$S \to S + S \to S + S + S \to S + a + S \to a + a + S \to a + a + a$$
$$\ldots$$

- only 1 leftmost derivation per parse tree

The lambda-calculus

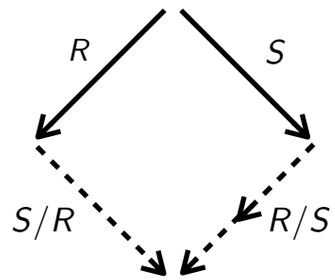
- many permutations of redex contractions

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



The lambda-calculus

- **local confluence**



- and 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

$$\underline{(I\,x)}\,(I\,x) \;\to\; \underline{(I\,x)}\,x \qquad\qquad I\,x\,\underline{(I\,x)} \;\to\; x\,\underline{(I\,x)}$$

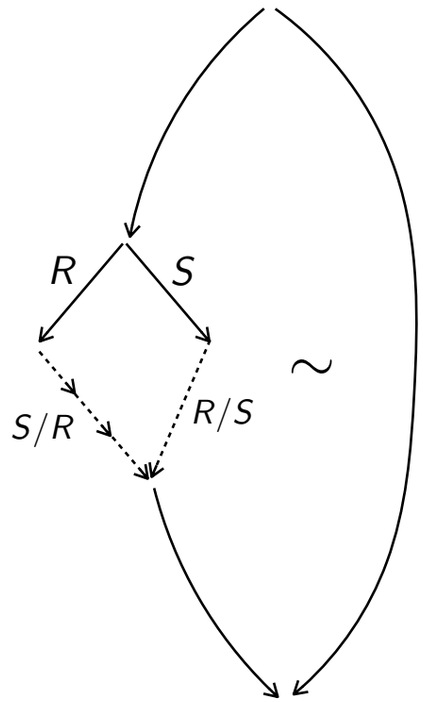
$$\Delta\,\underline{(I\,x)} \;\to\; \underline{(I\,x)}\;\underline{(I\,x)} \qquad\qquad (\lambda x.I\,x\,\underline{(I\,x)})\,y \;\to\; I\,y\,\underline{(I\,y)}$$

$$\underline{\Delta\,(I\,x)} \;\to\; \underline{(\Delta\,x)} \qquad\qquad \underline{(I\,x)} \;\to\; x$$

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

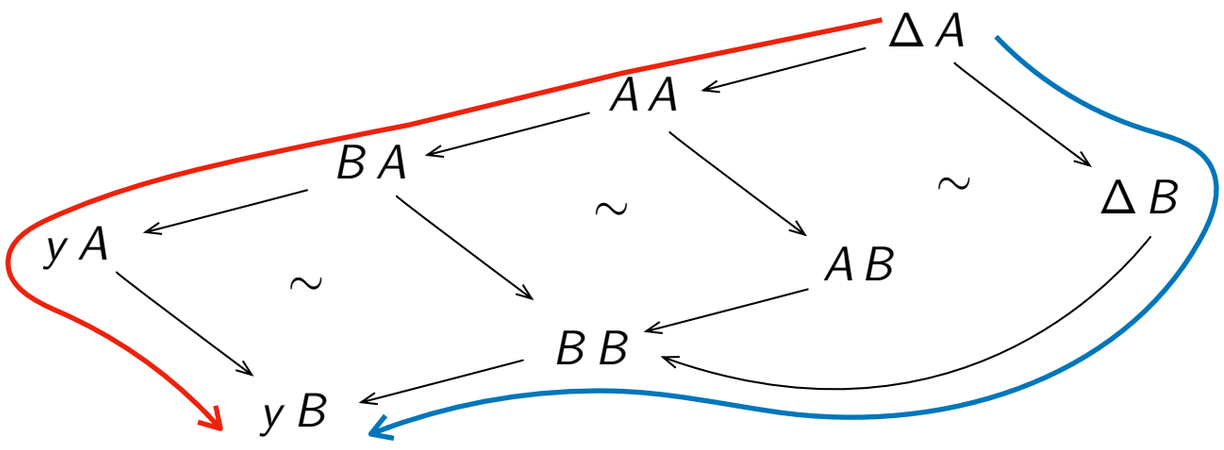
Equivalence by permutations

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



Equivalence by permutations

- example



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

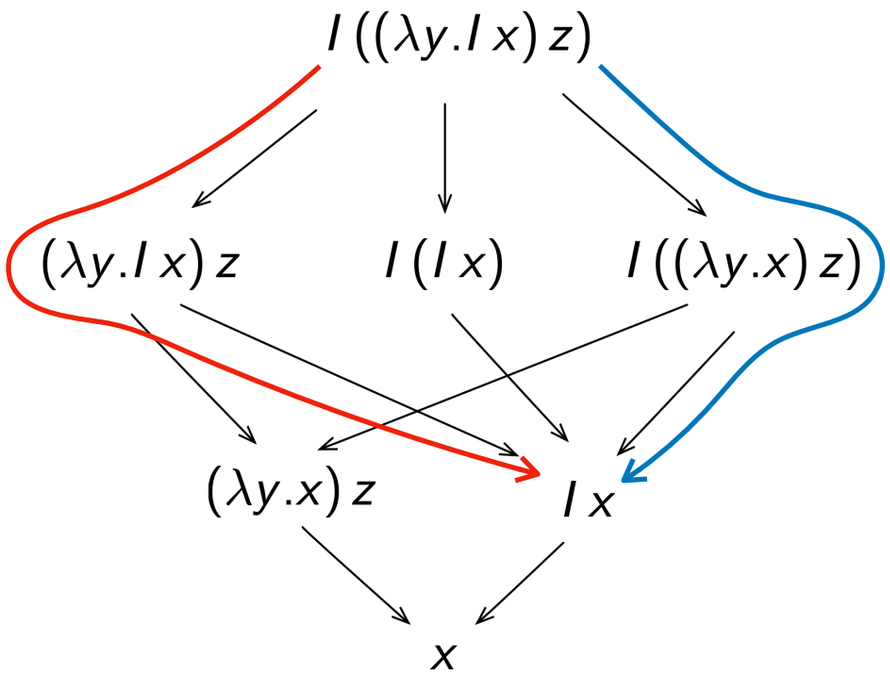
$F = \lambda f.f\,y$

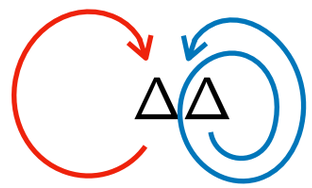
$I = \lambda x.x$

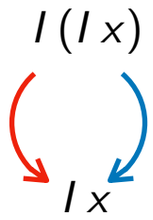
$A = F\,I$

$B = I\,y$

- counter-examples





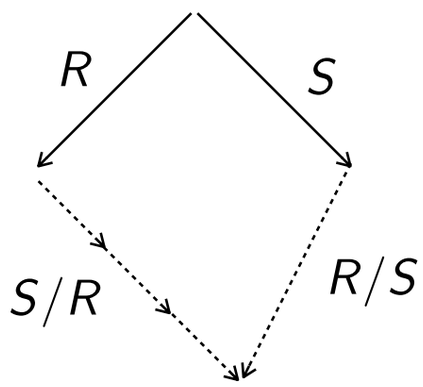


permutation equivalence

Permutation equivalence

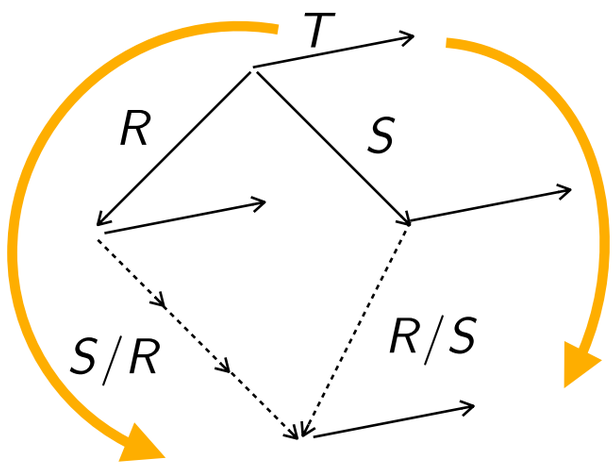
- single-redex reduction steps $M \xrightarrow{R} N$
- residuals S/R of another redex S in M are **disjoint** redexes
  - let $\mathcal{F}$ be a set of disjoint redexes
  - write $\rho : \mathcal{F}$ for any single-redex reduction ρ of redexes of $\mathcal{F}$ in any order
  - these reductions are all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

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

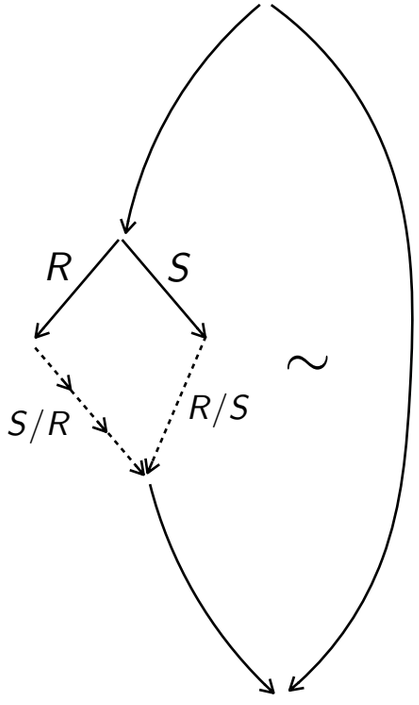
Permutation equivalence

- definition with permutations

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

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

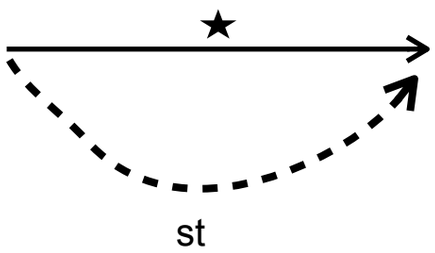
$$(ii) \quad \rho \sim \sigma \implies \tau;\rho;\upsilon \sim \tau;\sigma;\upsilon$$



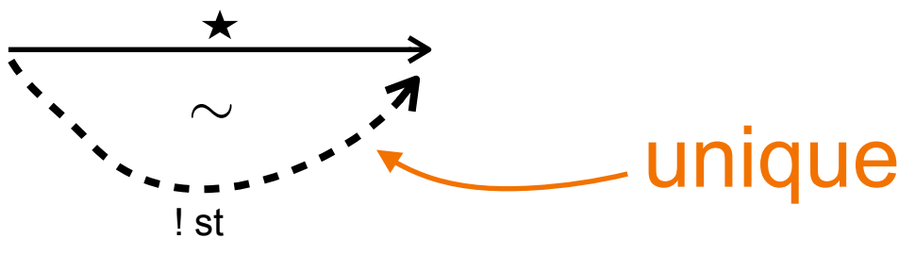
aka parse trees for contex-free languages

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



λ-calculus with permutations

- standard reductions are canonical representatives in equivalence classes

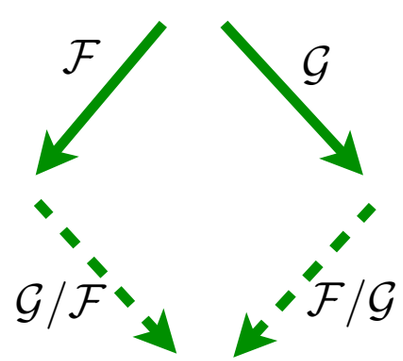
Finite developments

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

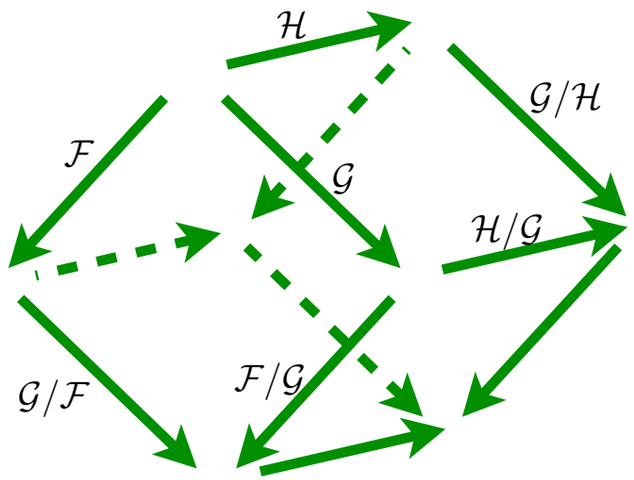
finite development thm [Curry] let $\mathcal{F}$ be a set of redexes in a term M

- 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 $\mathcal{F}$

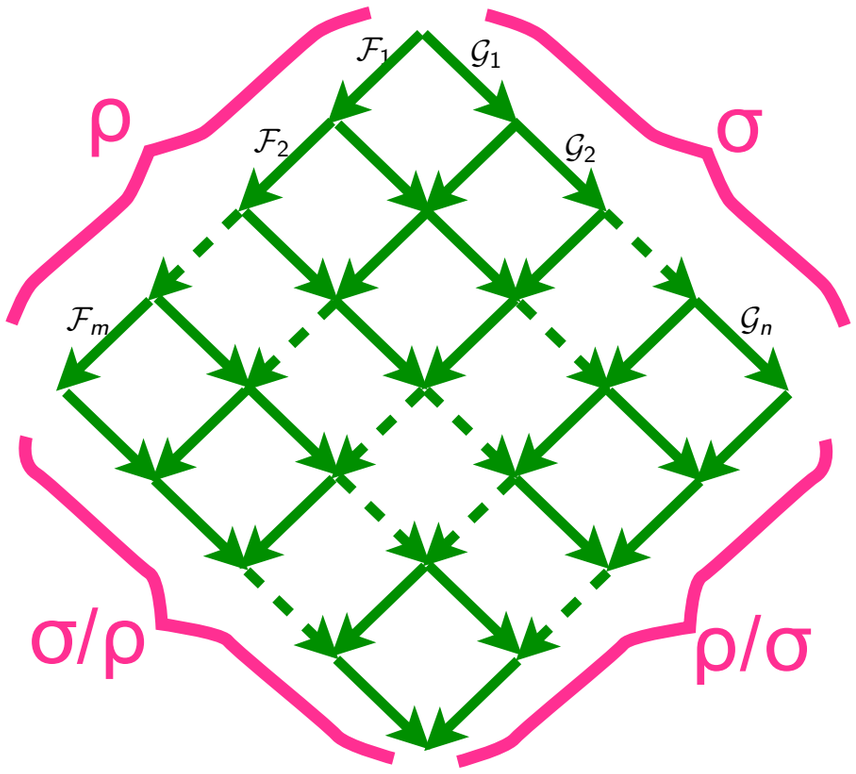
corollary: parallel moves [Curry]

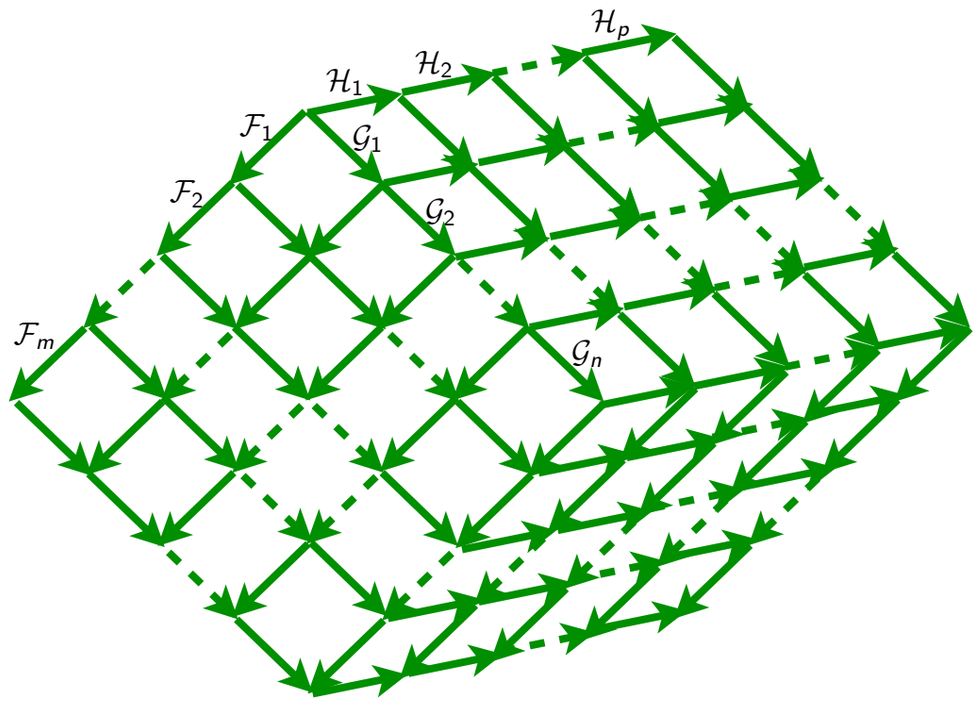


cube lemma



Residual of reductions





parallel moves +

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

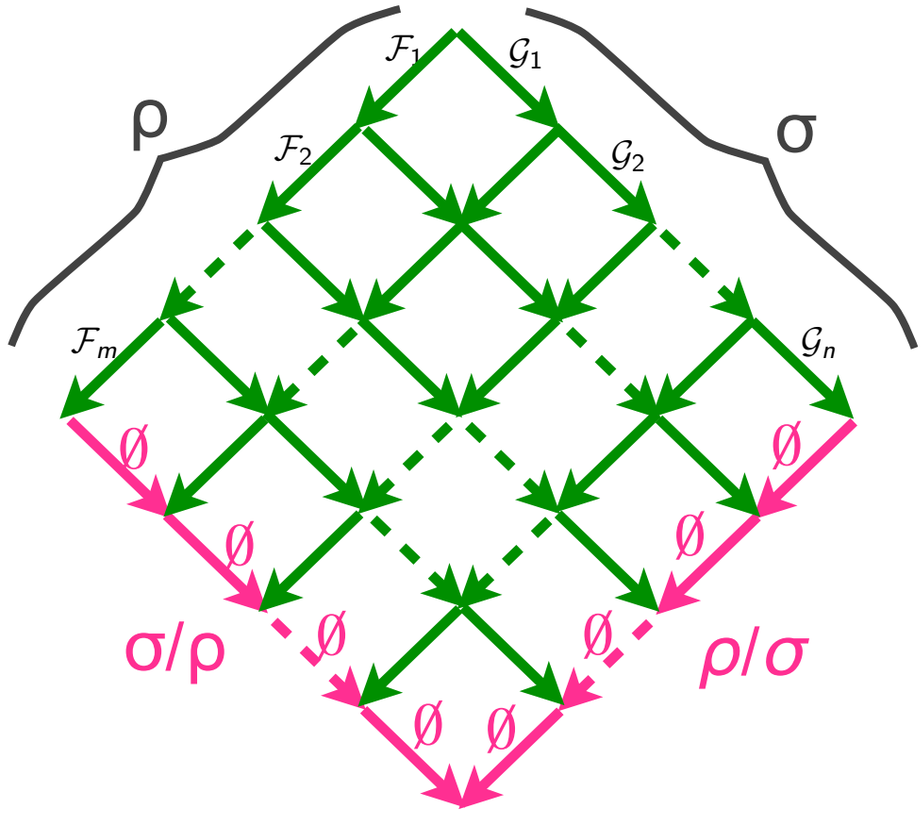
cube lemma +

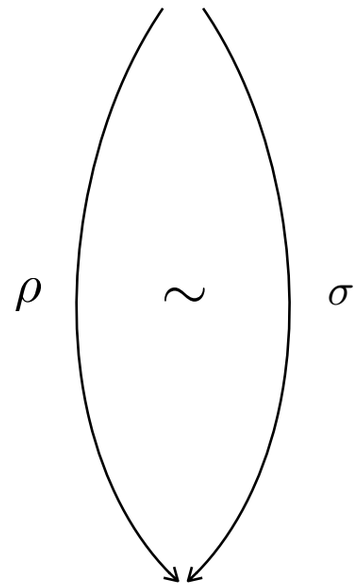
$\tau/(\rho \sqcup \sigma) = \tau/(\sigma \sqcup \rho)$

Permutation equivalence

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

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



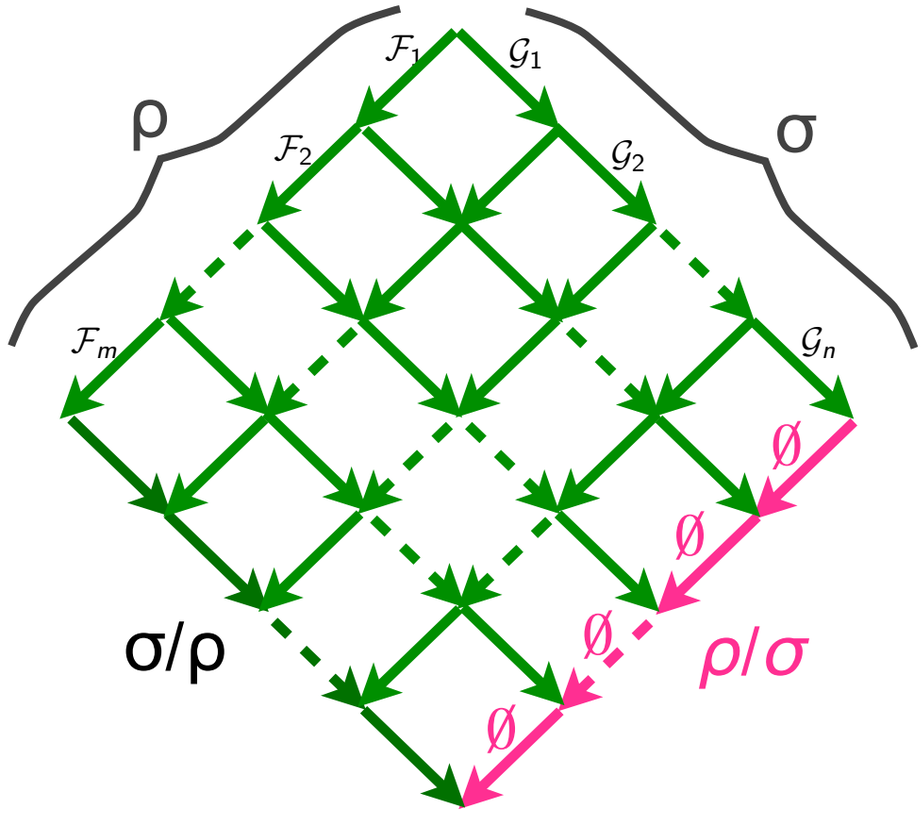


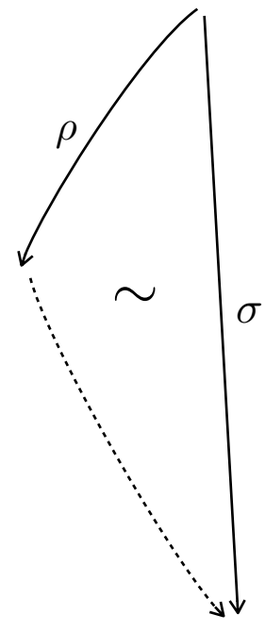
exercise Test previous example and counterexamples for permutation equivalence

Prefix modulo permutations

prefix modulo permutations Let ρ and σ be coinitial reductions.

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

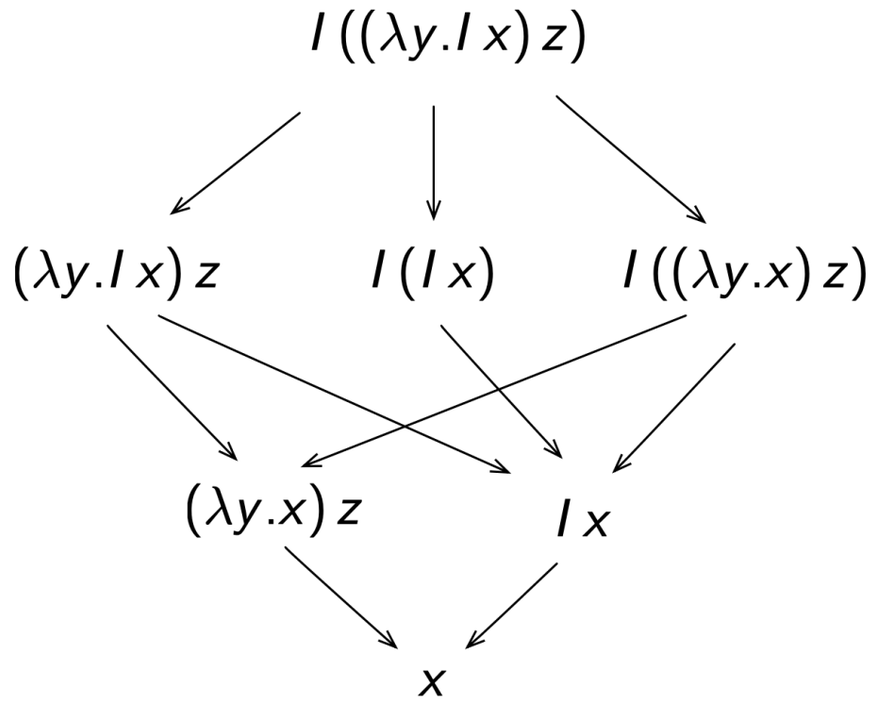


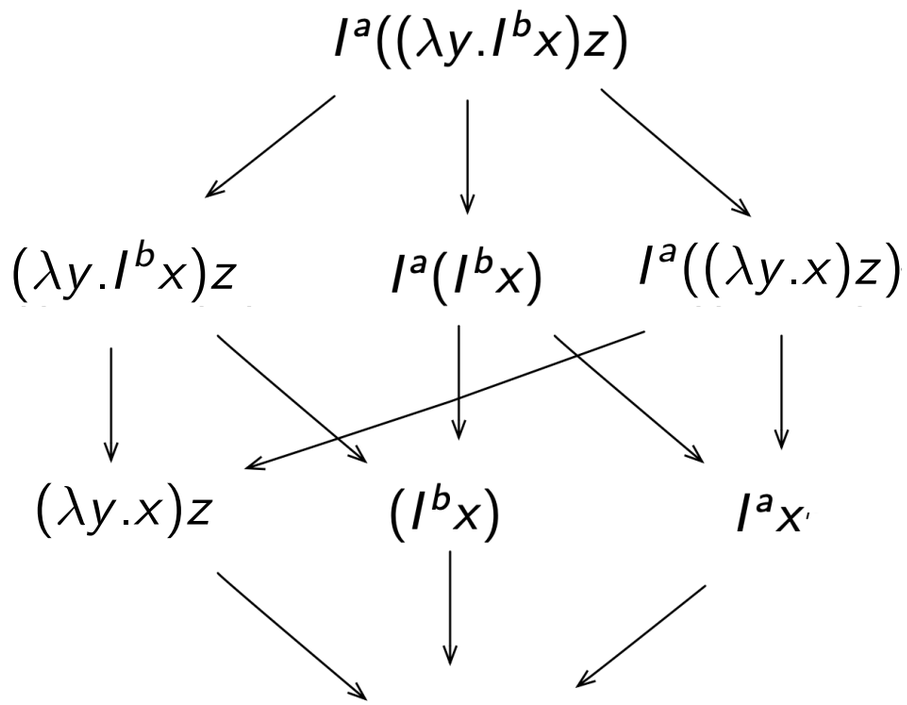


exercise Prove $\rho \leq \sigma$ iff $\exists \tau,\ \rho\,;\tau \sim \sigma$

Prefix modulo permutations

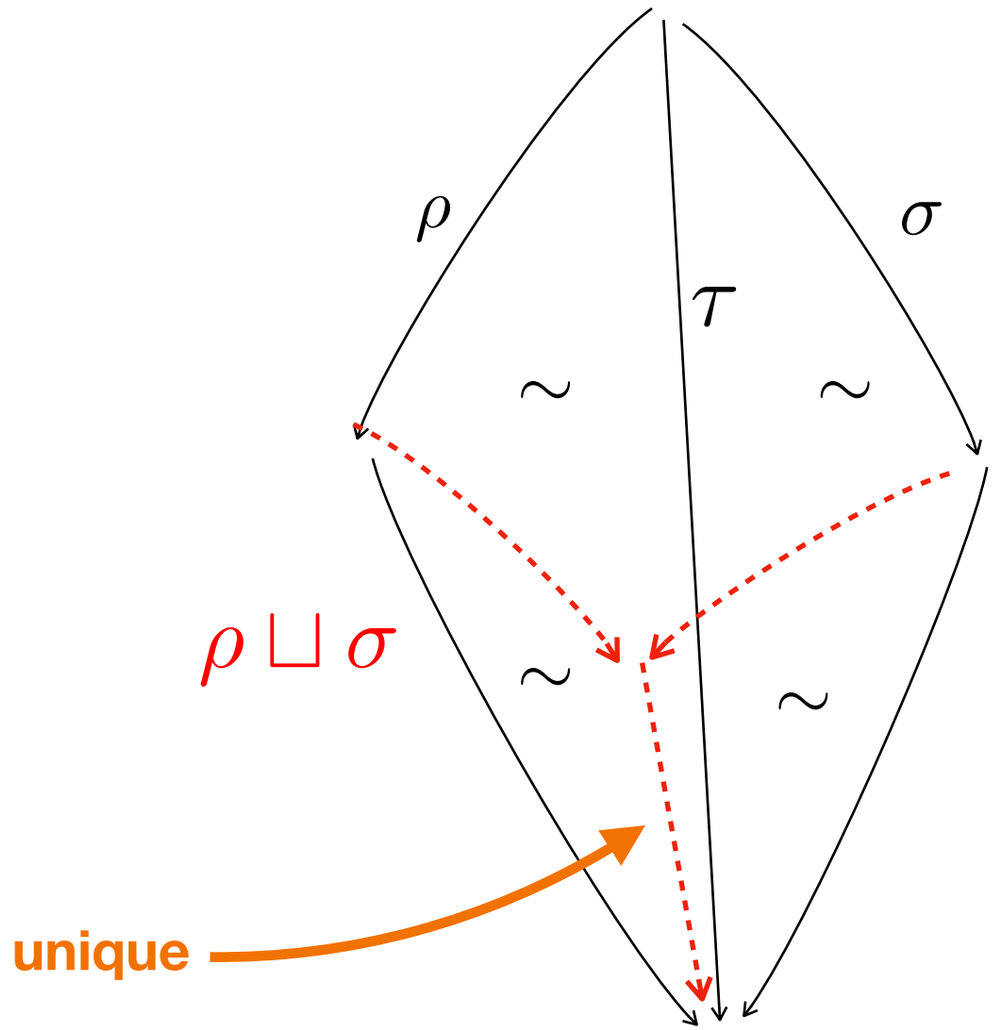
lattice of prefix modulo permutations





Prefix modulo permutations

lattice of prefix modulo permutations



Easy lemmas

properties of permutations

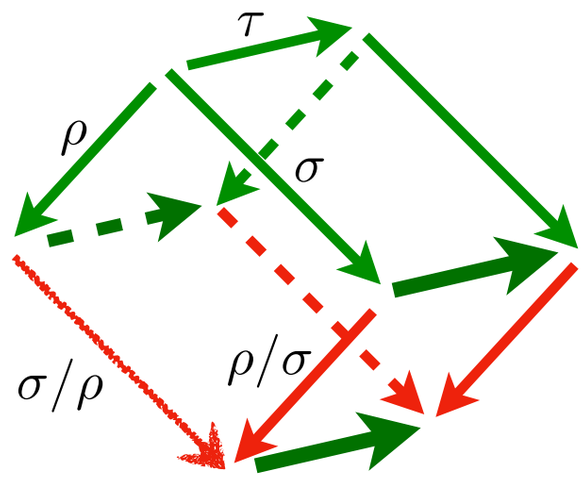
(i) $\rho \sim \sigma \iff \forall \tau . \tau/\rho = \tau/\sigma$

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

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

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

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



proof

(i) $\rho \sim \sigma \implies \sigma/\rho = \emptyset^n$ and $\rho/\sigma = \emptyset^m$
Thus $\tau/(\rho \sqcup \sigma) = \tau/(\rho;(\sigma/\rho)) = \tau/\rho/(\sigma/\rho) = \tau/\rho/\emptyset^n = \tau/\rho$
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) $\quad \rho \leq \sigma \leq \tau \implies \rho \leq \tau$

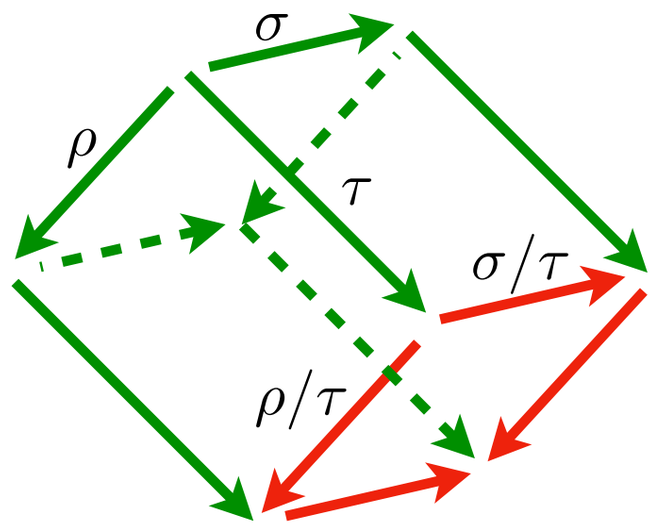
(iii) $\quad \rho \leq \rho \sqcup \sigma$ and $\sigma \leq \rho \sqcup \sigma$

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

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

proof

$\rho \leq \tau$ 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]

redex and history

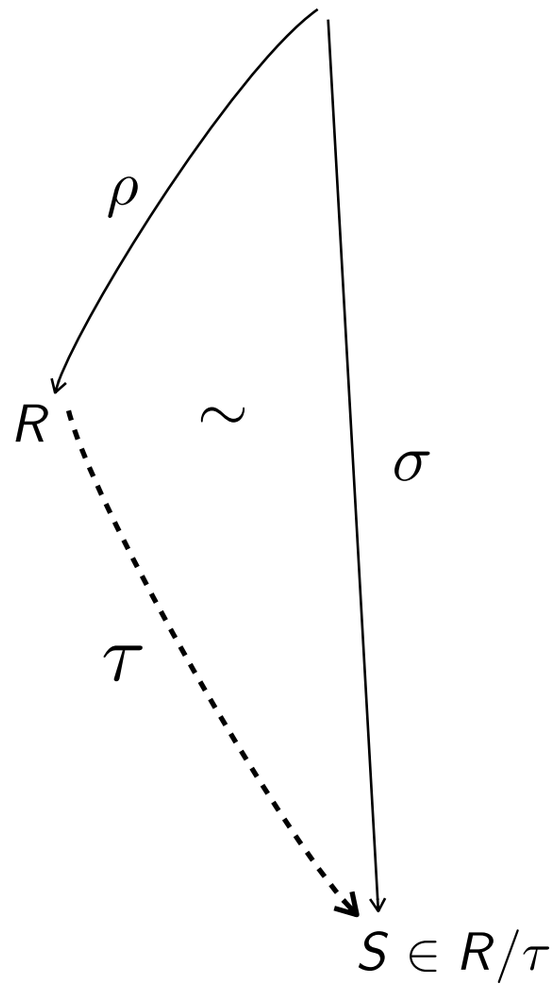
Initial motivation (bis)

- 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

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 $\rho; \tau \sim \sigma \ \wedge \ S \in R/\tau$



Residuals modulo permutations

- properties of residuals of h-redexes

$$(i) \quad \langle \rho, R \rangle \lesssim \langle \sigma, S \rangle \iff \rho \leq \sigma \ \wedge \ S \in R/(\sigma/\rho)$$

$$(ii) \quad \langle \rho, R \rangle \lesssim \langle \rho, R \rangle$$

$$(iii) \quad \langle \rho, R \rangle \lesssim \langle \sigma, S \rangle \lesssim \langle \rho, R \rangle \iff \rho \sim \sigma \ \wedge \ R = S$$

$$(iv) \quad \langle \rho, R \rangle \lesssim \langle \sigma, S \rangle \lesssim \langle \tau, T \rangle \implies \langle \rho, R \rangle \lesssim \langle \tau, T \rangle$$

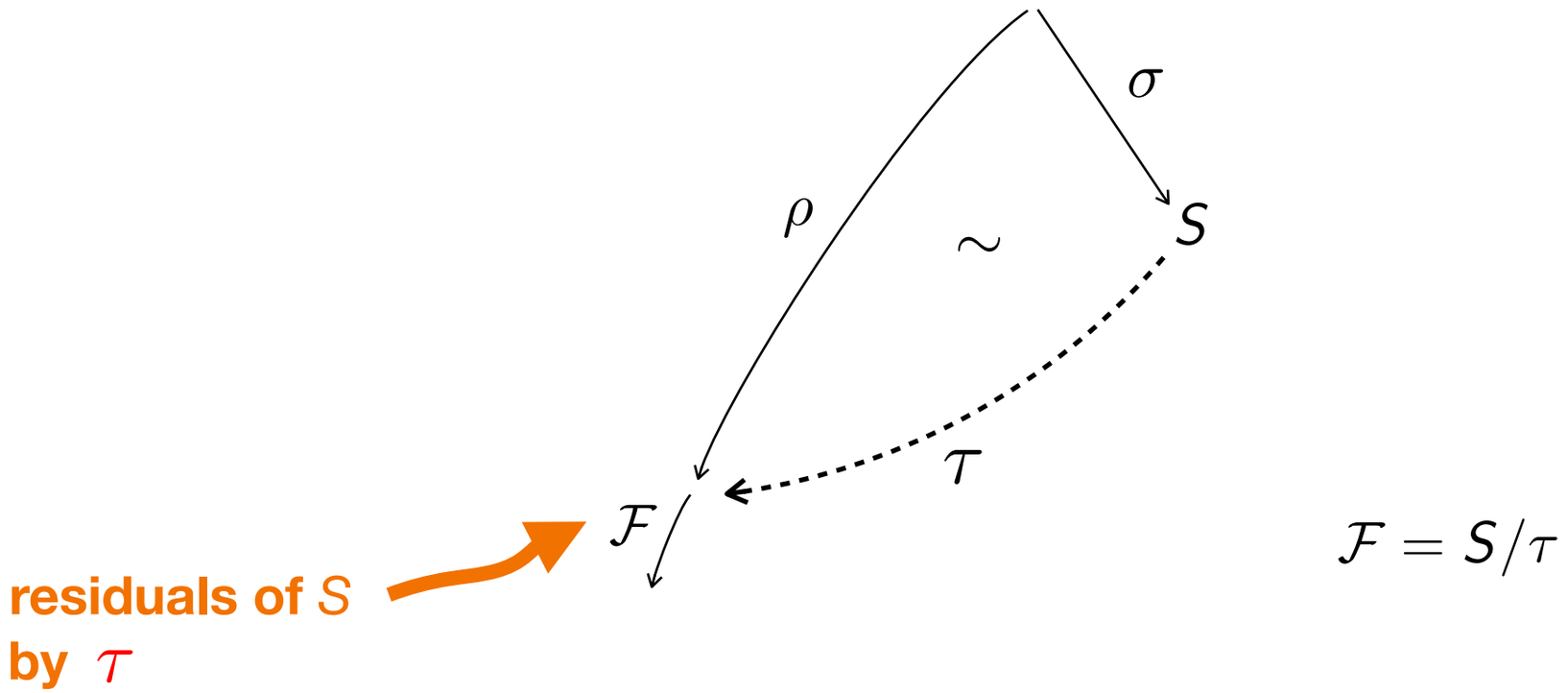
$$(v) \quad \langle \rho, R \rangle \lesssim \langle \tau, T \rangle \ \wedge \ \rho \leq \sigma \leq \tau \implies \exists! S, \ \langle \rho, R \rangle \lesssim \langle \sigma, S \rangle \lesssim \langle \tau, T \rangle$$

$$(vi) \quad \langle \rho, R \rangle \lesssim \langle \sigma, S \rangle \iff \langle \tau; \rho, \ R \rangle \lesssim \langle \tau; \sigma, \ S \rangle$$

- 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



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

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

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

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

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

The labeled λ-calculus

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

- their labeled λ-calculus [Asperti-Laneve]

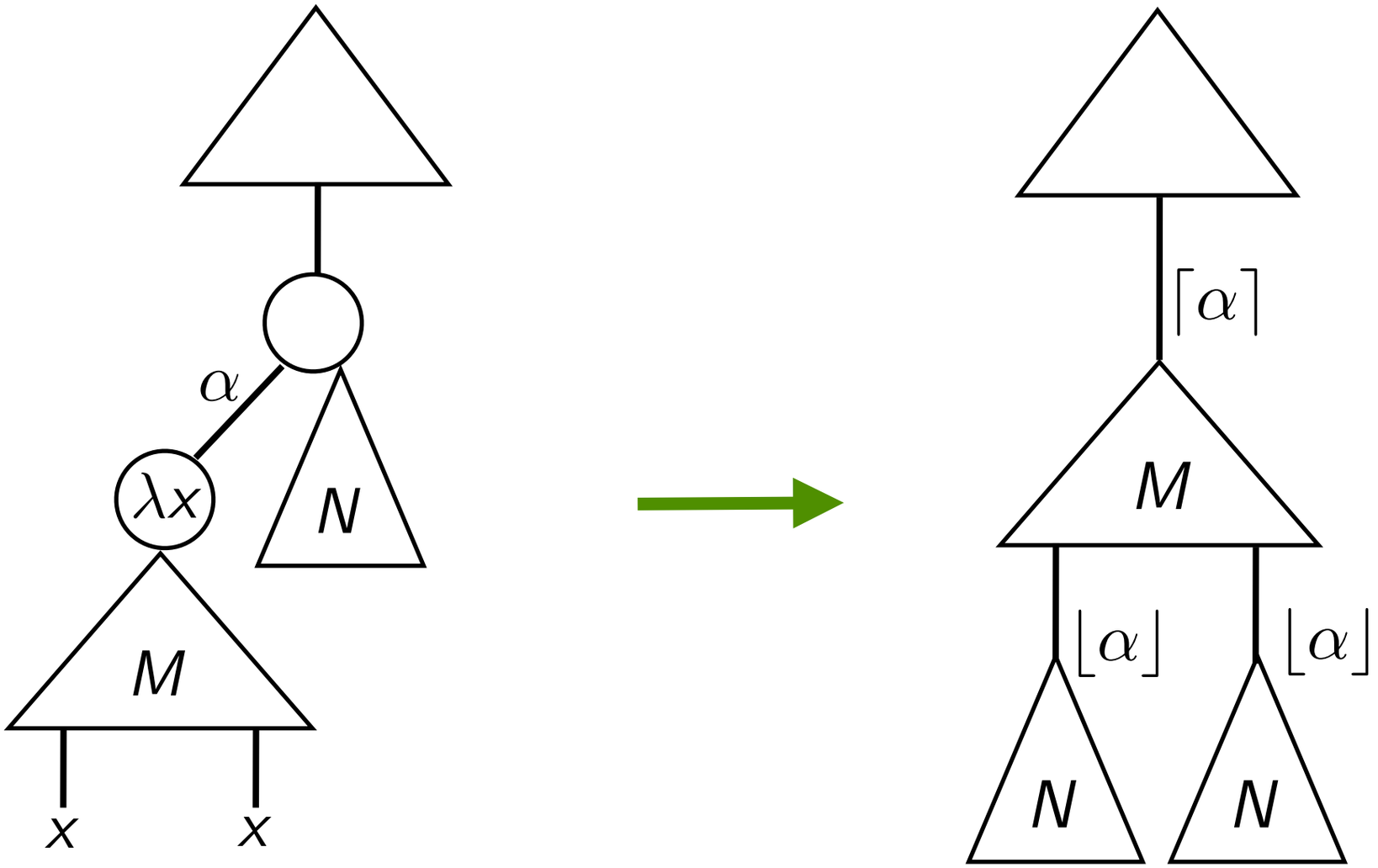
$$M, N, \ldots ::= x \mid MN \mid \lambda x.M \mid a : M$$

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

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

- correspondence with paths in initial term

The labeled λ-calculus



The labeled λ-calculus

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

$$\Delta^a \Delta \to (\Delta^{\gamma_1} \Delta^{\gamma_1})^{\lceil a \rceil} \to (\Delta^{\gamma_2} \Delta^{\gamma_2})^{\lceil \gamma_1 \rceil \lceil a \rceil} \to \dots$$

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

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

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

$$\Omega = \Delta^a\, \Delta$$
$$\to \Omega_1 = (\Delta^{\gamma_1}\, \Delta^{\gamma_1})^{\lceil a\rceil} \qquad \gamma_1 = \lfloor a\rfloor$$
$$\to \Omega_2 = (\Delta^{\gamma_2}\, \Delta^{\gamma_2})^{\lceil\gamma_1\rceil\lceil a\rceil} \qquad \gamma_2 = \gamma_1\lfloor\gamma_1\rfloor$$
$$\to \Omega_3 = (\Delta^{\gamma_3}\, \Delta^{\gamma_3})^{\lceil\gamma_2\rceil\lceil\gamma_1\rceil\lceil a\rceil} \qquad \gamma_3 = \gamma_2\lfloor\gamma_2\rfloor$$
$$\to \cdots$$

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

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

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

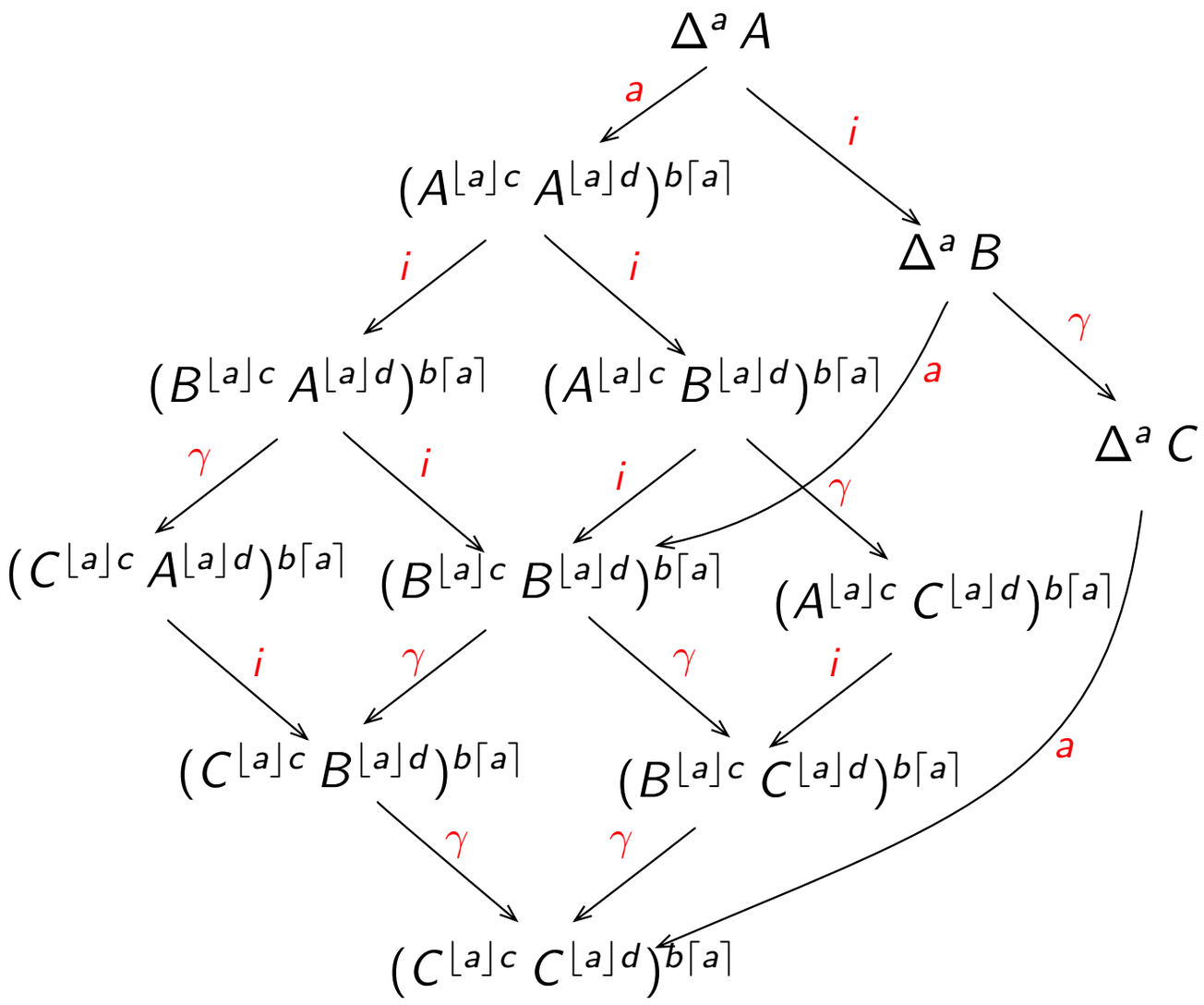
$I = \lambda x.x^v$

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

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

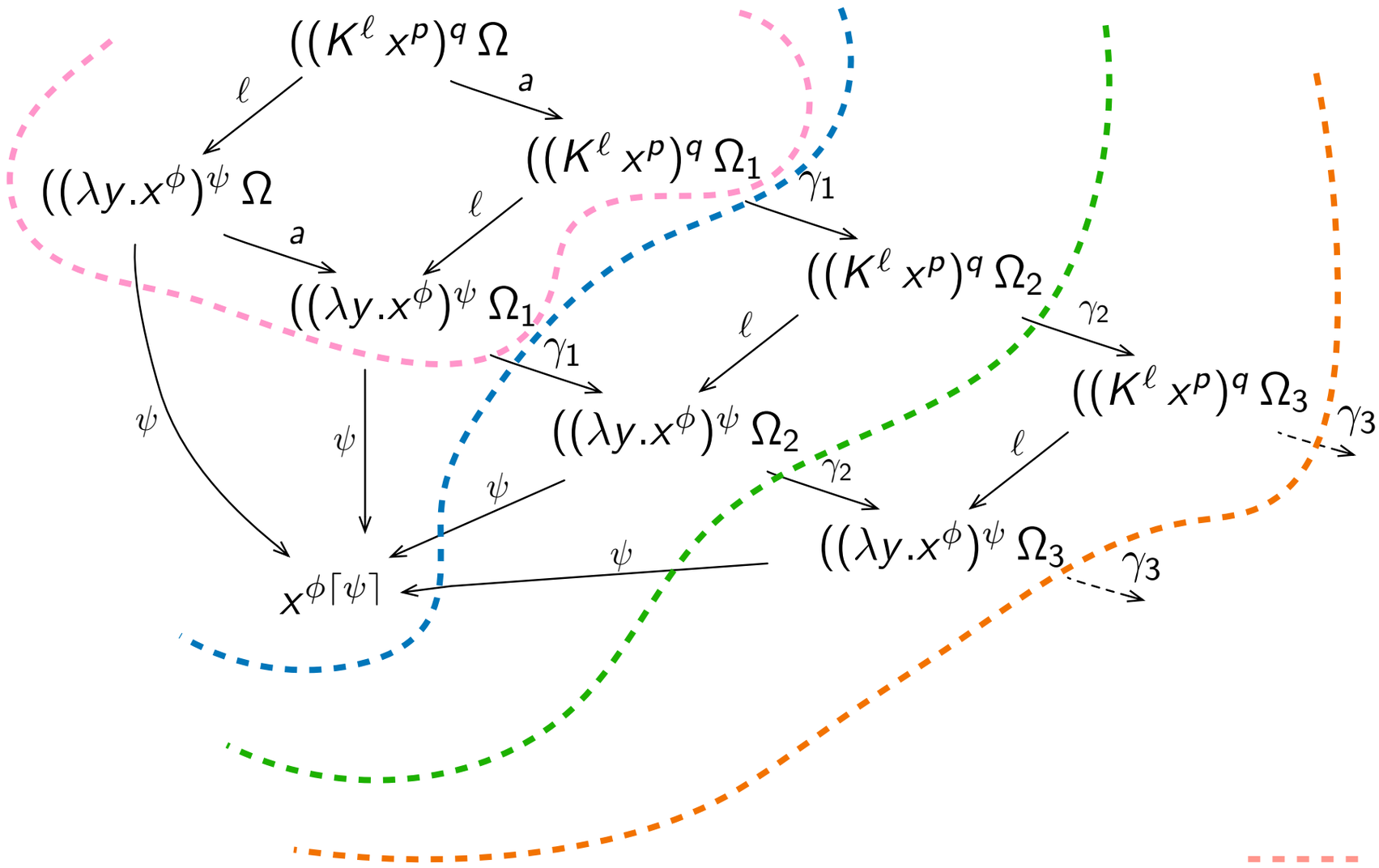
$C = y^{\ell\lfloor\gamma\rfloor v\lceil\gamma\rceil q}$

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



confluence

The labeled λ-calculus



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

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

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

strong normalization

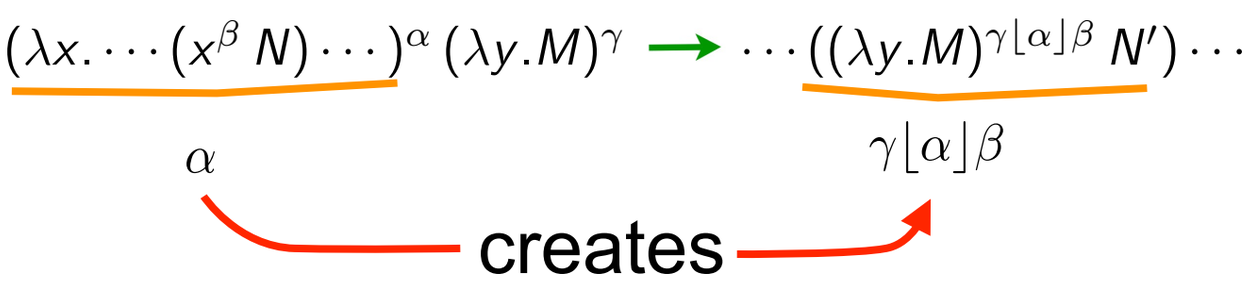
$\{\ell, a\}$

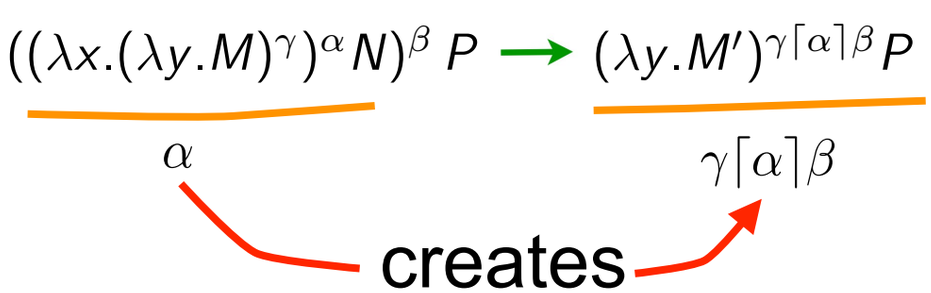
$\{\ell, a, \psi\}$

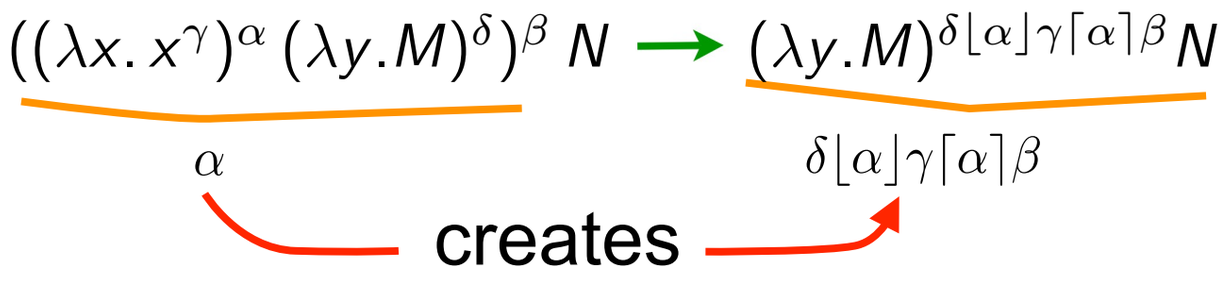
$\{\ell, a, \psi, \gamma_1\}$

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

The labeled λ-calculus

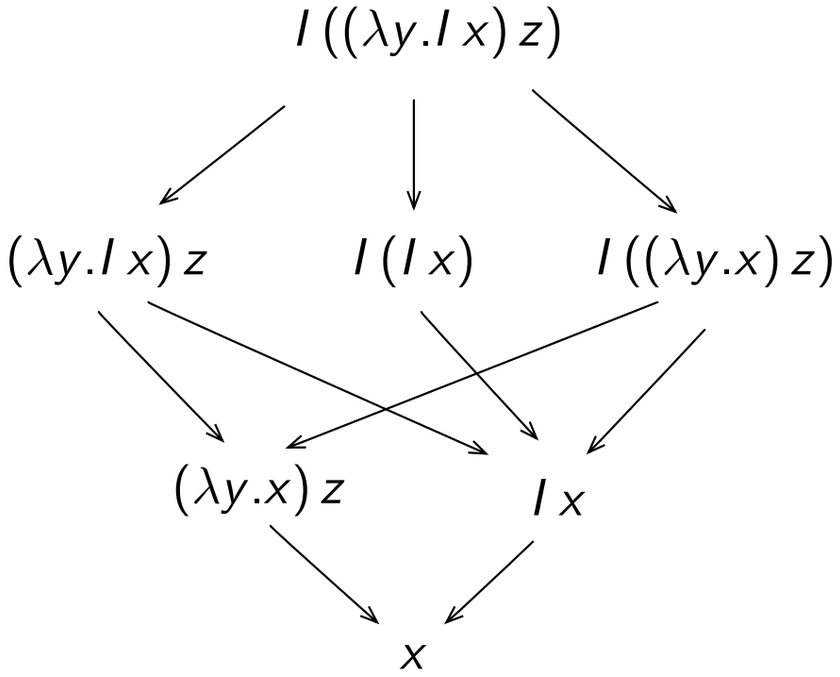






origin

The labeled λ-calculus

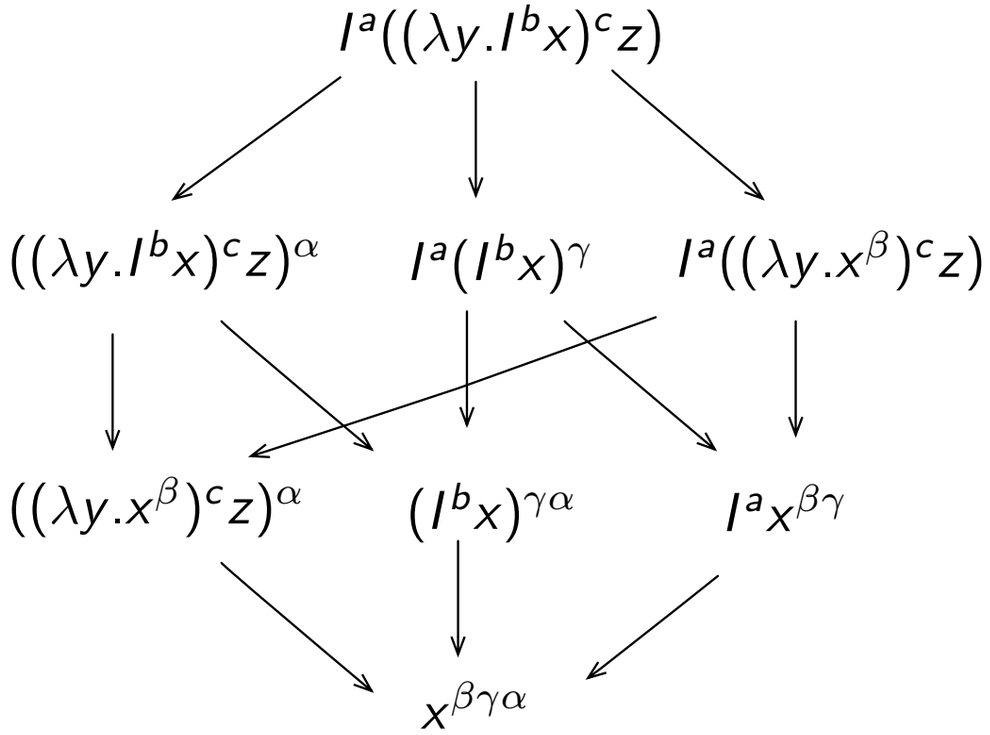


$I = \lambda x.x$

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

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

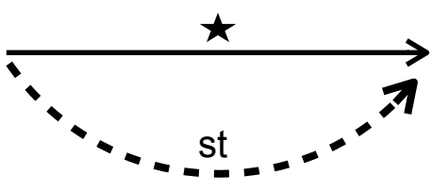
$\gamma = \lceil c \rceil$



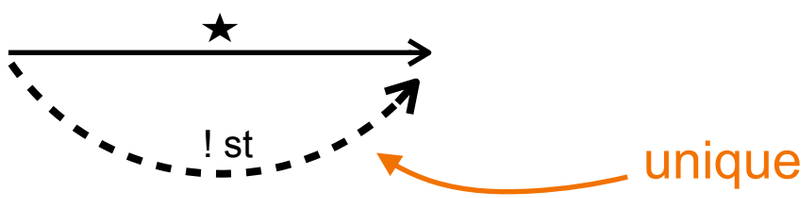
lattice

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]`



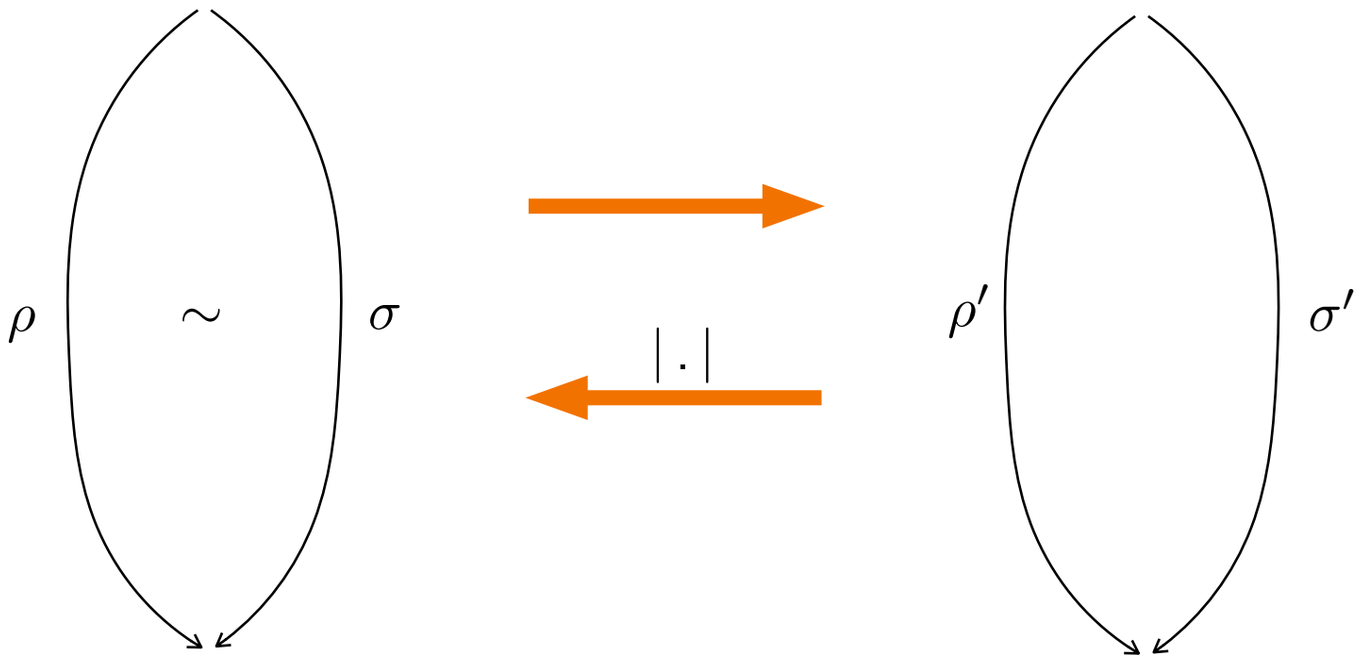
λ-calculus



labeled λ-calculus

Permutation equivalence

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



λ-calculus

$\rho \sim \sigma$

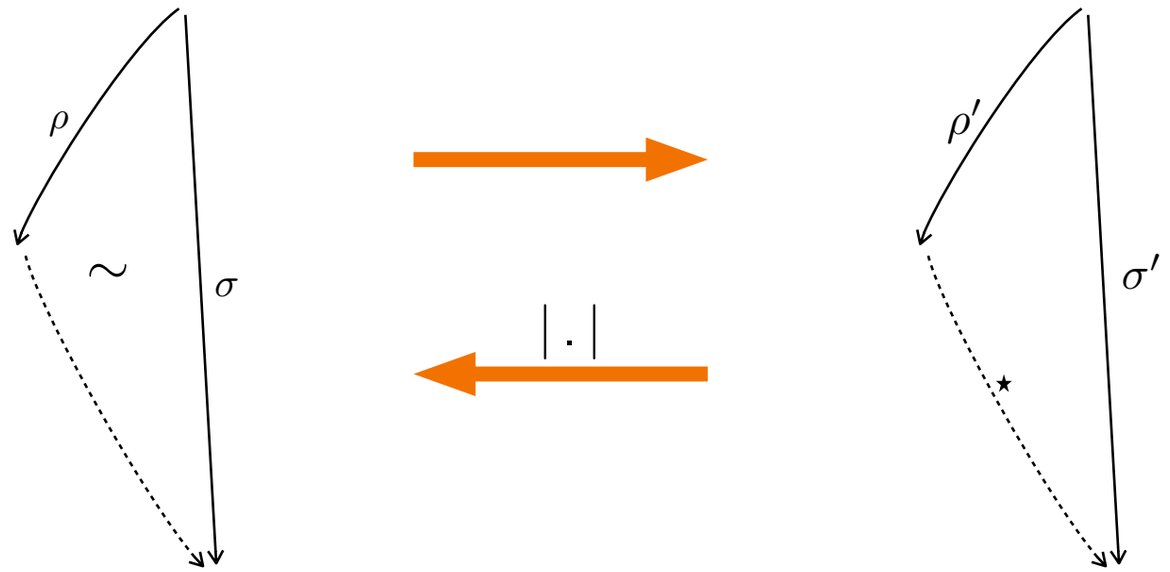
labeled **λ**-calculus

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

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

Prefix modulo permutations

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



λ-calculus

$\rho \leq \sigma$

labeled **λ**-calculus

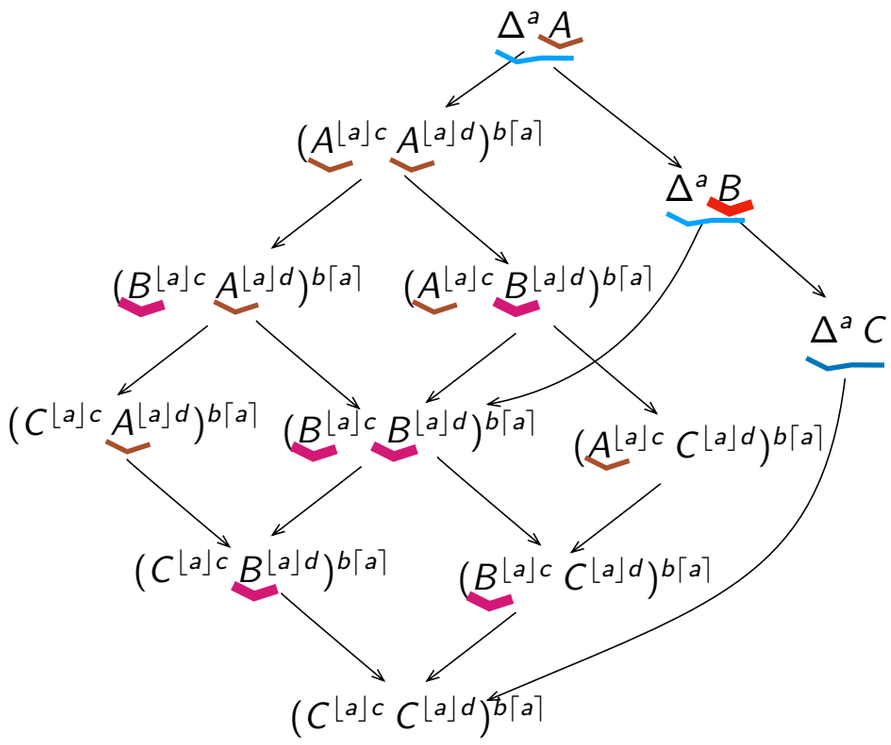
$\mathsf{org}(\rho') = \mathsf{org}(\sigma')$

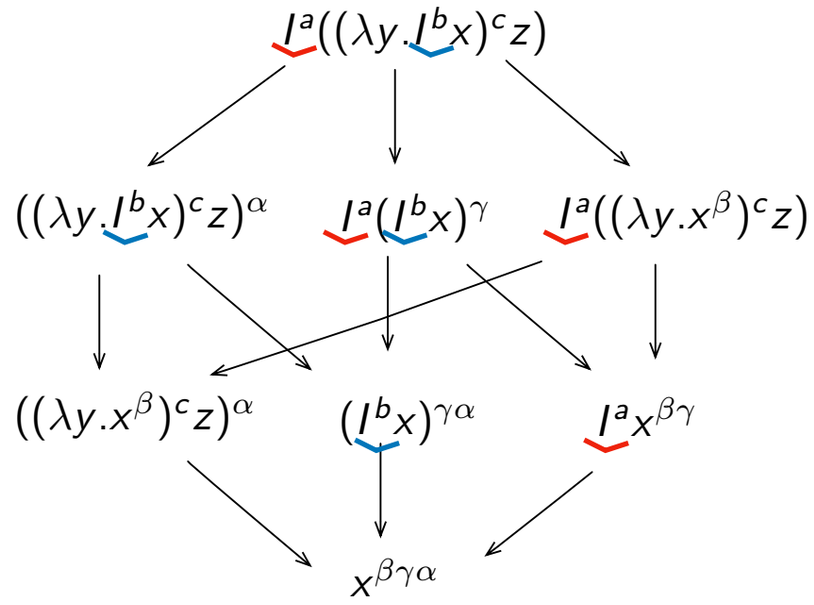
$\mathsf{end}(\rho') \twoheadrightarrow \mathsf{end}(\sigma')$

redex families

Residuals modulo permutations

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





$\Delta = \lambda x.(x^c x^d)^b \qquad A = (F^i I^u)^q$

$F = \lambda f.(f^k y^\ell)^j \qquad B = (I^\gamma y^\ell)^q$

$I = \lambda x.x^v \qquad C = y^{\ell \lfloor \gamma \rfloor v \lceil \gamma \rceil q}$

$\alpha = \lceil a \rceil$

$\beta = \lceil b \rceil$

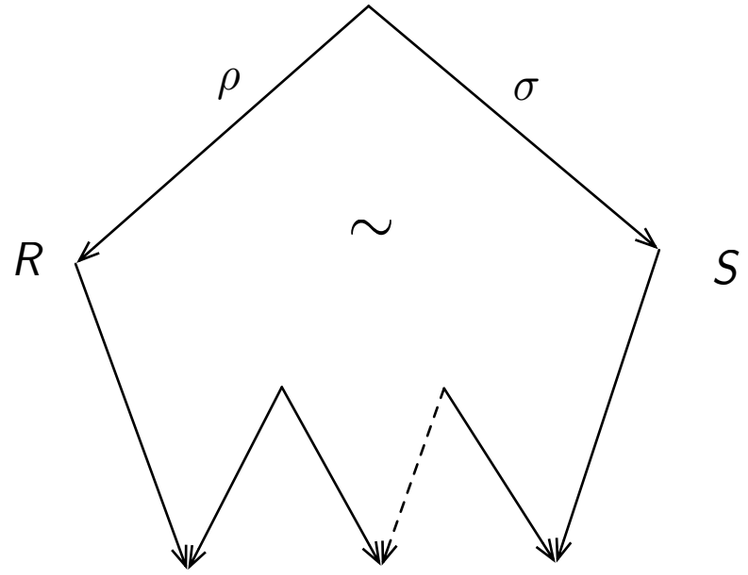
$\gamma = \lceil c \rceil$

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 $\lhd$ on h-redexes is defined as follows

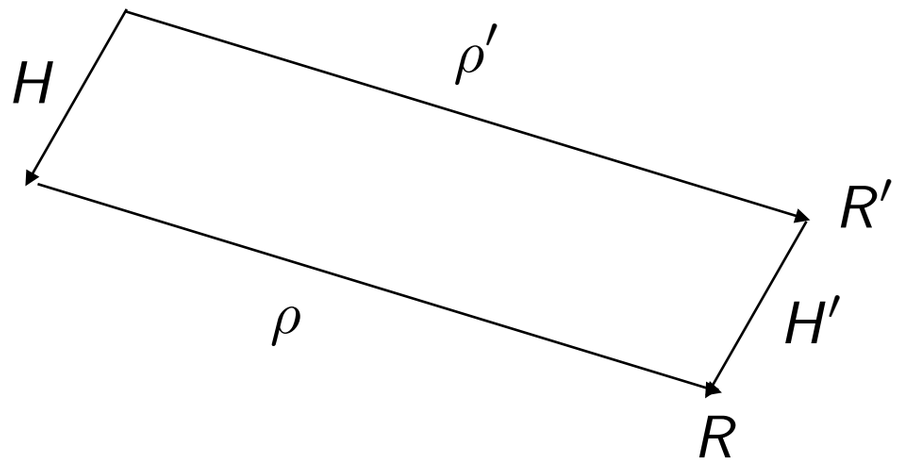
$$(i) \quad \langle o, R\rangle \lhd \langle o, R\rangle$$

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

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

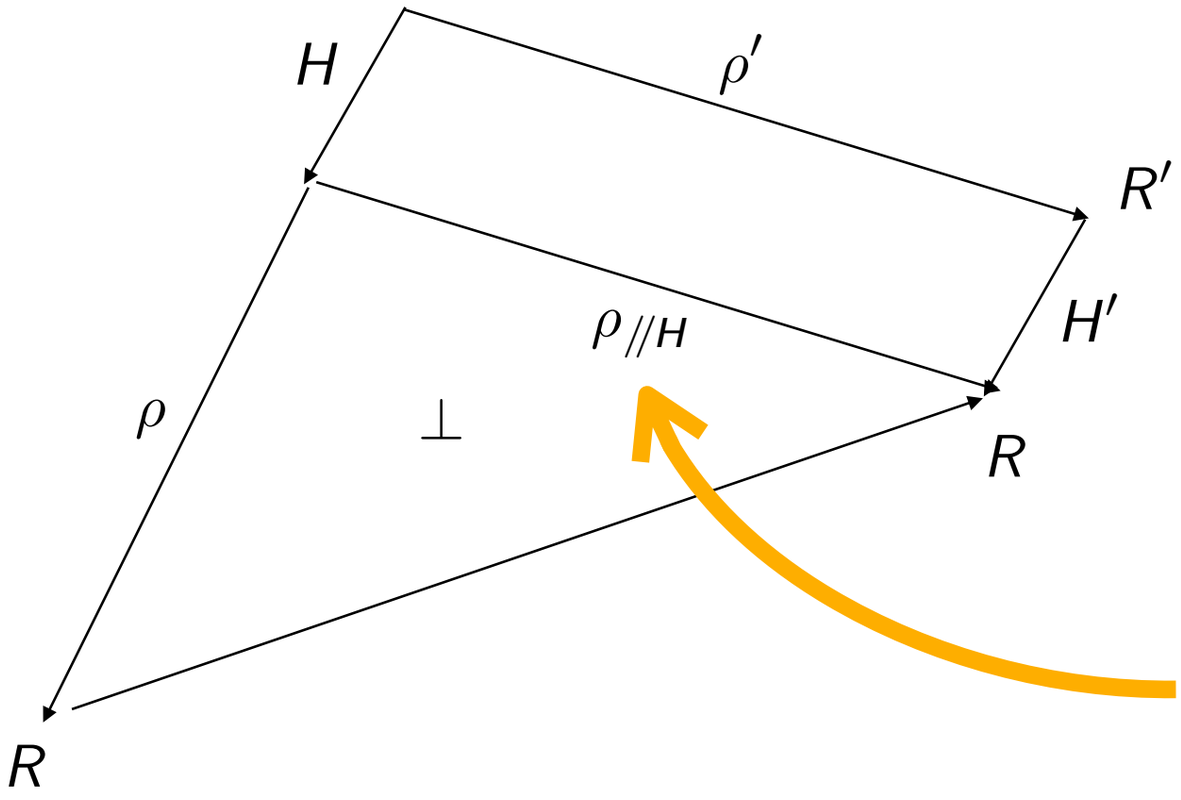
Redex families

- **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

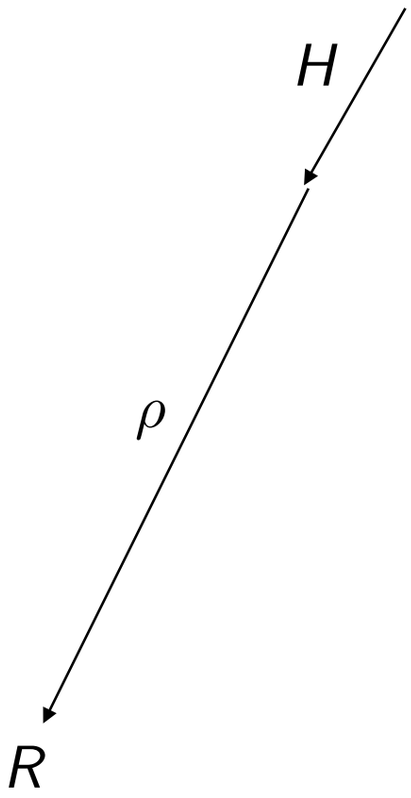
- **Case 2:** ρ is internal to an instance of a copy of the argument of H then ρ' is isomorphic to ρ in the argument of H



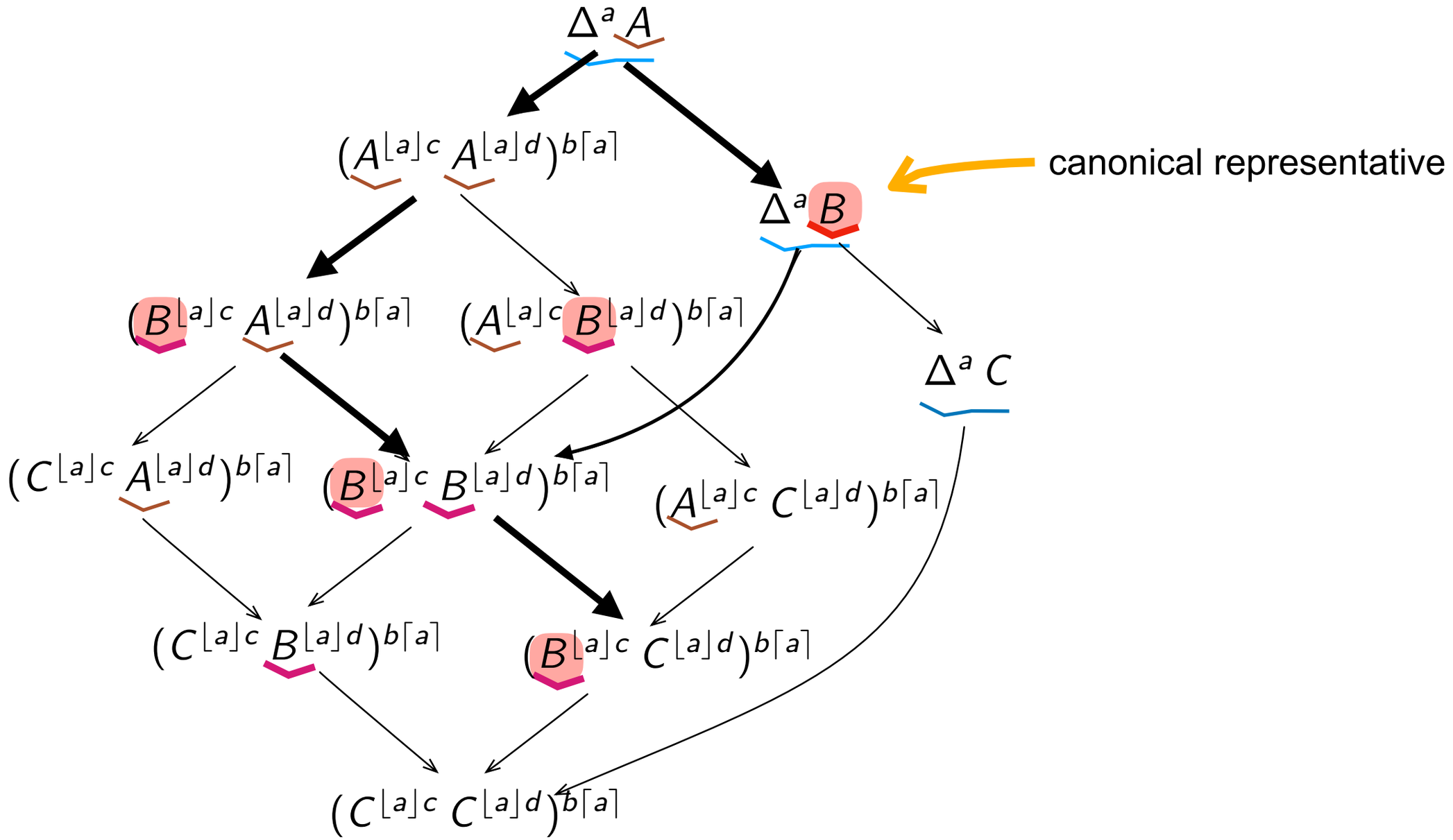
Redex families

- **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

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

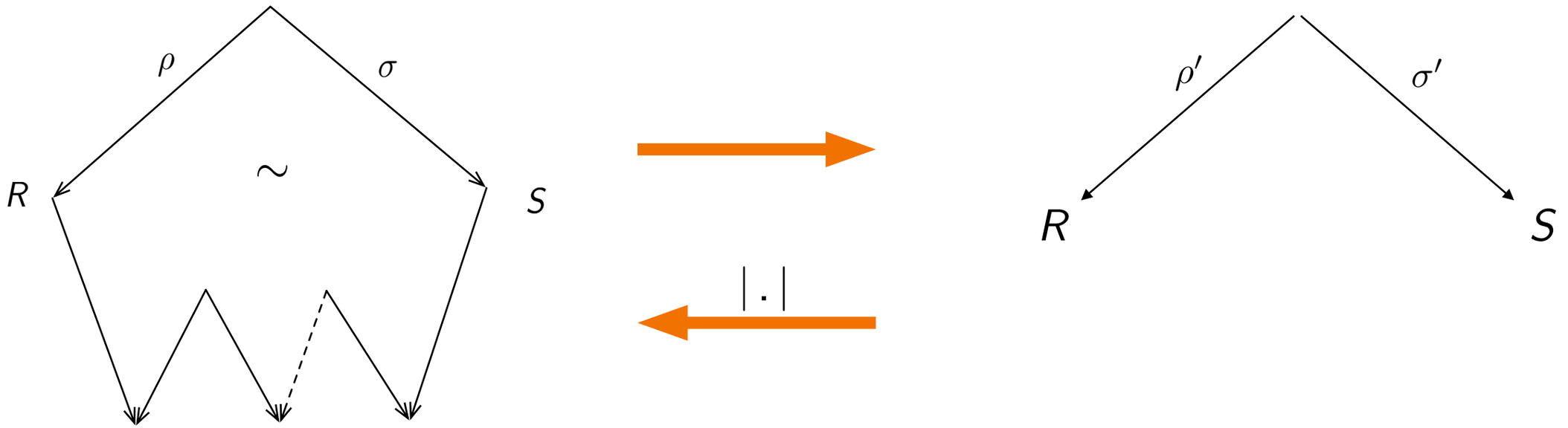
- family complete reductions are duplication complete reductions

- 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 equality of names in the labeled calculus



λ-calculus

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

labeled **λ**-calculus

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

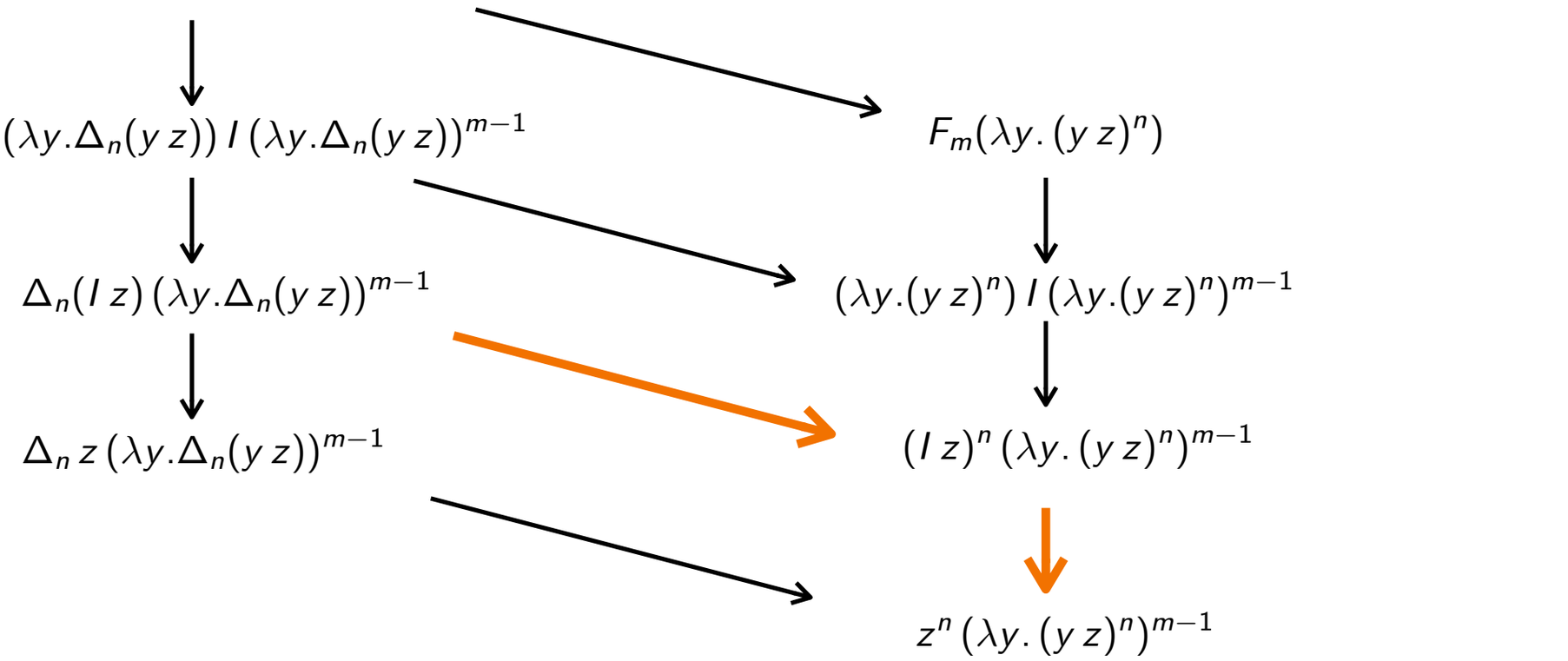
when initial term is **labeled with distinct letters**

- 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 $F_m = \lambda x.\,x\,I\,x\,x\,\ldots\,x$ and $\Delta_n = \lambda x.\,x\,x\,\ldots\,x$



Generalized finite developments

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

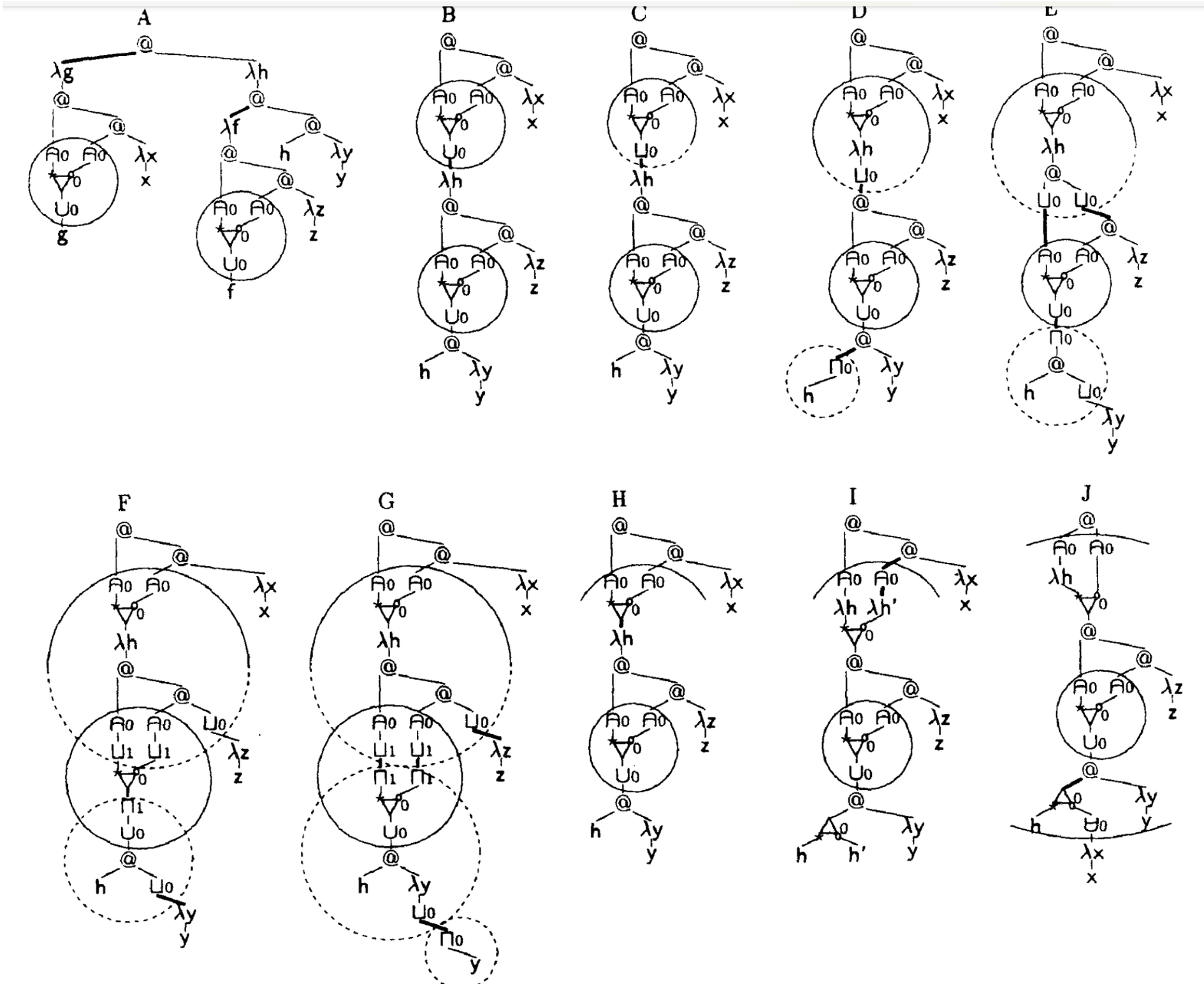
- all reductions contracting only h-redexes in $\mathcal{F}$ have finite length
- these maximal reductions are permutation equivalent

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

strong normalization $\Longleftrightarrow$ finiteness of redex families

Lamping algorithm

- implements optimal reductions [Lamping 90, Gonthier et al 92]



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 ($\lfloor \alpha \to \beta \rfloor = \alpha,\ \lceil \alpha \to \beta \rceil = \beta,\ \alpha\beta = \alpha$)

- simplification of the extraction process

- history-based information flow

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