Finite Developments in the λ-calculus

Part 1

jean-jacques.levy@inria.fr

ISR 2021

Madrid

July 6, 2021

http://jeanjacqueslevy.net/talks/21isr

λ-calculus

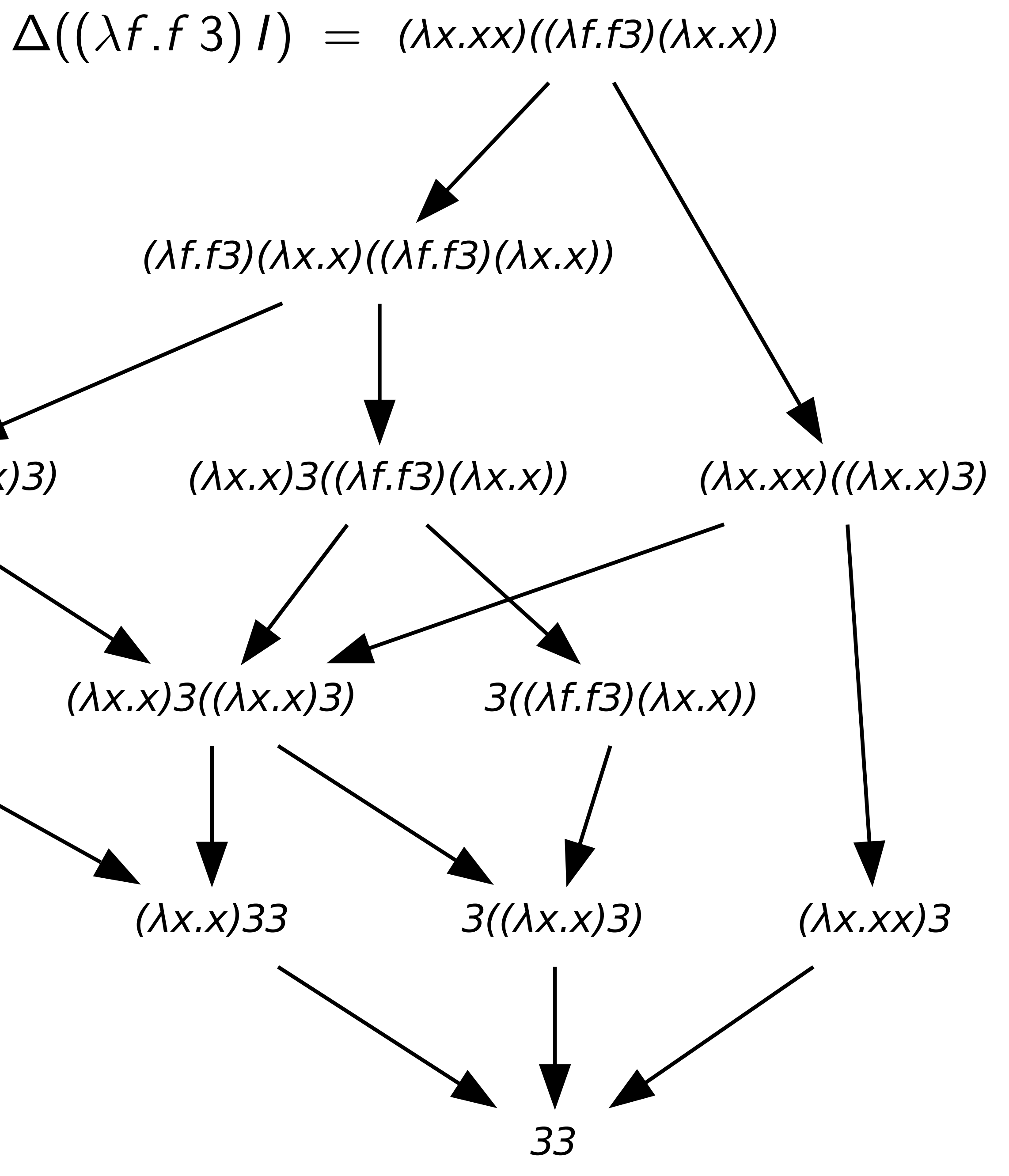
function	λ-term	β-reduction
$I\,x = x$	$I = \lambda x.x$	$I\,a \longrightarrow a$
$K\,x\,y = x$	$K = \lambda x.\lambda y.x$	$K\,a\,b \longrightarrow (\lambda y.a)\,b \longrightarrow a$
$\Delta\,x = x\,x$	$\Delta = \lambda x.x\,x$	$\Delta\,a \longrightarrow a\,a$
$\Omega = \Delta\,\Delta$		$\Omega \longrightarrow \Omega$

Exercise 1

$$\Delta(\lambda x.x\,x\,x) \longrightarrow \cdots$$

$$Y_f = (\lambda x.f\,(x\,x))(\lambda x.f\,(x\,x)) \longrightarrow \cdots$$

λ-calculus



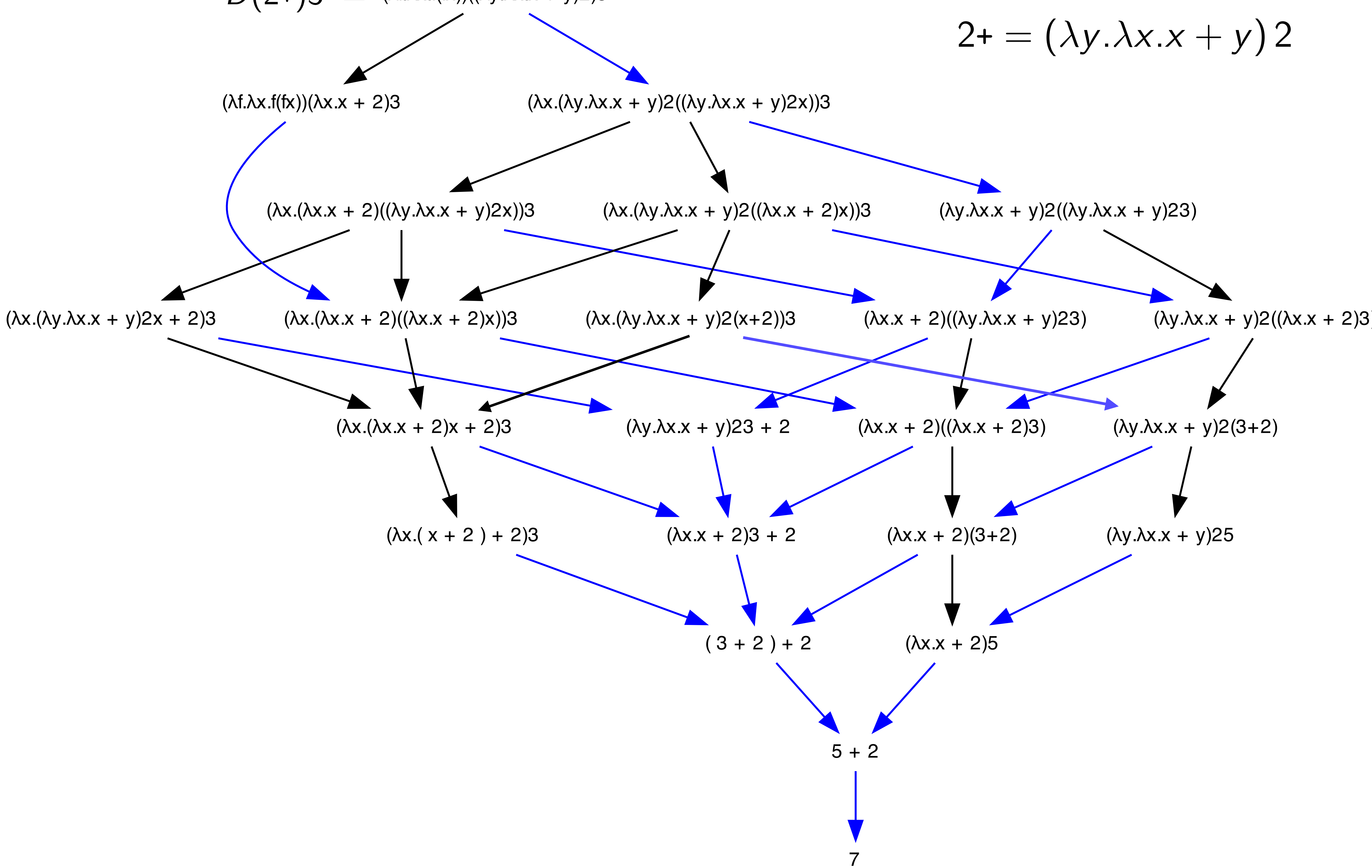
$\Delta((\lambda f.f\ 3)\ I)\ =$ (λx.xx)((λf.f3)(λx.x))
(λf.f3)(λx.x)((λf.f3)(λx.x))
(λf.f3)(λx.x)((λx.x)3)
(λx.x)3((λf.f3)(λx.x))
(λx.xx)((λx.x)3)
(λf.f3)(λx.x)3
(λx.x)3((λx.x)3)
3((λf.f3)(λx.x))
(λx.x)33
3((λx.x)3)
(λx.xx)3
33

λ-calculus

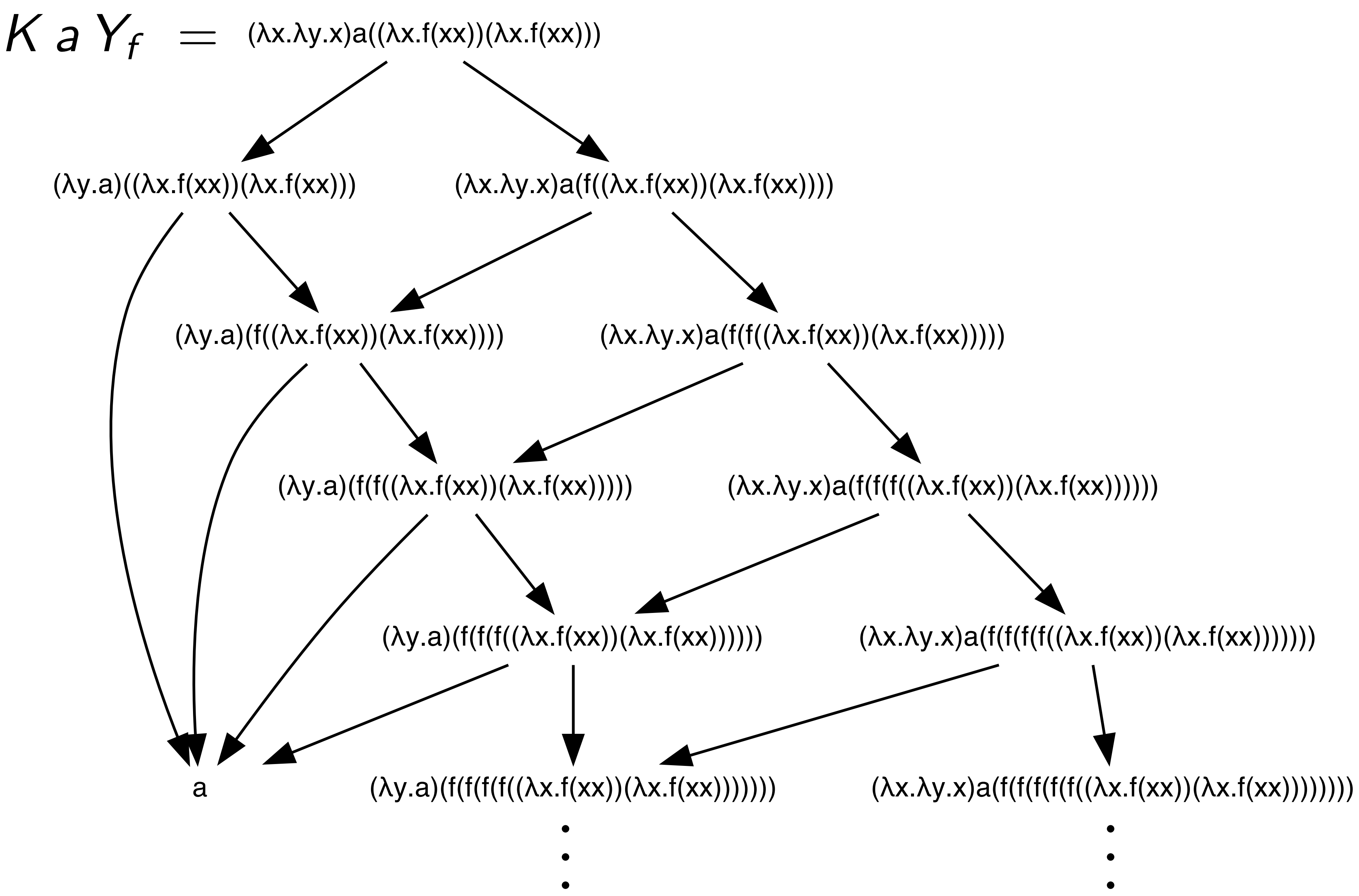
$$D = \lambda f.\lambda x.f(f\,x)$$

$$2{+} = (\lambda y.\lambda x.x + y)\,2$$

$D(2{+})3 \;=$ (λf.λx.f(fx))((λy.λx.x + y)2)3



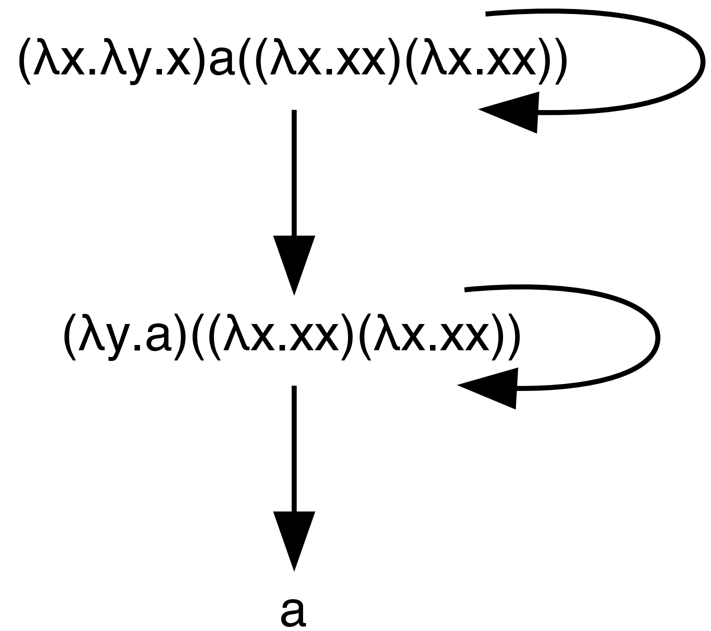
λ-calculus



K a Y_f =
(λx.λy.x)a((λx.f(xx))(λx.f(xx)))
(λy.a)((λx.f(xx))(λx.f(xx)))
(λx.λy.x)a(f((λx.f(xx))(λx.f(xx))))
(λy.a)(f((λx.f(xx))(λx.f(xx))))
(λx.λy.x)a(f(f((λx.f(xx))(λx.f(xx)))))
(λy.a)(f(f((λx.f(xx))(λx.f(xx)))))
(λx.λy.x)a(f(f(f((λx.f(xx))(λx.f(xx))))))
(λy.a)(f(f(f((λx.f(xx))(λx.f(xx))))))
(λx.λy.x)a(f(f(f(f((λx.f(xx))(λx.f(xx)))))))
a
(λy.a)(f(f(f(f((λx.f(xx))(λx.f(xx)))))))
(λx.λy.x)a(f(f(f(f(f((λx.f(xx))(λx.f(xx))))))))

λ-calculus

$K\ a\,\Omega\ =$



Empirical facts

- **deterministic** result when it exists — **Church-Rosser**
- multiple reduction strategies — **CBN - CBV - ..**
- **terminating** strategy ? — **normalisation**
- **efficient** reduction strategy ? — **optimal reduction**
- **worst** reduction strategy ? — **perpetual reduction**
- when all reductions are finite ? — **strong normalisation**
- the reduction graph has a **lattice** structure ? — **NO!**

Redexes

- a redex is any reductible expression: $(\lambda x.M)N$
- the β-conversion rule is:

$$(\lambda x.M)N \longrightarrow M\{x := N\}$$

- a reduction step contracts a given redex $R = (\lambda x.A)B$

and is written: $M \xrightarrow{R} N$

- a reduction step contracts a singleton set of redexes $M \xrightarrow{\{R\}} N$

- a more precise notation would be with occurences of subterms. We avoid it here (but it is sometimes mandatory to avoid ambiguity)
- we replaced occurences by giving names (labels) to redexes.

Bound variables

$$(\lambda x.x\,(\lambda y.x\,y))y = (\lambda x.x\,(\lambda z.x\,z))y$$

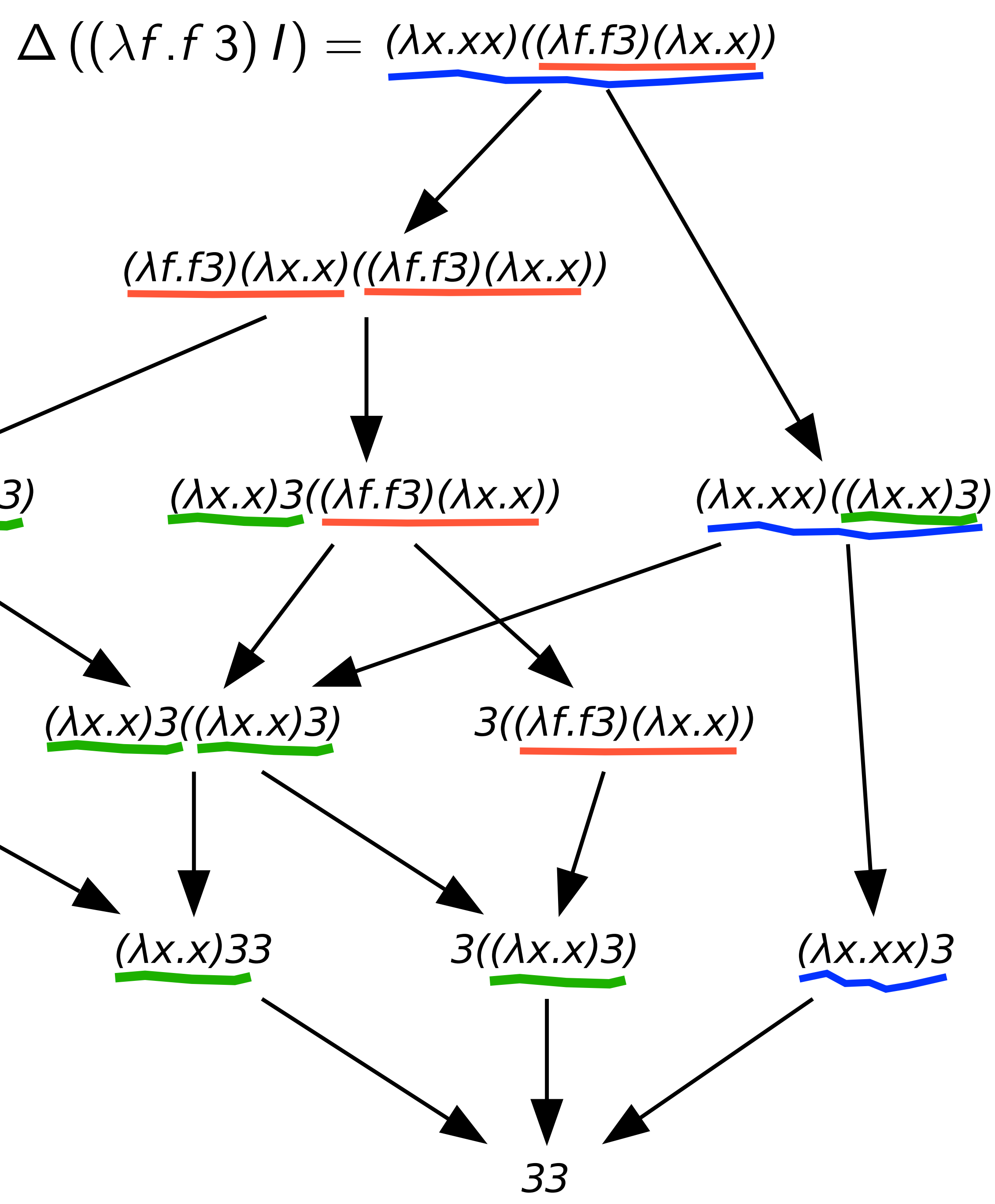
$$y\,(\lambda z.y\,z)$$

- names of bound variables are not important
- we consider λ-terms up-to renaming of bound variables (**α-conversion**)
- free variables of M are formally defined by:

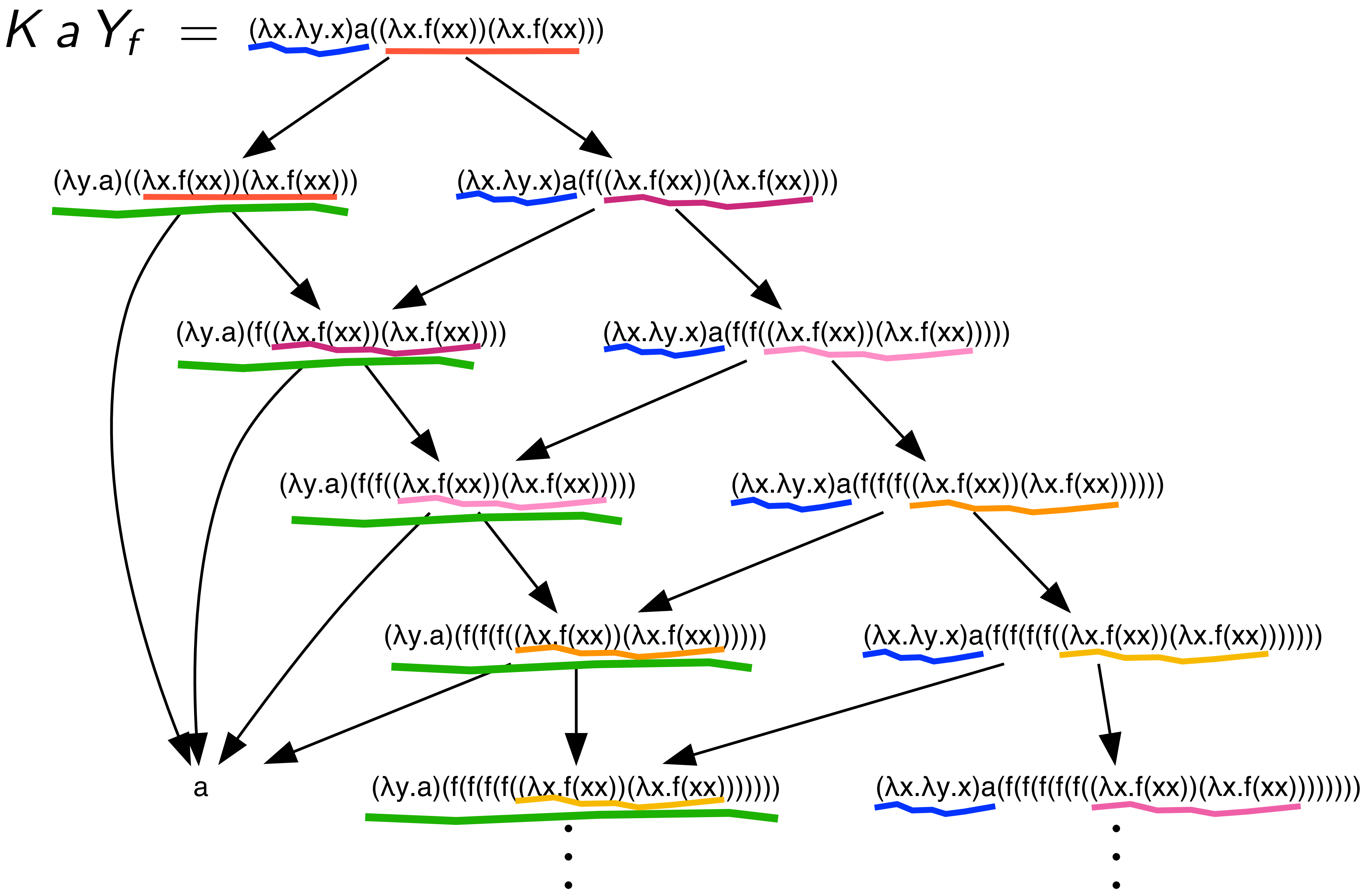
$$\mathsf{FV}(x) = \{x\}$$
$$\mathsf{FV}(\lambda x.M) = \mathsf{FV}(M) - \{x\}$$
$$\mathsf{FV}(MN) = \mathsf{FV}(M) \cup \mathsf{FV}(N)$$

forget **α**-conversion

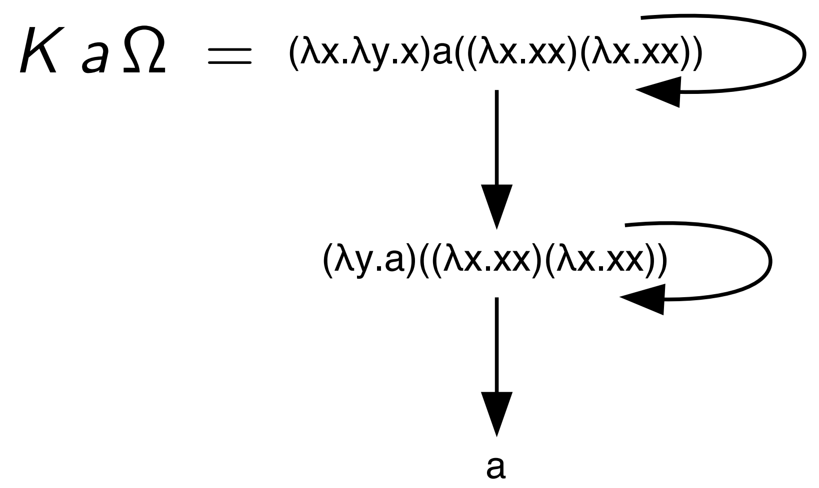
Tracing redexes

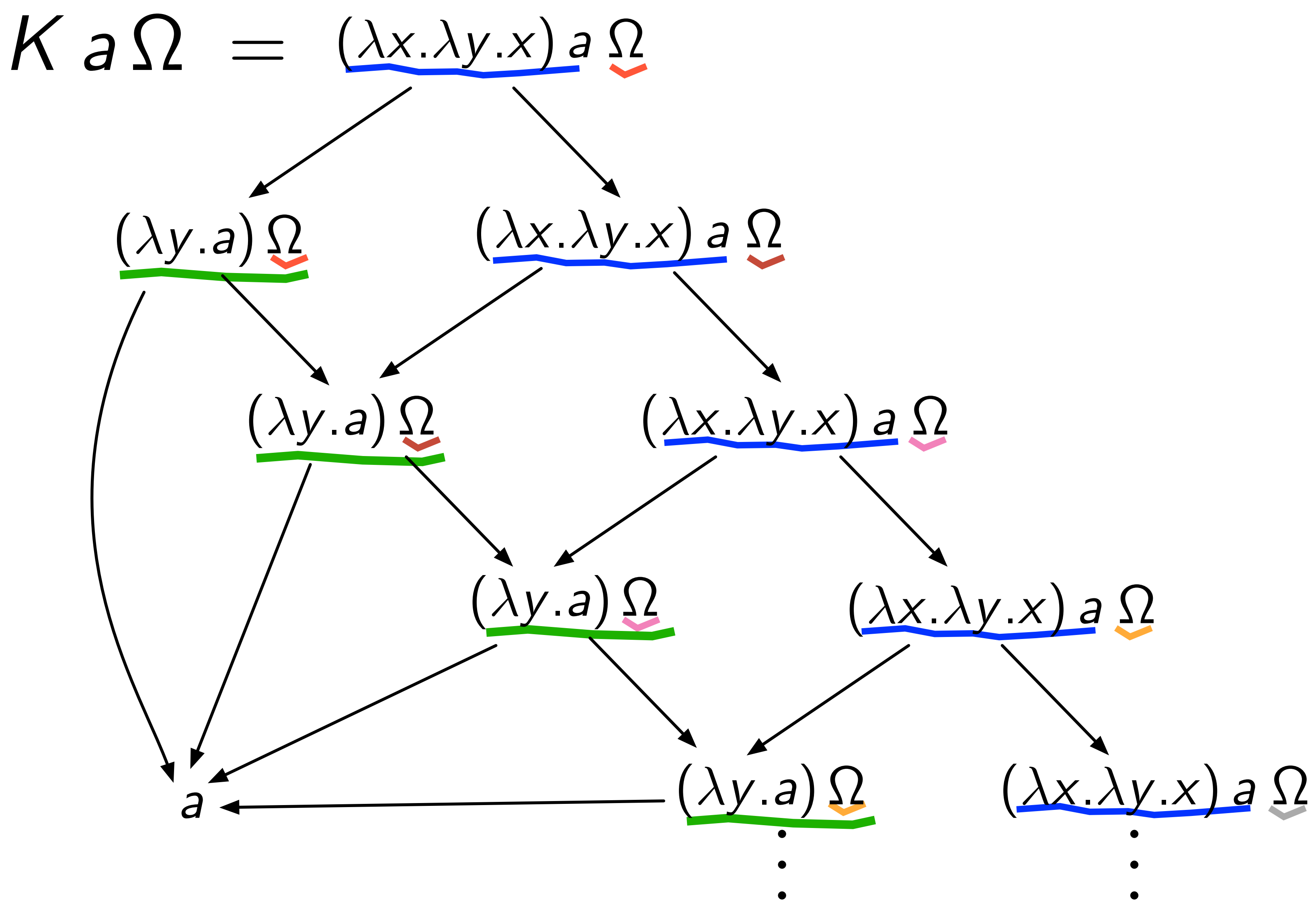


Tracing redexes

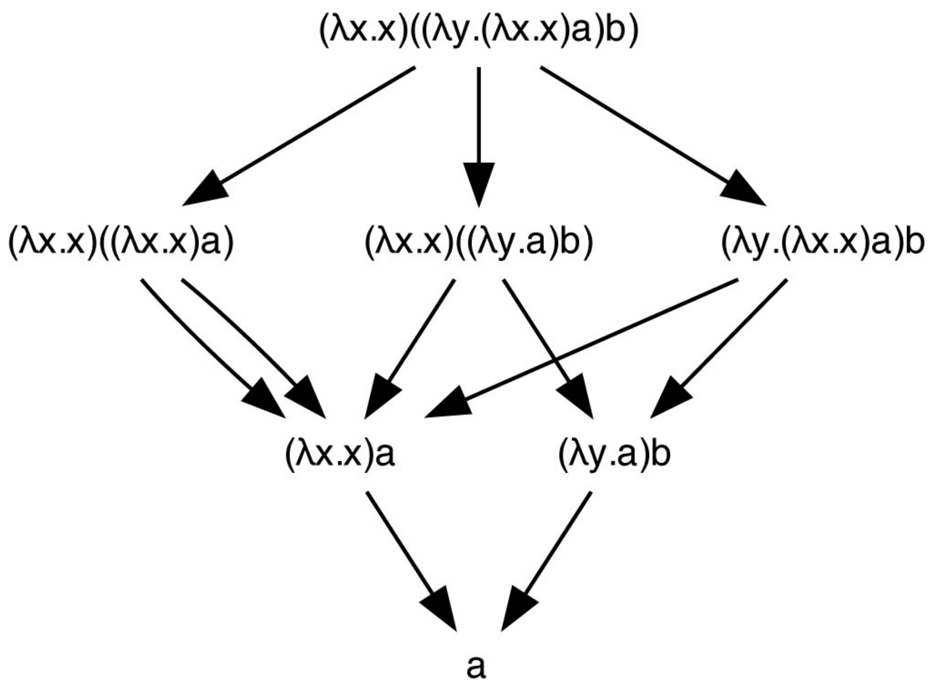


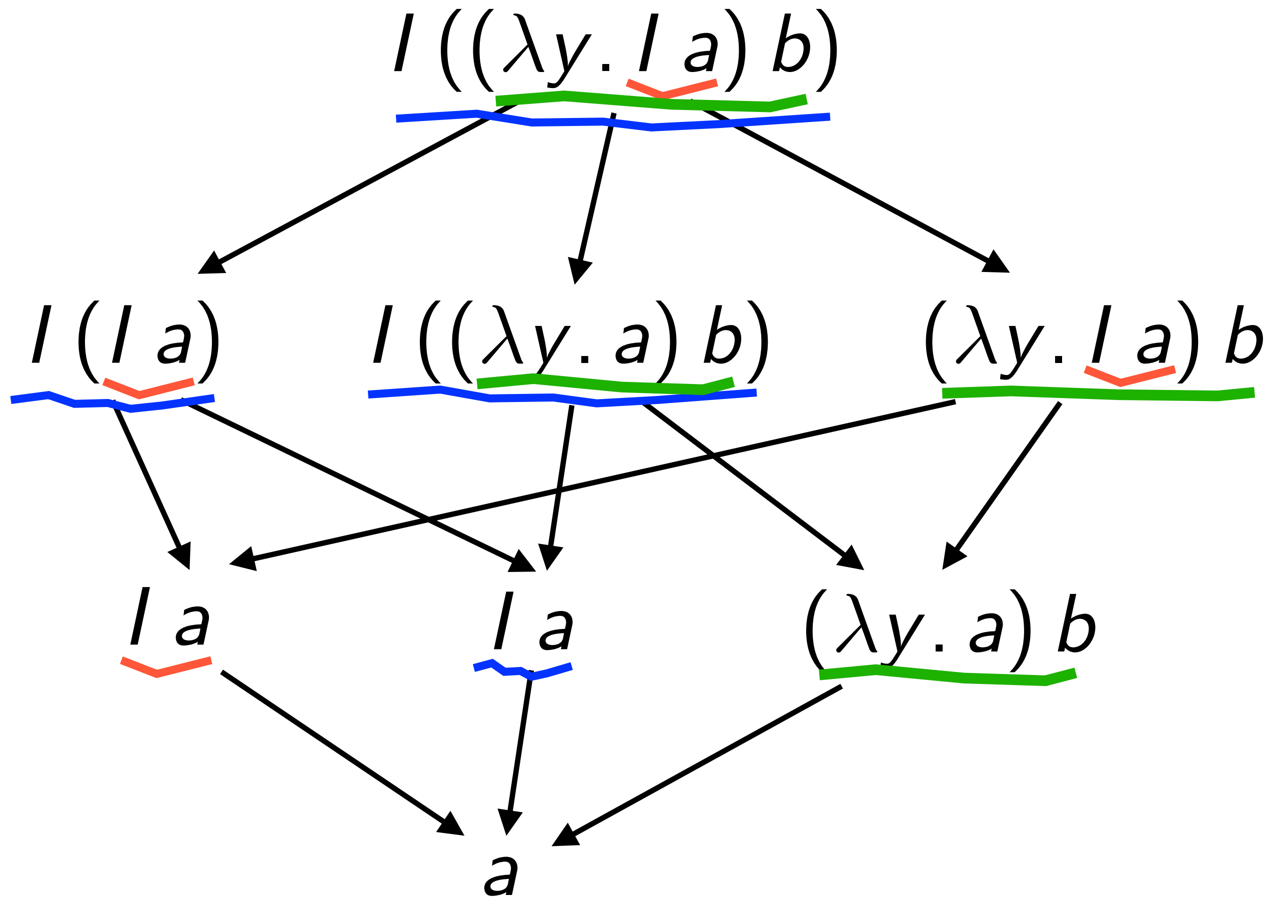
Tracing redexes





Tracing redexes





Empirical facts

- initial redexes in the initial term
- and **newly** created redexes along reductions
- **infinite** reduction iff length of creation is unbounded ?
- **deterministic** result when finite families of redexes are contracted ?

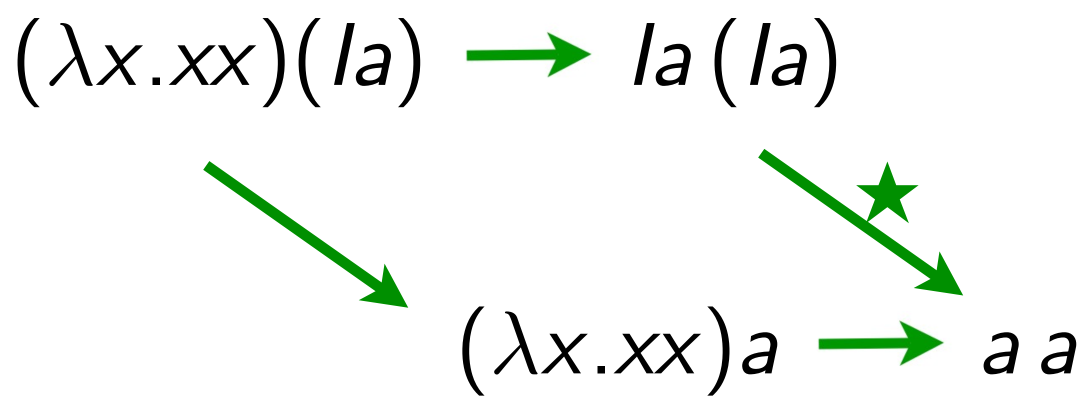
Finite Developments Theorem

Curry'50 JJL'78

Parallel reduction steps

Parallel reductions (1/3)

- permutation of reductions has to cope with copies of redexes



- in fact, a parallel reduction $Ia(Ia) \not\twoheadrightarrow aa$

- in λ-calculus, need to define parallel reductions for nested sets

Fact In the λ-calculus, disjoint redexes may become nested $(\lambda x.Ix)(\Delta y) \longrightarrow I(\Delta y)$

Parallel reductions (2/3)

- the axiomatic way (à la Martin-Löf)

[Var Axiom] $x \mathrel{\rightarrow\!\!\!\!\!/\!\!/} x$ [Const Axiom] $c \mathrel{\rightarrow\!\!\!\!\!/\!\!/} c$

[App Rule] $$\frac{M \mathrel{\rightarrow\!\!\!\!\!/\!\!/} M' \quad N \mathrel{\rightarrow\!\!\!\!\!/\!\!/} N'}{MN \mathrel{\rightarrow\!\!\!\!\!/\!\!/} M'N'}$$

[Abs Rule] $$\frac{M \mathrel{\rightarrow\!\!\!\!\!/\!\!/} M'}{\lambda x.M \mathrel{\rightarrow\!\!\!\!\!/\!\!/} \lambda x.M'}$$

[//Beta Rule] $$\frac{M \mathrel{\rightarrow\!\!\!\!\!/\!\!/} M' \quad N \mathrel{\rightarrow\!\!\!\!\!/\!\!/} N'}{(\lambda x.M)N \mathrel{\rightarrow\!\!\!\!\!/\!\!/} M'\{x := N'\}}$$

inside-out (possibly void) parallel reductions

- examples:

$$(\lambda x.Ix)(Iy) \mathrel{\rightarrow\!\!\!\!\!/\!\!/} (\lambda x.x)y$$

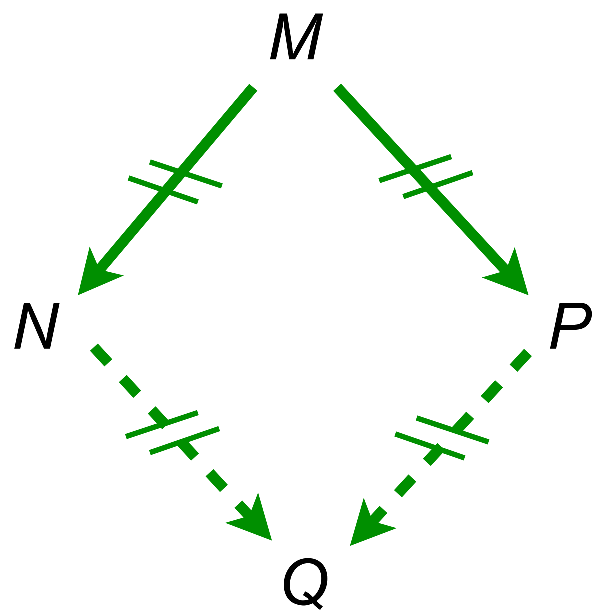
$$(\lambda x.(\lambda y.yy)x)(Ia) \mathrel{\rightarrow\!\!\!\!\!/\!\!/} Ia(Ia)$$

$$(\lambda x.(\lambda y.yy)x)(Ia) \mathrel{\rightarrow\!\!\!\!\!/\!\!/} (\lambda y.yy)a$$

Parallel reductions (3/3)

- **Parallel moves lemma** [Curry 50]

  If $M \mathrel{/\!\!/\!\!\!\!\!\!\rightarrow} N$ and $M \mathrel{/\!\!/\!\!\!\!\!\!\rightarrow} P$, then $N \mathrel{/\!\!/\!\!\!\!\!\!\rightarrow} Q$ and $P \mathrel{/\!\!/\!\!\!\!\!\!\rightarrow} Q$ for some Q.



lemma 1-1-1-1
(strong confluency)

Enough to prove Church Rosser theorem since $\rightarrow \subset \mathrel{/\!\!/\!\!\!\!\!\!\rightarrow} \subset \xrightarrow{\star}$

[Tait--Martin Löf 60?]

Reduction of a set of redexes (1/4)

- Goal: parallel reduction of a given set of redexes

$$M, N ::= x \mid \lambda x.M \mid MN \mid (\lambda x.M)^a N$$

$$a, b, c, \ldots ::= \text{redex labels}$$

(labeled β-rule)

$$(\lambda x.M)^a N \longrightarrow M\{x := N\}$$

- Substitution as before with add-on:

$$((\lambda y.P)^a Q)\{x := N\} = (\lambda y.P\{x := N\})^a Q\{x := N\}$$

Reduction of a set of redexes (2/4)

- let $\mathcal{F}$ be a set of redex labels

[Var Axiom] $x \xrightarrow{\mathcal{F}} x$

[Const Axiom] $c \xrightarrow{\mathcal{F}} c$

[App Rule] $$\frac{M \xrightarrow{\mathcal{F}} M' \quad N \xrightarrow{\mathcal{F}} N'}{MN \xrightarrow{\mathcal{F}} M'N'}$$

[Abs Rule] $$\frac{M \xrightarrow{\mathcal{F}} M'}{\lambda x.M \xrightarrow{\mathcal{F}} \lambda x.M'}$$

[//Beta Rule] $$\frac{M \xrightarrow{\mathcal{F}} M' \quad N \xrightarrow{\mathcal{F}} N' \quad a \in \mathcal{F}}{(\lambda x.M)^a N \xrightarrow{\mathcal{F}} M'\{x := N'\}}$$

[Redex'] $$\frac{M \xrightarrow{\mathcal{F}} M' \quad N \xrightarrow{\mathcal{F}} N' \quad a \notin \mathcal{F}}{(\lambda x.M)^a N \xrightarrow{\mathcal{F}} (\lambda x.M')^a N'}$$

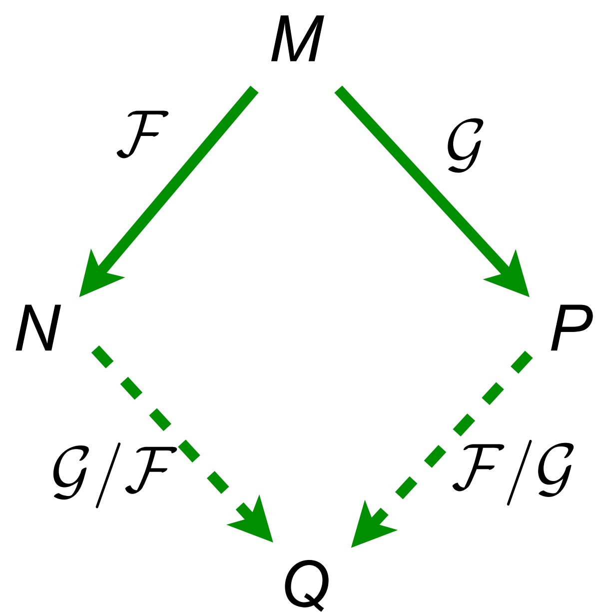
inside-out parallel reductions of redexes labeled in $\mathcal{F}$

- let $\mathcal{F}$, $\mathcal{G}$ be set of redexes in M and let $M \xrightarrow{\mathcal{F}} N$, then the set $\mathcal{G}/\mathcal{F}$ of residuals of $\mathcal{G}$ by $\mathcal{F}$ is the set of $\mathcal{G}$ redexes in N.

Reduction of a set of redexes (3/4)

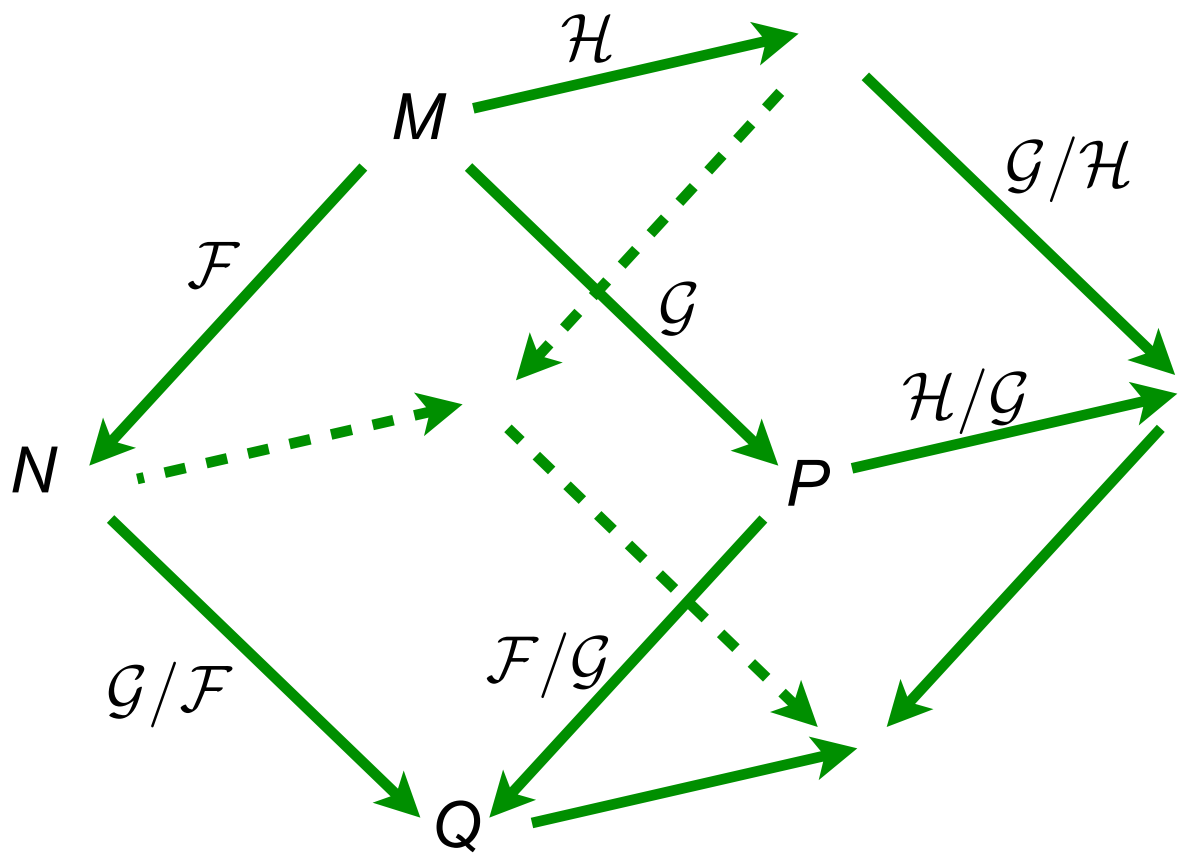
- **Parallel moves lemma+** [Curry 50]

  If $M \xrightarrow{\mathcal{F}} N$ and $M \xrightarrow{\mathcal{G}} P$, then $N \xrightarrow{\mathcal{G}/\mathcal{F}} Q$ and $P \xrightarrow{\mathcal{F}/\mathcal{G}} Q$ for some Q.



Reduction of a set of redexes (4/4)

- **Parallel moves lemma++** `[Curry 50] The Cube Lemma`



$$(\mathcal{H}/\mathcal{F})/(\mathcal{G}/\mathcal{F}) = (\mathcal{H}/\mathcal{G})/(\mathcal{F}/\mathcal{G})$$

Residuals of redexes

Redexes

- a redex is any reductible expression: $(\lambda x.M)N$

- a reduction step contracts a given redex $R = (\lambda x.A)B$

and is written: $M \xrightarrow{R} N$

- a reduction step contracts a singleton set of redexes $M \xrightarrow{\{R\}} N$

- a more precise notation would be with occurences of subterms. We avoid it here (but it is sometimes mandatory to avoid ambiguity)

- we replaced occurences by giving names (labels) to redexes.

Residuals of redexes (1/4)

- **residuals** of redexes were defined by considering **labels**
- residuals are redexes with **same labels**
- a closer look w.r.t. their relative positions give following cases:

let $R = (\lambda x.A)B$, let $M \xrightarrow{R} N$ and $S = (\lambda y.C)D$ be an other redex in M. Then:

Residuals of redexes (2/4)

Case 1:

$$M = \cdots\cdots R \cdots\cdots \underbrace{S}_{} \cdots\cdots \xrightarrow{R} \cdots\cdots R' \cdots\cdots \underbrace{S}_{} \cdots\cdots = N$$

or

$$M = \cdots\cdots \underbrace{S}_{} \cdots\cdots R \cdots\cdots \xrightarrow{R} \cdots\cdots \underbrace{S}_{} \cdots\cdots R' \cdots\cdots = N$$

Case 2:

$$M = \cdots\cdots \underbrace{R}_{} \cdots\cdots \xrightarrow{R} \cdots\cdots R' \cdots\cdots = N \quad (R \text{ and } S \text{ coincide})$$

Case 3:

$$M = \cdots \underbrace{(\lambda y. \cdots R \cdots)D} \cdots \xrightarrow{R} \cdots \underbrace{(\lambda y. \cdots R' \cdots)D} \cdots = N$$

Case 4:

$$M = \cdots \underbrace{(\lambda y.C)(\cdots R \cdots)} \cdots \xrightarrow{R} \cdots \underbrace{(\lambda y.C)(\cdots R' \cdots)} \cdots = N$$

Residuals of redexes (3/4)

Case 3:

$$M = \cdots(\lambda x.\cdots S \cdots)B \cdots \xrightarrow{R} \cdots\cdots S\{x := B\} \cdots\cdots = N$$

Case 4:

$$M = \cdots(\lambda x.\cdots x \cdots x \cdots)(\cdots S \cdots)\cdots$$

$$\xrightarrow{R} \cdots\cdots(\cdots S \cdots)\cdots(\cdots S \cdots)\cdots\cdots = N$$

Residuals of redexes (4/4)

Examples: $\Delta = \lambda x.xx,\ I = \lambda x.x$

$$\Delta(\underbrace{I\,x}) \longrightarrow \underbrace{I\,x}(\underbrace{I\,x})$$

$$\underbrace{I\,x}\,(\Delta(I\,x)) \longrightarrow \underbrace{I\,x}\,(I\,x(I\,x))$$

$$\underbrace{I\,(\Delta(I\,x))} \longrightarrow \underbrace{I\,(I\,x(I\,x))}$$

$$\underbrace{\Delta(I\,x)} \longrightarrow I\,x(I\,x)$$

$$I\,x\,(\Delta(\underbrace{I\,x})) \longrightarrow I\,x\,(\underbrace{I\,x}(\underbrace{I\,x}))$$

$$\underbrace{\Delta\Delta} \longrightarrow \Delta\Delta$$

Residuals of reductions

Parallel reductions

- Consider reductions where each step is parallel

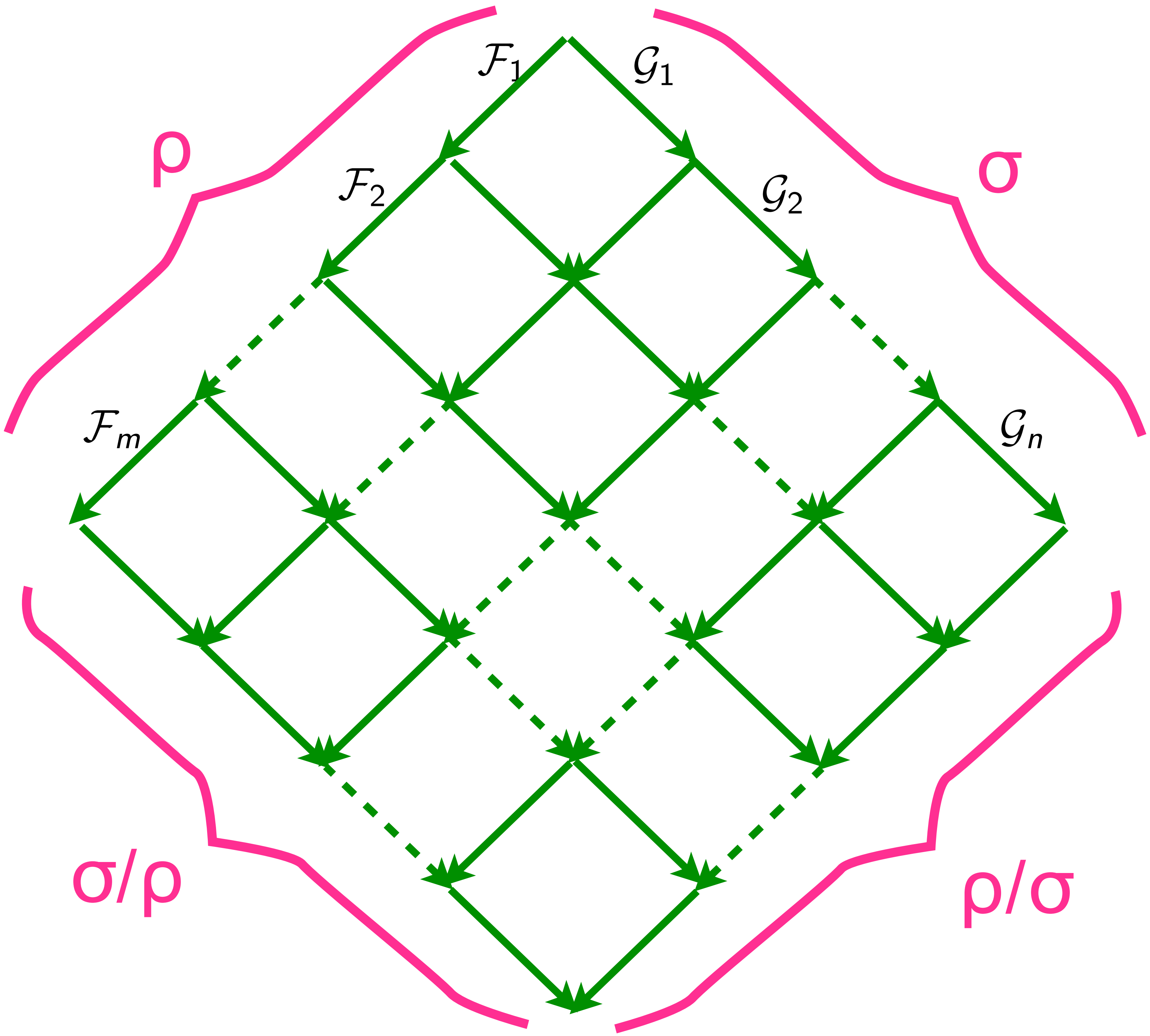
$$\rho : M = M_0 \xrightarrow{\mathcal{F}_1} M_1 \xrightarrow{\mathcal{F}_2} M_2 \cdots \xrightarrow{\mathcal{F}_n} M_n = N$$

- We also write

$$\rho = 0 \text{ when } n = 0$$

$$\rho = \mathcal{F}_1 \mathcal{F}_2 \cdots \mathcal{F}_n \text{ when } M \text{ clear from context}$$

Residuals of reductions (1/4)



Residuals of reductions (2/4)

- **Definition** [JJL 76]

$$\rho/0 = \rho$$

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

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

$\mathcal{F}/\mathcal{G}$ already defined

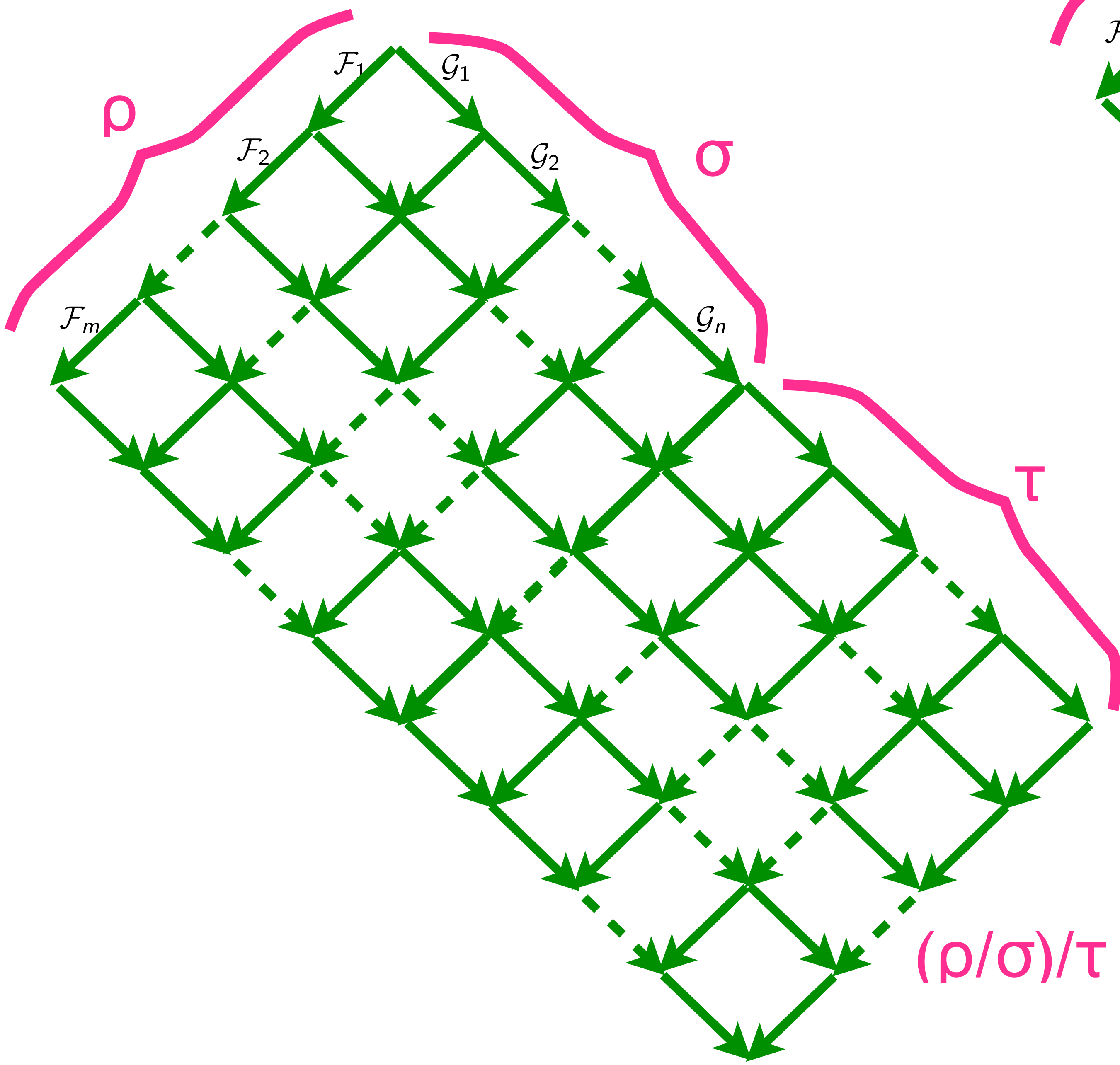
- **Notation**

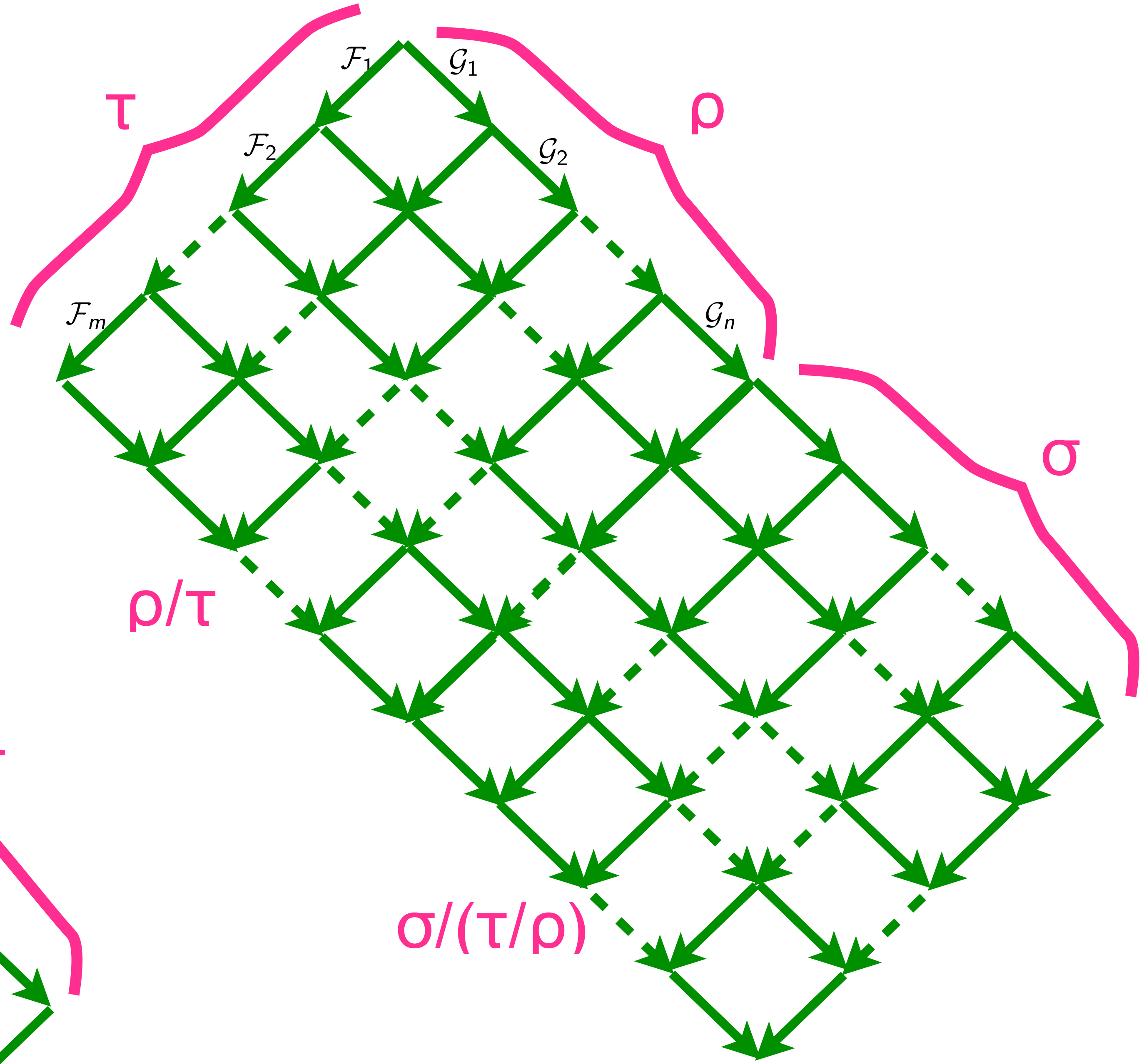
$$\rho \sqcup \sigma = \rho\,(\sigma/\rho)$$

- **Proposition** [Parallel Moves +]:

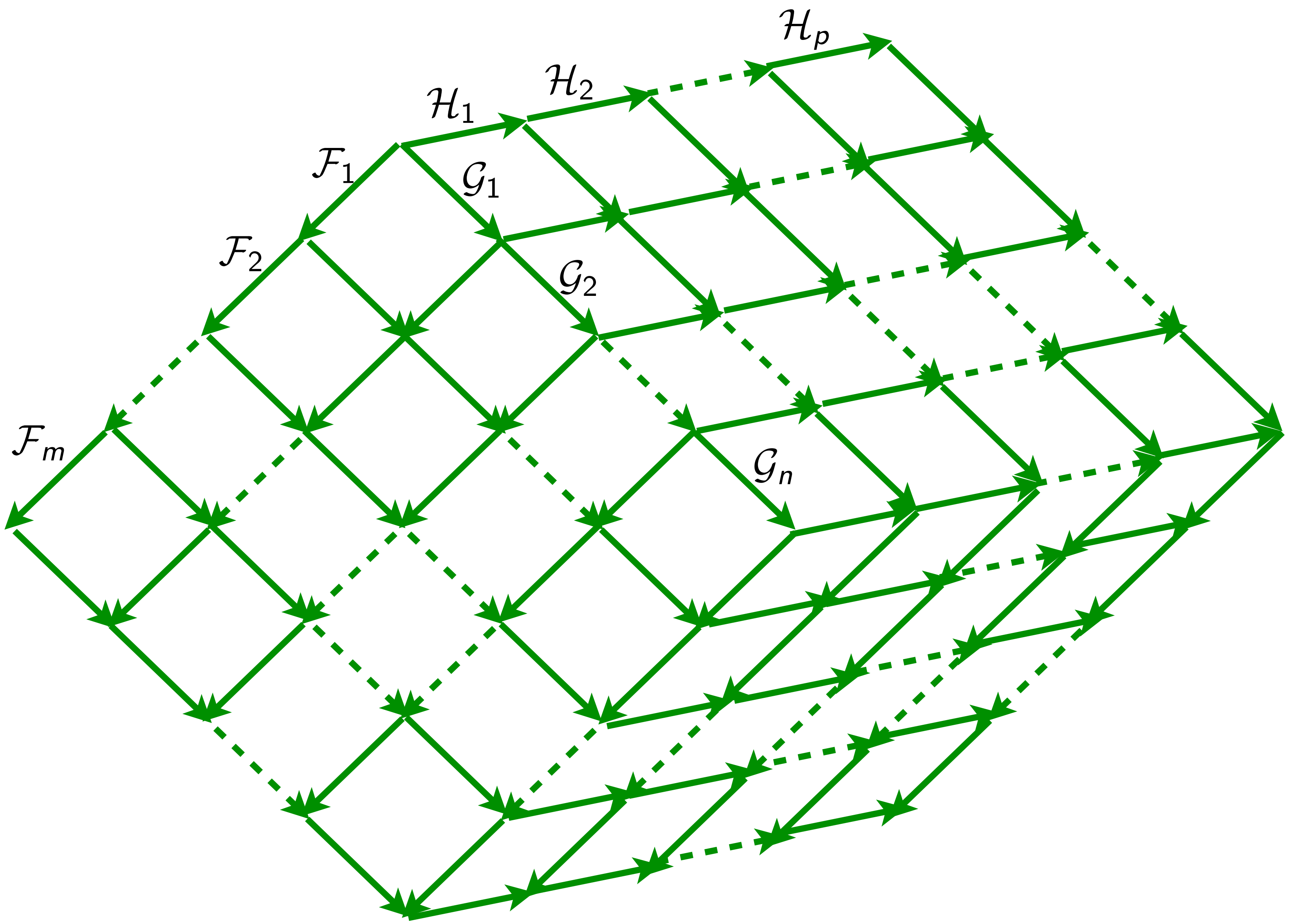
$\rho \sqcup \sigma$ and $\sigma \sqcup \rho$ are cofinal

Residuals of reductions (3/4)





Residuals of reductions (4/4)



- **Proposition** `[Cube Lemma ++]`:

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

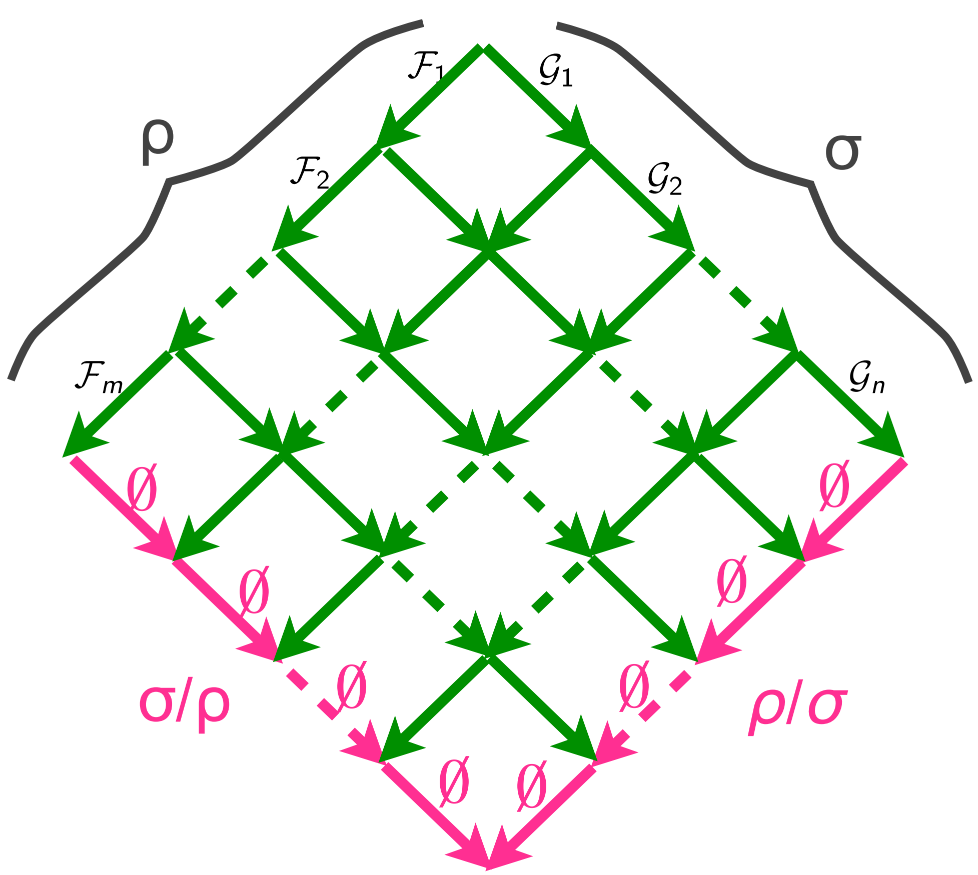
Equivalence by permutations

Equivalence by permutations (1/4)

- **Definition:**

  Let ρ and σ be 2 coinitial reductions. Then ρ is equivalent to σ by permutations, $\rho \simeq \sigma$, iff:

$$\rho/\sigma = \emptyset^m \quad \text{and} \quad \sigma/\rho = \emptyset^n$$



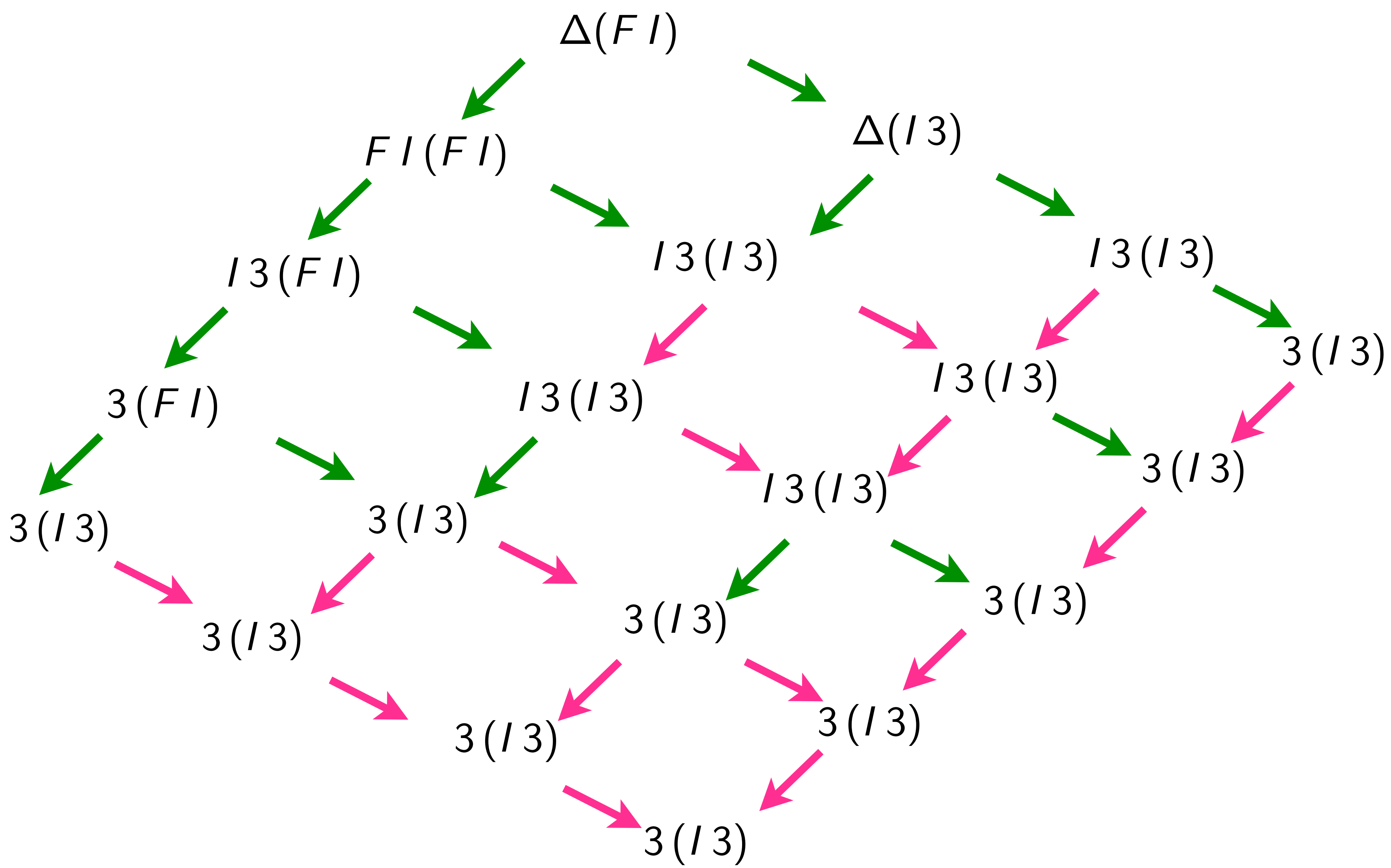
$\rho \simeq \sigma$ means that ρ and σ are coinitial and cofinal
but converse is not true (see later)

Equivalence by permutations (2/4)

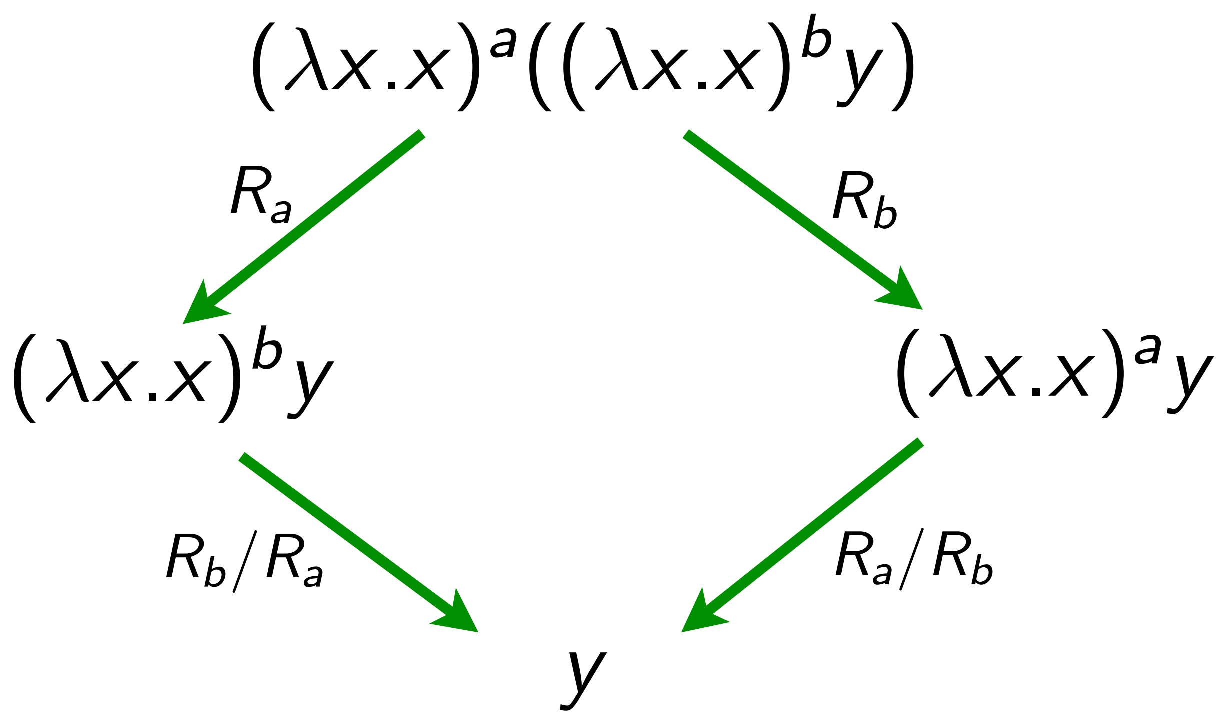
$\Delta = \lambda x.xx$

$F = \lambda f.f\,3$

$I = \lambda x.x$



Equivalence by permutations (3/4)



$$\rho : M = I^a(I^b y) \xrightarrow{R_a} I^b y$$
$$\sigma : M = I^a(I^b y) \xrightarrow{R_b} I^a y$$

- Here $\rho \not\simeq \sigma$ while ρ and σ are coinitial and cofinal in the calculus with no labels

Equivalence by permutations (4/4)

- Same with $0 \not\simeq \rho$ when $\rho : \Delta\Delta \longrightarrow \Delta\Delta$

  $\Delta = \lambda x.xx$

Exercise 1: Give other examples of non-equivalent reductions between same terms.

Exercise 2: Show following equalities

$$\rho/0 = \rho \qquad \emptyset^n/\rho = \emptyset^n$$

$$0/\rho = 0 \qquad 0 \simeq \emptyset^n$$

$$\rho/\emptyset^n = \rho \qquad \rho/\rho = \emptyset^n$$

Equivalence by permutations (4/4)

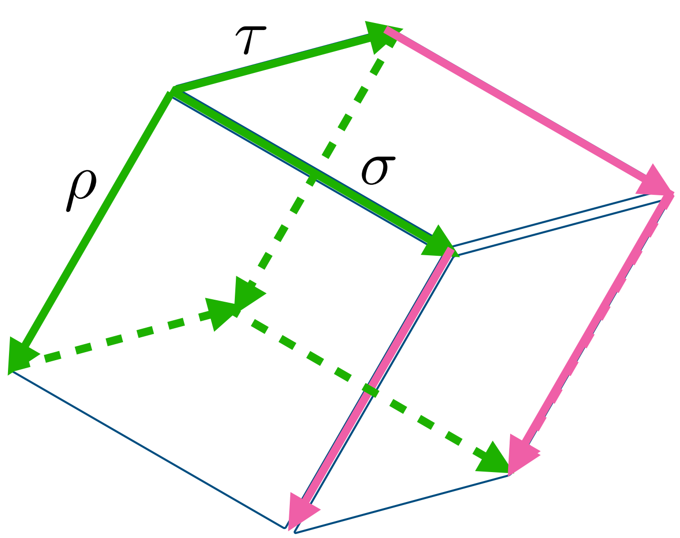
Exercise 3: Show that $\simeq$ is an equivalence relation.

Proof

(i) $\rho \simeq \rho$ obvious

(ii) same with $\rho \simeq \sigma$ implies $\sigma \simeq \rho$

(iii) $\rho \simeq \sigma \simeq \tau$ implies $\rho \simeq \tau$??



Properties of permutations (1/3)

- **Proposition**

$$(i) \quad \rho \simeq \sigma \;\text{ iff }\; \forall \tau.\, \tau/\rho = \tau/\sigma$$

$$(ii) \quad \rho \simeq \sigma \;\text{ implies }\; \rho/\tau = \sigma/\tau$$

$$(iii) \quad \rho \simeq \sigma \;\text{ iff }\; \tau\rho \simeq \tau\sigma$$

$$(iv) \quad \rho \simeq \sigma \;\text{ implies }\; \rho\tau \simeq \sigma\tau$$

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

Proof

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

Thus $\tau/(\rho \sqcup \sigma) = \tau/(\rho\,(\sigma/\rho)) = \tau/\rho/(\sigma/\rho) = \tau/\rho/\emptyset^m = \tau/\rho$

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

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

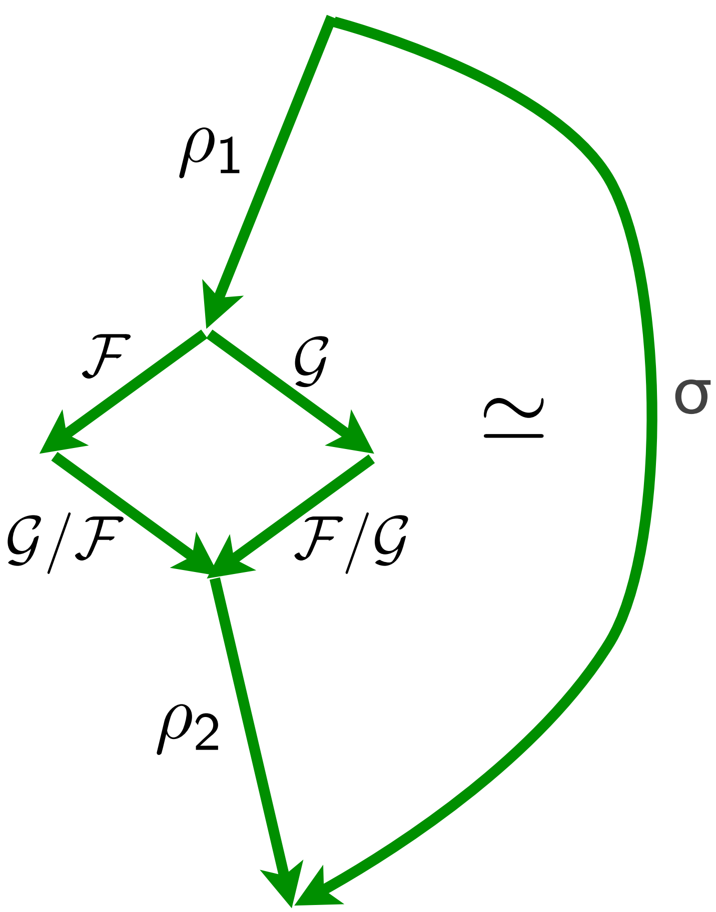
Conversely, take $\tau = \rho$ and $\tau = \sigma$.

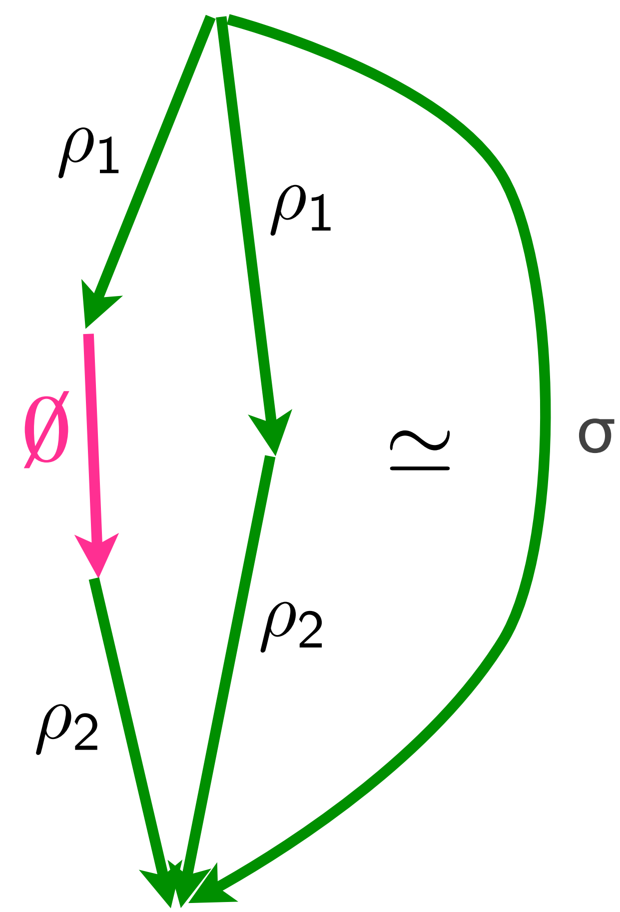
Properties of permutations (2/3)

- **Proposition** $\simeq$ is the smallest congruence containing

$$\mathcal{F}\,(\mathcal{G}/\mathcal{F}) \simeq \mathcal{G}\,(\mathcal{F}/\mathcal{G})$$

$$0 \simeq \emptyset$$



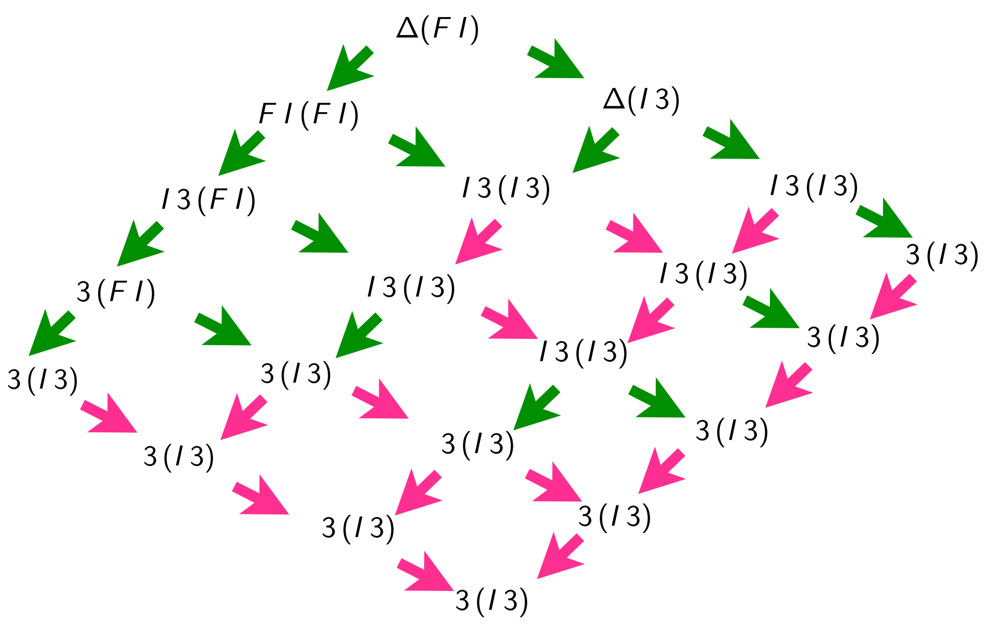


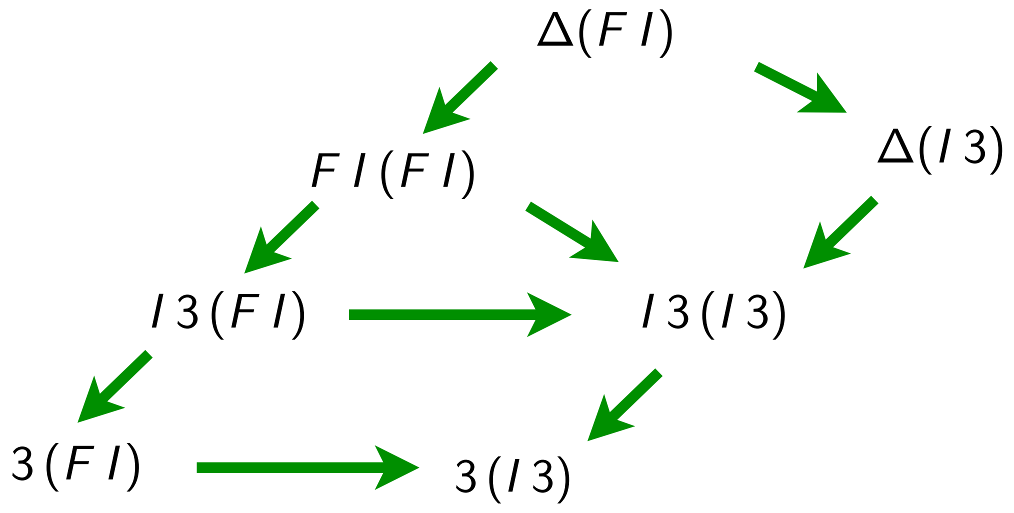
Properties of permutations (3/3)

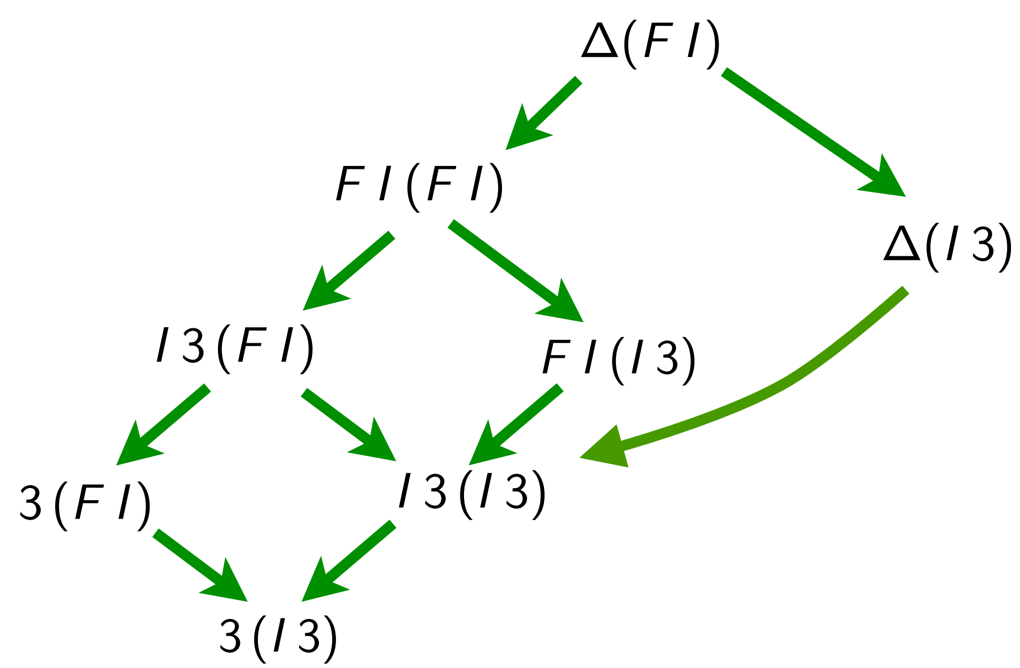
$\Delta = \lambda x.xx$

$F = \lambda f.f\,3$

$I = \lambda x.x$







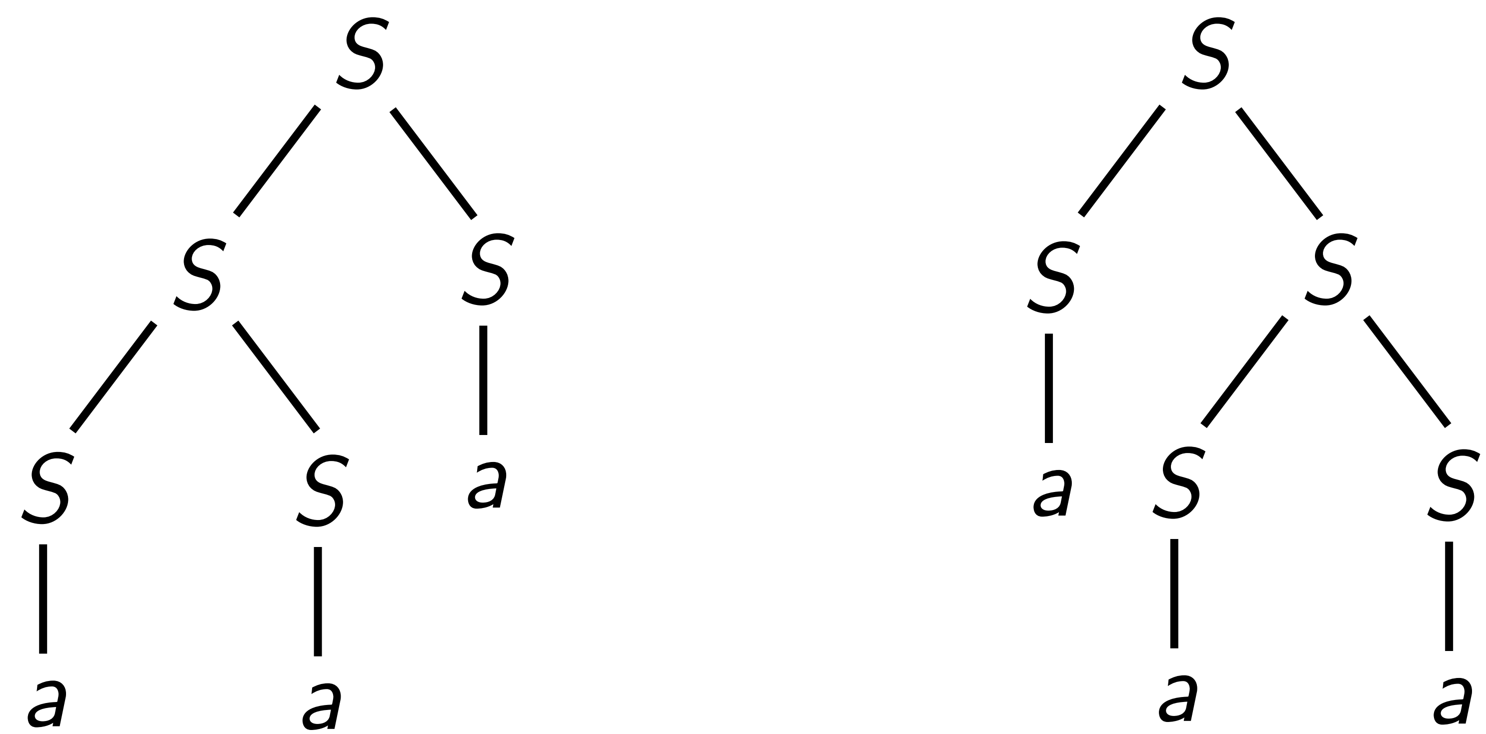
Beyond the λ-calculus

Context-free languages

- permutations of derivations in contex-free languages

$S \rightarrow SS$

$S \rightarrow a$



- each parse tree corresponds to an equivalence class

Term rewriting

- recursive program schemes [Berry-JJL'77]
- permutations of derivations in orthogonal TRS [Huet-JJL'81]
- permutations of derivations are defined with critical pairs
- critical pairs make conflicts
- only 2nd definition of equivalence works [Boudol'82]
- interaction systems [Asperti-Laneve'93]

Process algebras

- similar to TRS [Boudol-Castellani'82]
- connection to event structures [Laneve'84]

PCF

- LCF considered as a programming language [Plotkin'74]

M, N, P ::=	x	variable
	| λx.M | M N	abstraction application
	| $\underline{n}$	integer constant
	| M ⊗ N	⊗ ∈ {+,−,×,÷}
	| ifz M then N then N	conditionnal
	| μx.M	recursive definition

β	(λx.M)N → M {x := N}
op	$\underline{m}$ ⊗ $\underline{n}$ → $\underline{m \otimes n}$
cond1	ifz $\underline{0}$ then M else N → M
cond2	ifz $\underline{n+1}$ then M else N → N
μ	μx.M → M {x := μx.M}

Exemples de termes

$\texttt{Fact}(3)$

$\texttt{Fact} = Y(\lambda f.\lambda x.\ \texttt{ifz}\ x\ \texttt{then}\ 1\ \texttt{else}\ x \star f(x-1))$

$Y = \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$

s'écrit

$(\lambda\,\texttt{Fact}\,.\,\texttt{Fact}(3))$

$(\ (\lambda Y.Y(\lambda f.\lambda x.\ \texttt{ifz}\ x\ \texttt{then}\ 1\ \texttt{else}\ x \star f(x-1)))$

$(\lambda f.(\lambda x.f(xx))(\lambda x.f(xx)))\ )$

(\Fact.Fact3)((\y.y(\f.\x.ifz x then 1 else x * f(x-1)))(\f.Yf))

(\y.y(\f.\x.ifz x then 1 else x * f(x-1)))(\f.Yf)3

(\f.Yf)(\f.\x.ifz x then 1 else x * f(x-1))3

(\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))3

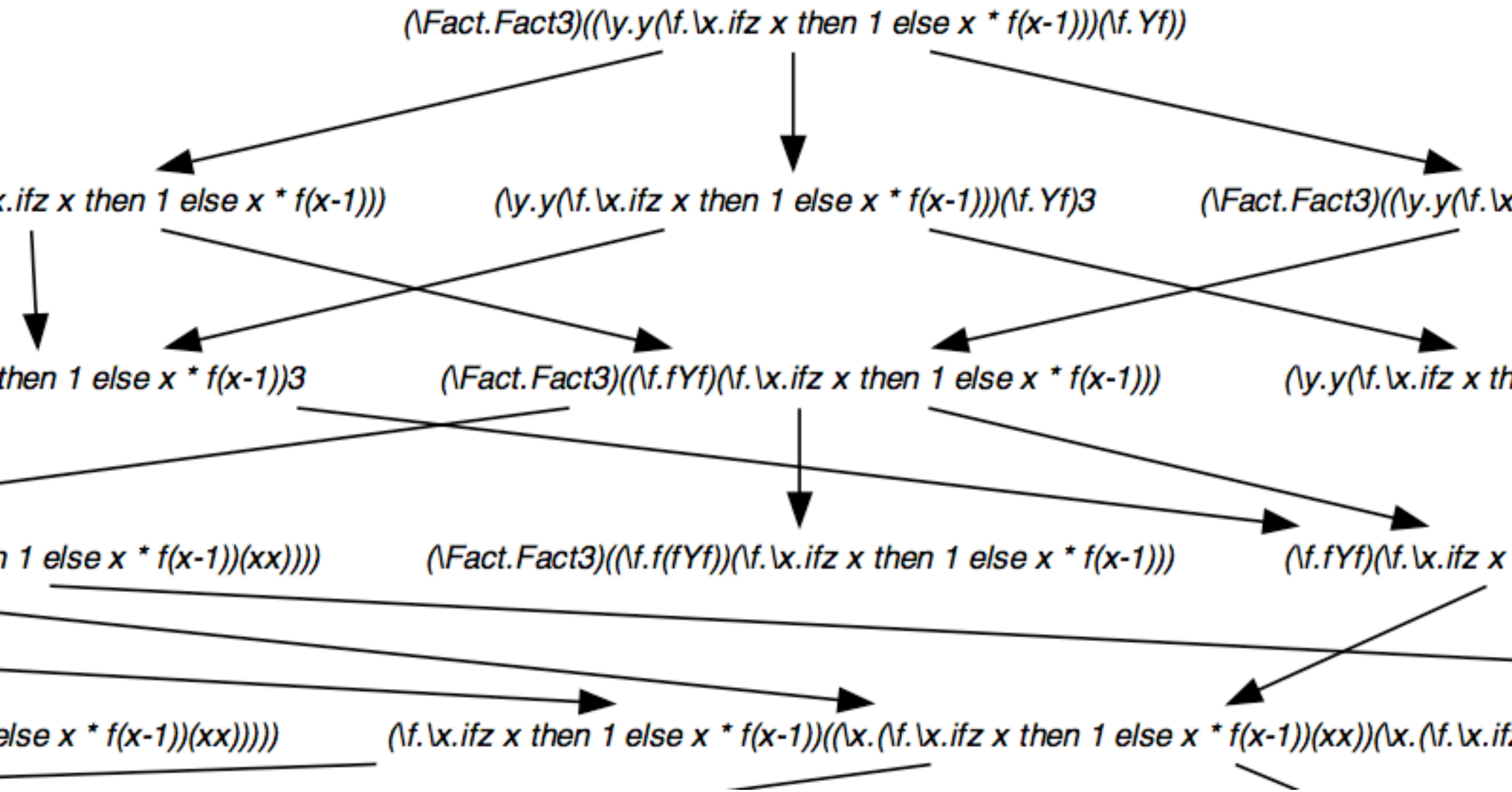
(\f.\x.ifz x then 1 else x * f(x-1))((\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx))(\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx)))3

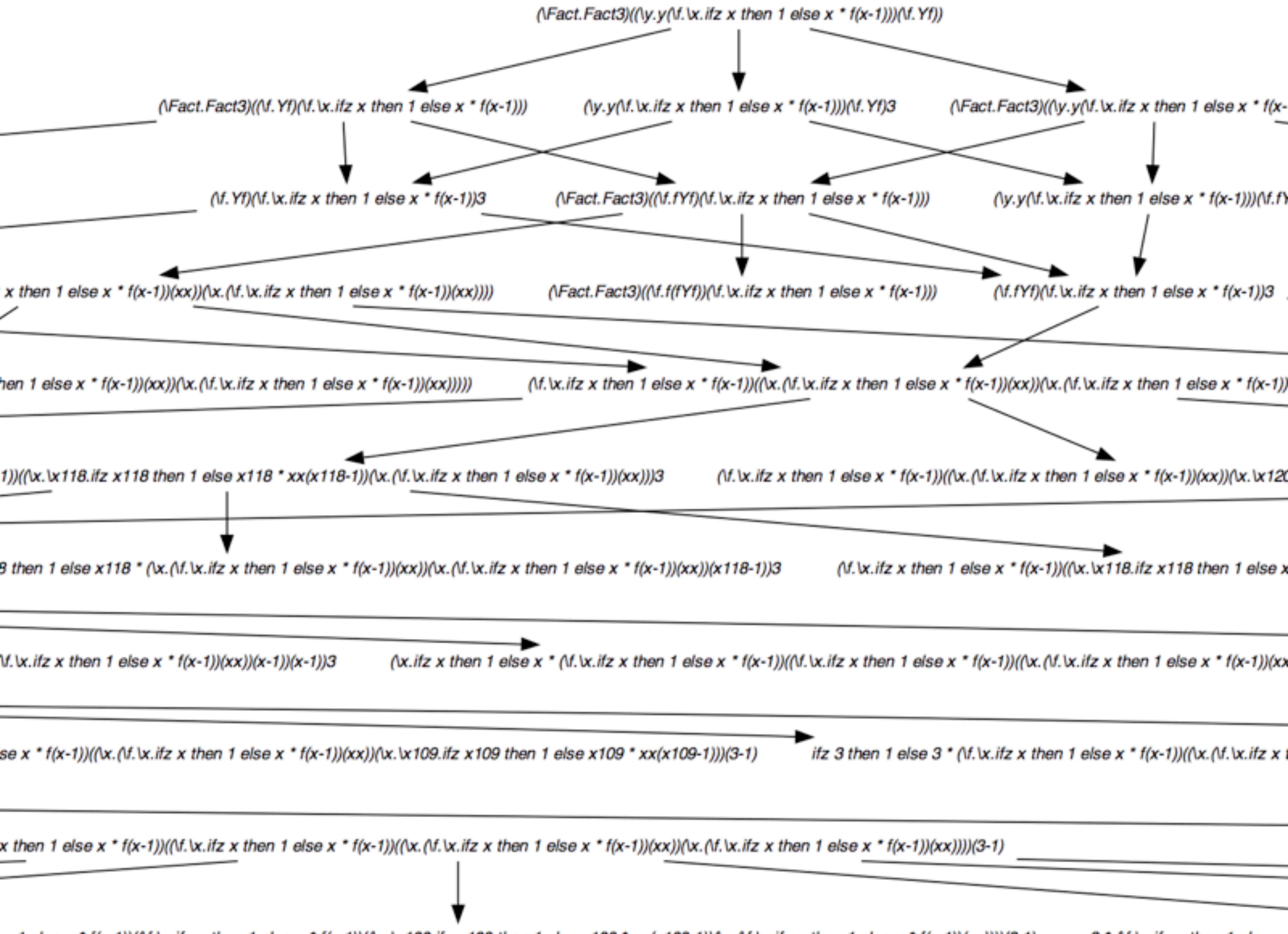
(\x.ifz x then 1 else x * (\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx))(\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx))(x-1))3

ifz 3 then 1 else 3 * (\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))(3-1)

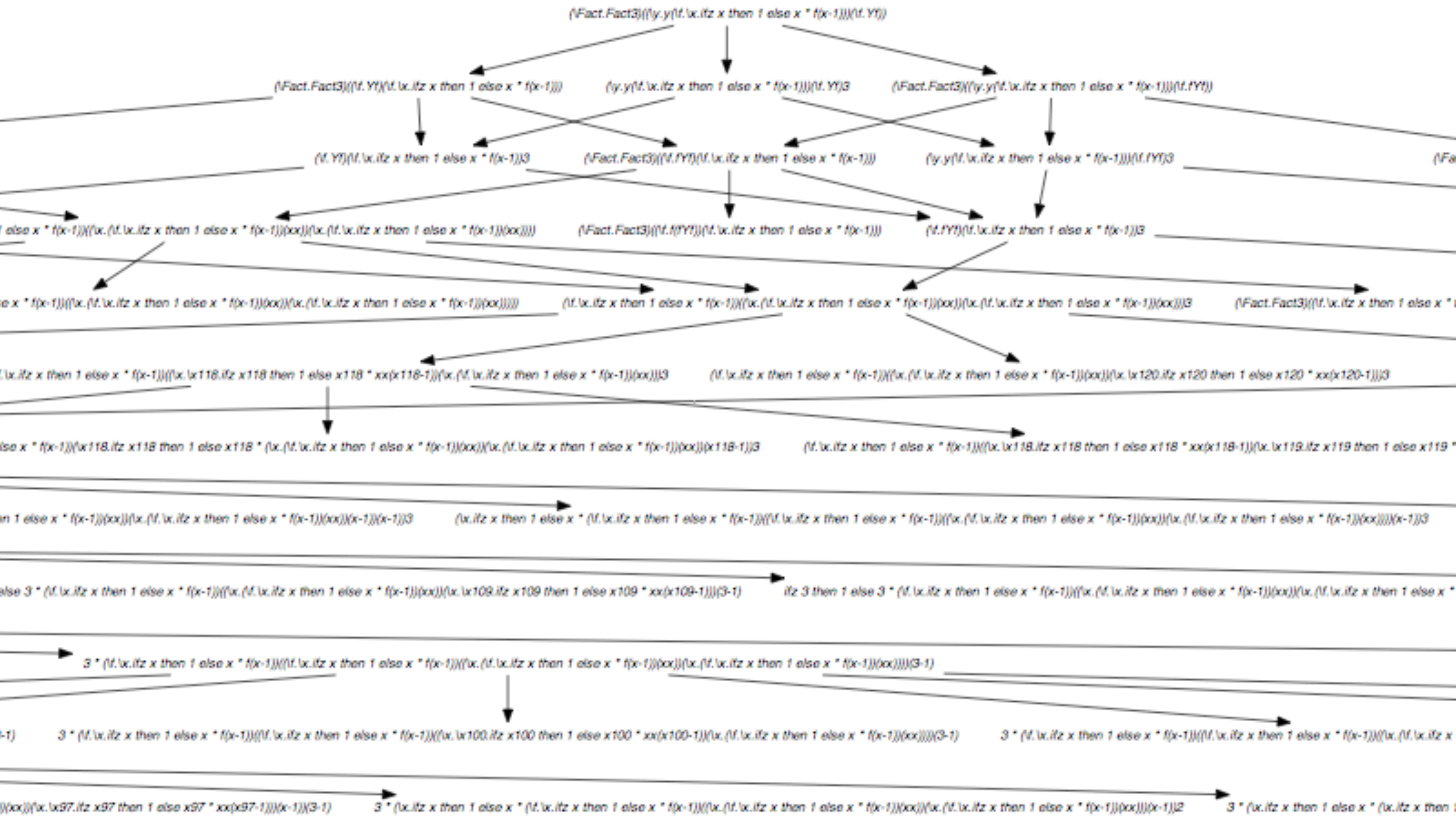
3 * (\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx))(\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx))(3-1)

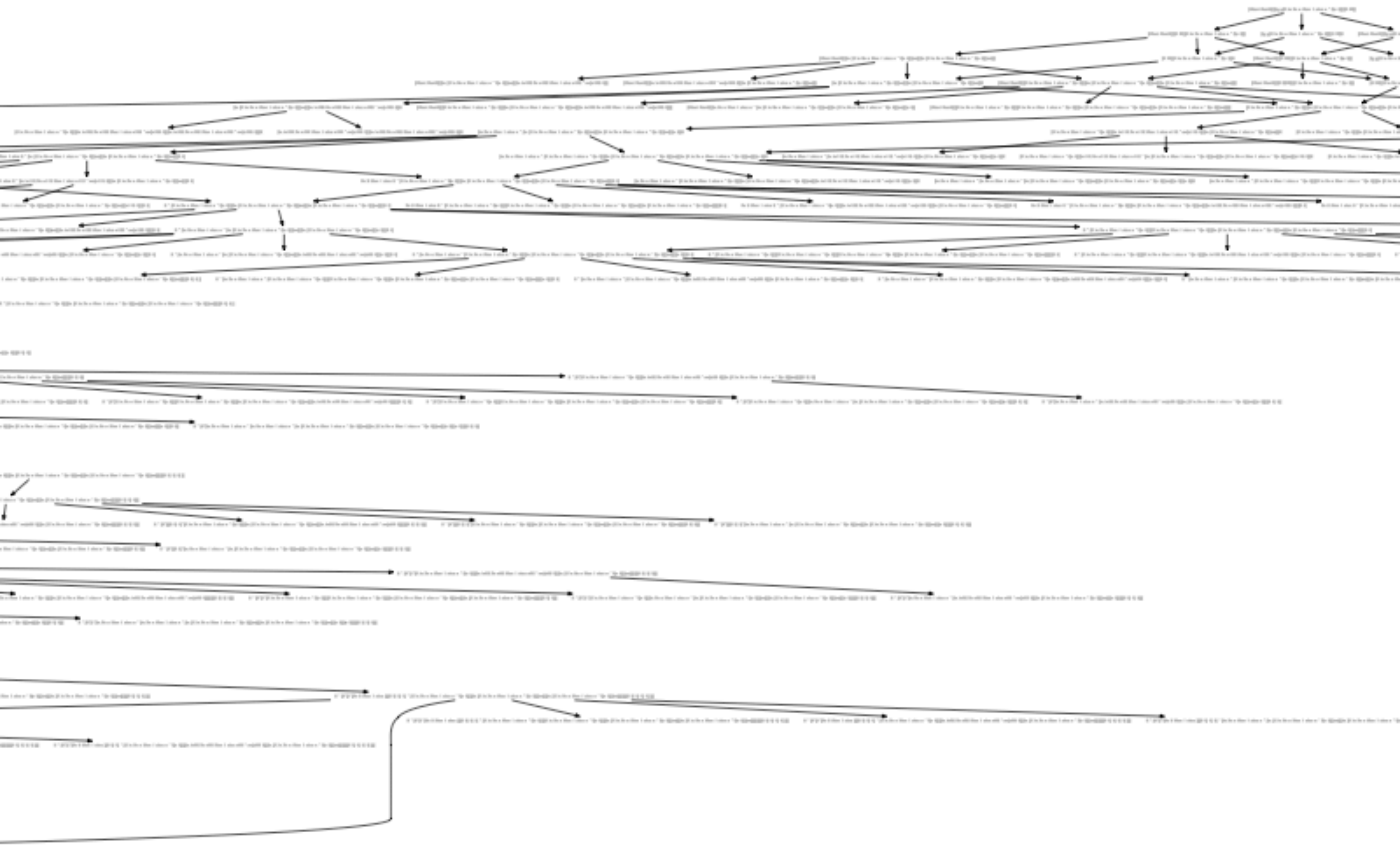
3 * (\f.\x.ifz x then 1 else x * f(x-1))((\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx))(\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx)))(3-1)

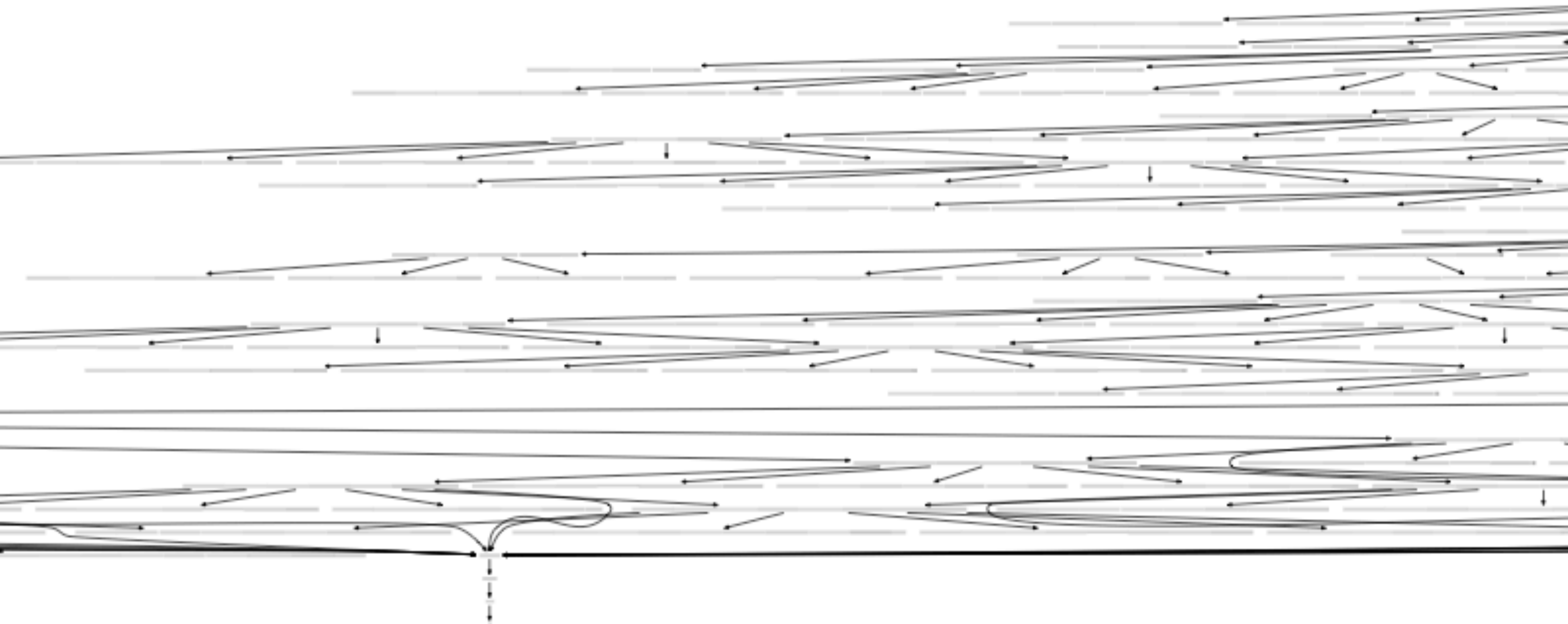

(\Fact.Fact3)((\y.y(\f.\x.ifz x then 1 else x * f(x-1)))(\f.Yf))
.ifz x then 1 else x * f(x-1)))
(\y.y(\f.\x.ifz x then 1 else x * f(x-1)))(\f.Yf)3
(\Fact.Fact3)((\y.y(\f.\x
then 1 else x * f(x-1))3
(\Fact.Fact3)((\f.fYf)(\f.\x.ifz x then 1 else x * f(x-1)))
(\y.y(\f.\x.ifz x th
1 else x * f(x-1))(xx))))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

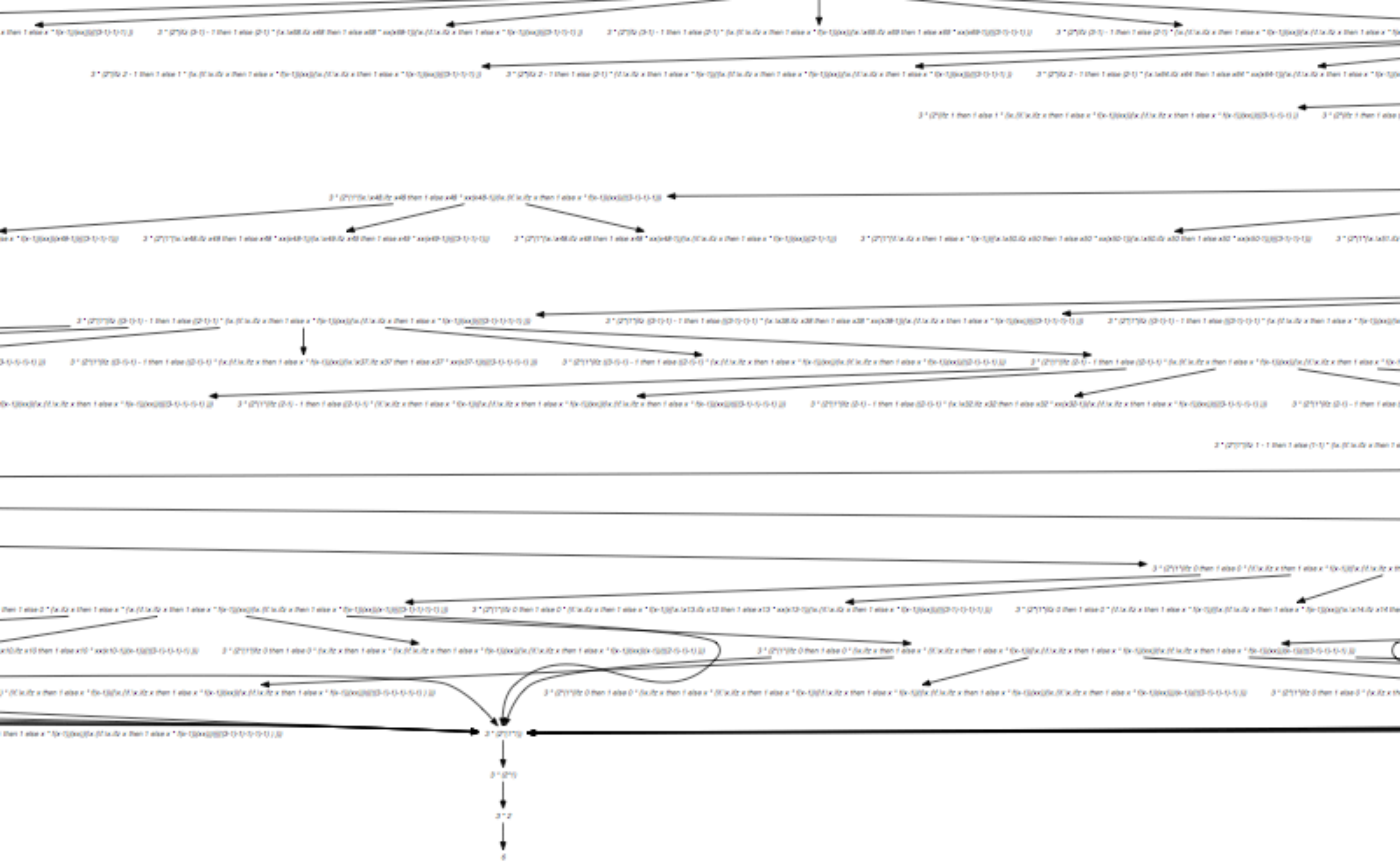


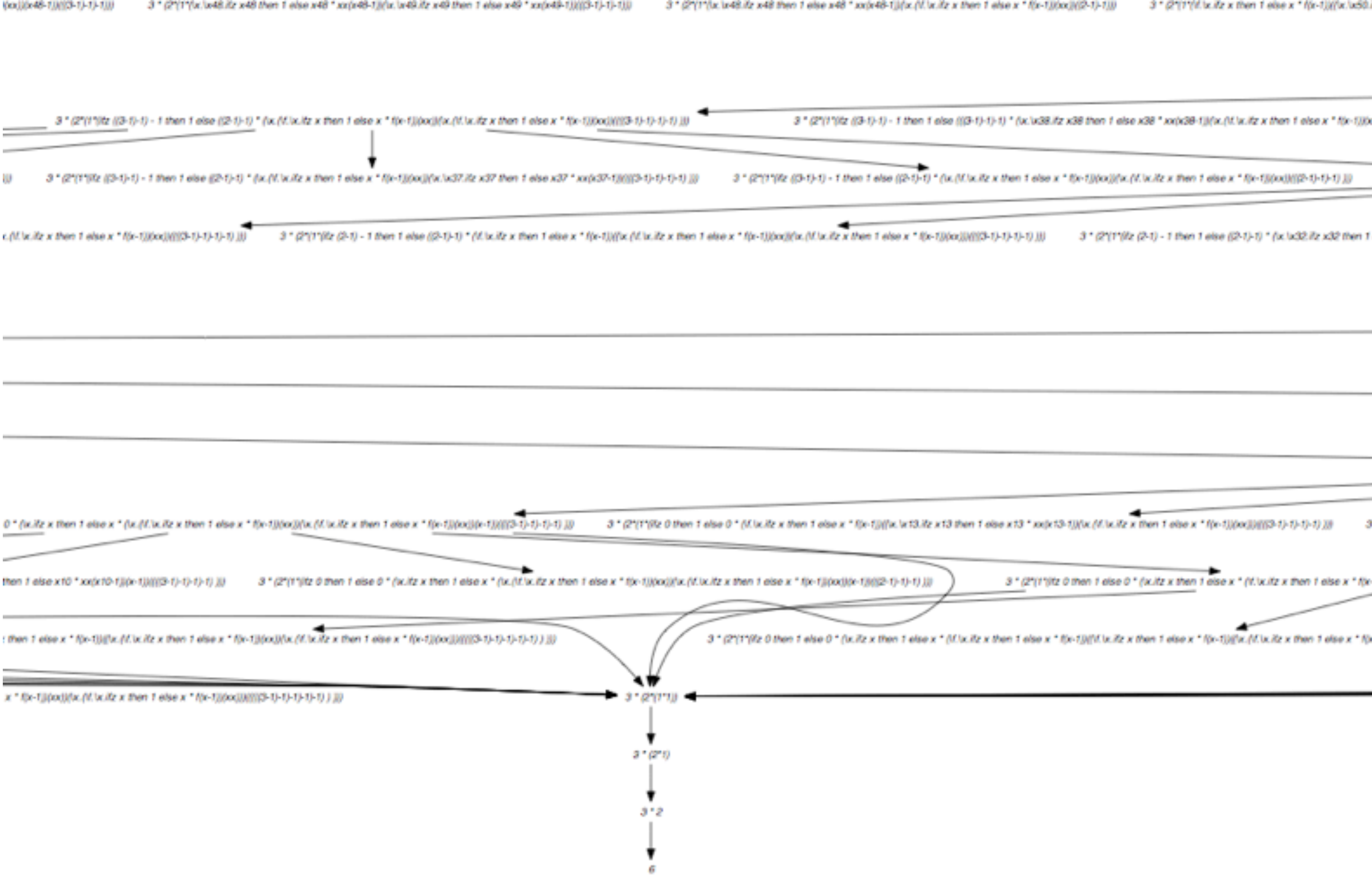
(\Fact.Fact3)((\y.y(\f.\x.ifz x then 1 else x * f(x-1)))(\f.Yf))
(\Fact.Fact3)((\f.Yf)(\f.\x.ifz x then 1 else x * f(x-1)))
(\y.y(\f.\x.ifz x then 1 else x * f(x-1)))(\f.Yf)3
(\Fact.Fact3)((\y.y(\f.\x.ifz x then 1 else x * f(x-
(\f.Yf)(\f.\x.ifz x then 1 else x * f(x-1))3
(\Fact.Fact3)((\f.fYf)(\f.\x.ifz x then 1 else x * f(x-1)))
(\y.y(\f.\x.ifz x then 1 else x * f(x-1)))(\f.fY
x then 1 else x * f(x-1))(xx))(\x.(\f.\x.ifz x then 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 then 1 else x * f(x-1))3
hen 1 else x * f(x-1))(xx))(\x.(\f.\x.ifz x then 1 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.ifz x then 1 else x * f(x-1))
1))((\x.\x118.ifz x118 then 1 else x118 * xx(x118-1))(\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx)))3
(\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.\x120
8 then 1 else x118 * (\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))(x118-1))3
(\f.\x.ifz x then 1 else x * f(x-1))((\x.\x118.ifz x118 then 1 else x
\f.\x.ifz x then 1 else x * f(x-1))(xx))(x-1))(x-1))3
(\x.ifz x then 1 else x * (\f.\x.ifz x then 1 else x * f(x-1))((\f.\x.ifz x then 1 else x * f(x-1))((\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx
se x * f(x-1))((\x.(\f.\x.ifz x then 1 else x * f(x-1))(xx))(\x.\x109.ifz x109 then 1 else x109 * xx(x109-1)))(3-1)
ifz 3 then 1 else 3 * (\f.\x.ifz x then 1 else x * f(x-1))((\x.(\f.\x.ifz x t
x then 1 else x * f(x-1))((\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.ifz x then 1 else x * f(x-1))(xx))))(3-1)

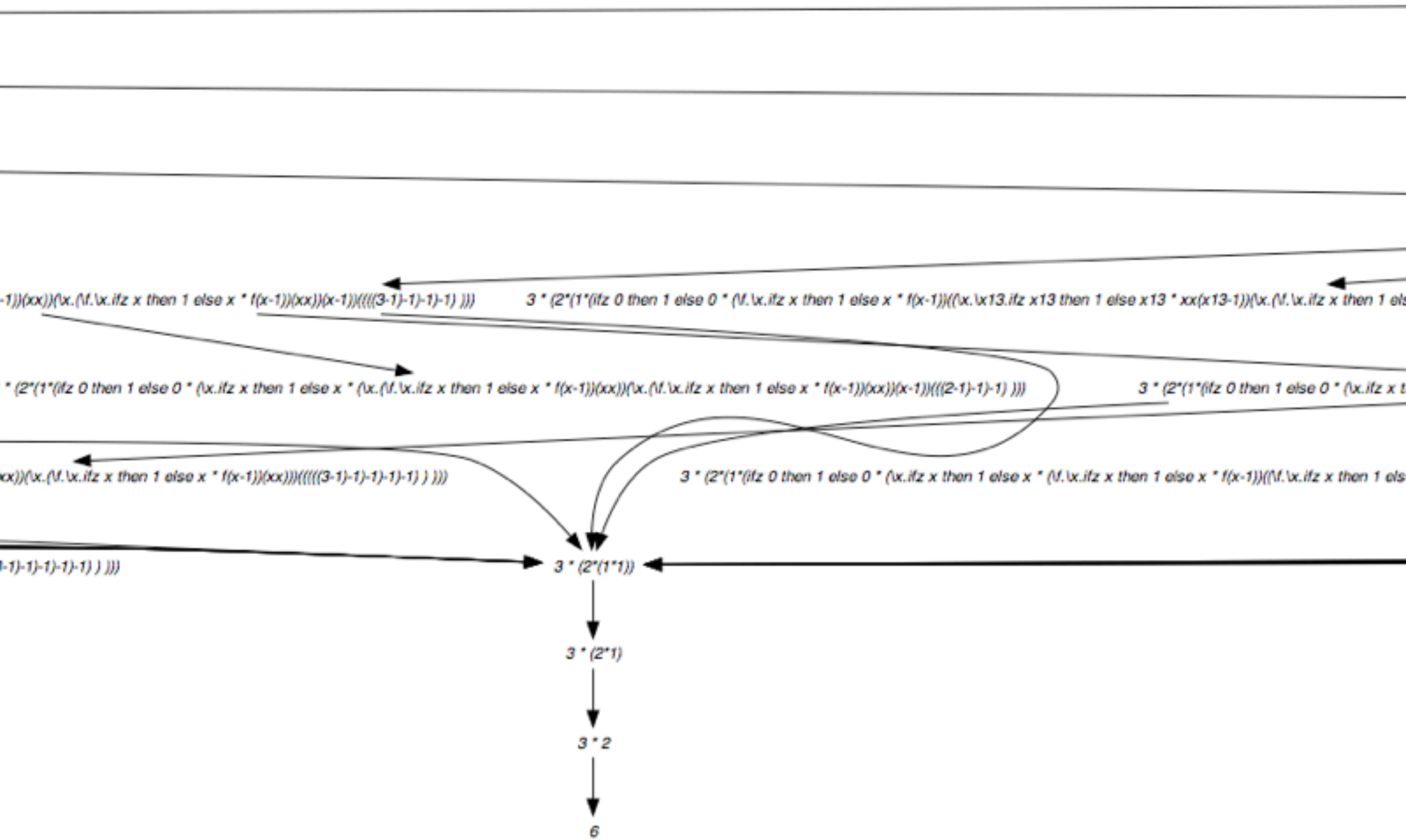

(\Fact.Fact3)((\y.y(\f.\x.ifz x then 1 else x * f(x-1)))(\f.Yf))
(\Fact.Fact3)((\f.Yf)(\f.\x.ifz x then 1 else x * f(x-1)))
(\y.y(\f.\x.ifz x then 1 else x * f(x-1)))(\f.Yf)3
(\Fact.Fact3)((\y.y(\f.\x.ifz x then 1 else x * f(x-1)))(\f.fYf))
(\f.Yf)(\f.\x.ifz x then 1 else x * f(x-1))3
(\Fact.Fact3)((\f.fYf)(\f.\x.ifz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

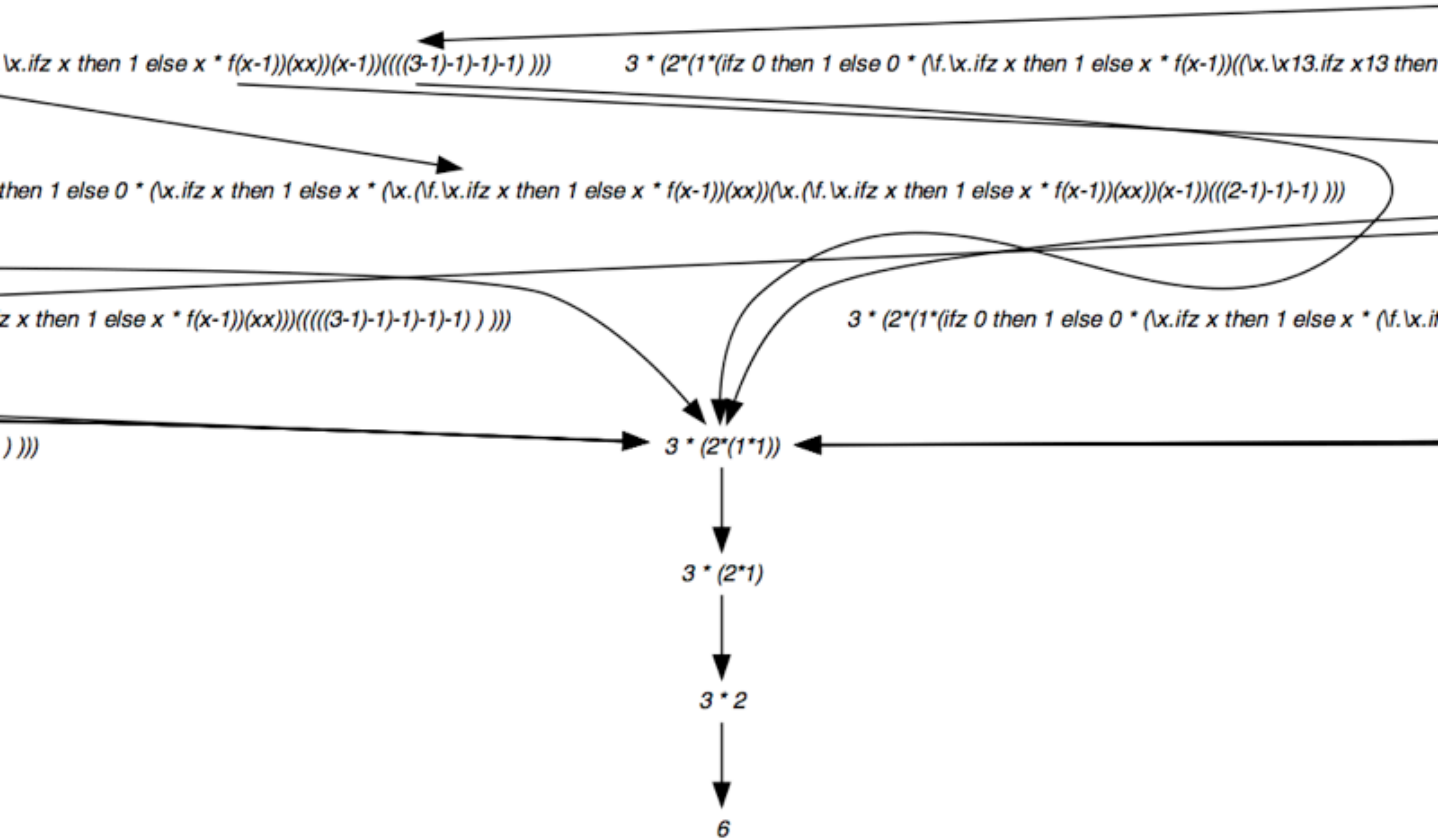







3 * (2*(1*1))
3 * (2*1)
3 * 2
6


-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.i
))))
3 * (2*(1*1))
3 * (2*1)
3 * 2
6

Exercises

Parallel moves

- **Lemma** $M \xrightarrow{\mathcal{F}} N,\ M \xrightarrow{\mathcal{G}} P \ \Rightarrow\ N \xrightarrow{\mathcal{G}} Q,\ P \xrightarrow{\mathcal{F}} Q$

Proof

Case 1: $M = x = N = P = Q$. Obvious.

Case 2: $M = \lambda x.M_1,\ N = \lambda x.N_1,\ P = \lambda x.P_1$. Obvious by induction on M_1

Case 3: (App-App) $M = M_1 M_2,\ N = N_1 N_2,\ P = P_1 P_2$. Obvious by induction on $M_1,\ M_2$.

Case 4: (Red'-Red') $M = (\lambda x.M_1)^a M_2,\ N = (\lambda x.N_1)^a N_2,\ P = (\lambda x.P_1)^a P_2,\ a \notin \mathcal{F} \cup \mathcal{G}$

Then induction on $M_1,\ M_2$.

Case 4: (beta-Red') $M = (\lambda x.M_1)^a M_2,\ N = N_1\{x := N_2\},\ P = (\lambda x.P_1)^a P_2,\ a \in \mathcal{F}, a \notin \mathcal{G}$

By induction $N_1 \xrightarrow{\mathcal{G}} Q_1,\ P_1 \xrightarrow{\mathcal{F}} Q_1$. And $N_2 \xrightarrow{\mathcal{G}} Q_2,\ P_1 \xrightarrow{\mathcal{F}} Q_2$.

By lemma, $N_1\{x := N_2\} \xrightarrow{\mathcal{G}} Q_1\{x := Q_2\}$. And $(\lambda x.P_1)^a P_2 \xrightarrow{\mathcal{F}} Q_1\{x := Q_2\}$

Case 5: (beta-beta) $M = (\lambda x.M_1)^a M_2,\ N = N_1\{x := N_2\},\ P = P_1\{x := P_2\},\ a \in \mathcal{F} \cap \mathcal{G}$

As before with same lemma.

Parallel moves

- Lemma $M \xrightarrow{\mathcal{F}} N,\ P \xrightarrow{\mathcal{F}} Q \ \Rightarrow\ M\{x := P\} \xrightarrow{\mathcal{F}} N\{x := Q\}$

  Proof: exercise!

- Lemma [subst] $M\{x := N\}\{y := P\} = M\{y := P\}\{x := N\{y := P\}\}$

  when x not free in P

  Proof: exercise!

  this lemma about distribution of substitution is critical for the Church-Rosser property.