fête
pour

Georges Gonthier

23 / 06 / 25

Inria Paris

jean-jacques.levy@inria.fr
jeanjacqueslevy.net/talks

Le Monde

Consulter le journal

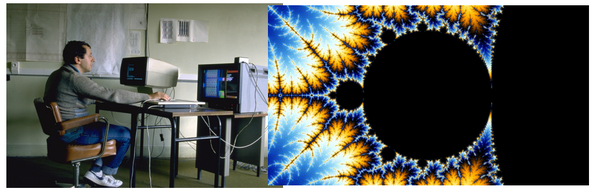
Actualités · Économie · Vidéos · Débats · Culture · Le Goût du Monde

SCIENCES

Portrait Georges Gonthier, « hacker » de théorèmes

L'informaticien a développé des programmes de vérifications des preuves mathématiques

Plan



colorix ‘86

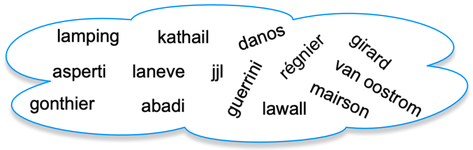
sophia

phD ‘88

[Sémantiques et modèles d'exécution des langages réactifs synchrones](#) :
application à Esterel / Georges Gonthier ;
[sous la direction de Gérard Berry]

bell labs

lambda opt ‘91



rocq

join calculus

The join-calculus is a process calculus developed at INRIA.
with Cédric Fournet

gc tlp ’95

Verifying the Safety of a Practical Concurrent Garbage Collector

Georges Gonthier

INRIA Rocquencourt
78153 LE CHESNAY CEDEX
FRANCE

Portable, unobtrusive garbage collection for multiprocessor systems (POPL'94)

Damien Doligez, Georges Gonthier

a501

cambridge

4 couleurs ‘04

joint centre

Microsoft Research - Inria
JOINT CENTRE

feit thompson ’12

math comp
[ssr, bigops, .20 ans]

mathlib

saclay

?

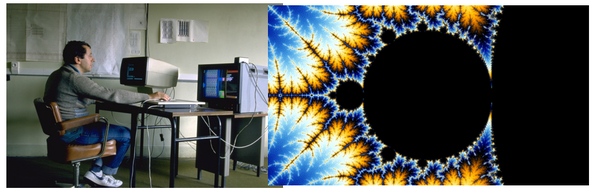
X- pi

X- cc

X-concours

ÉCOLE POLYTECHNIQUE

Plan



colorix ‘86

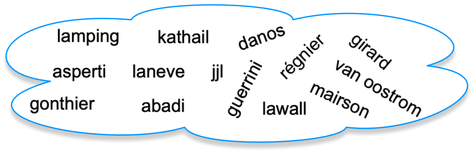
sophia

phD ‘88

[Sémantiques et modèles d'exécution des langages réactifs synchrones](#) :
application à Esterel / Georges Gonthier ;
[sous la direction de Gérard Berry]

bell labs

lambda opt ‘91



rocq

join calculus

The join-calculus is a process calculus developed at INRIA.
with Cédric Fournet

gc tlp ’95

Verifying the Safety of a Practical Concurrent Garbage Collector

Georges Gonthier

INRIA Rocquencourt
78153 LE CHESNAY CEDEX
FRANCE

Portable, unobtrusive garbage collection for multiprocessor systems (POPL'94)

Damien Doligez, Georges Gonthier

a501

cambridge

4 couleurs ‘04

joint centre

Microsoft Research - Inria
JOINT CENTRE

feit thompson ’12

math comp
[ssr, bigops, .20 ans]

mathlib

saclay

?

X- pi

X- cc

X-concours

ÉCOLE POLYTECHNIQUE

Plan

sophia

bell labs

rocq

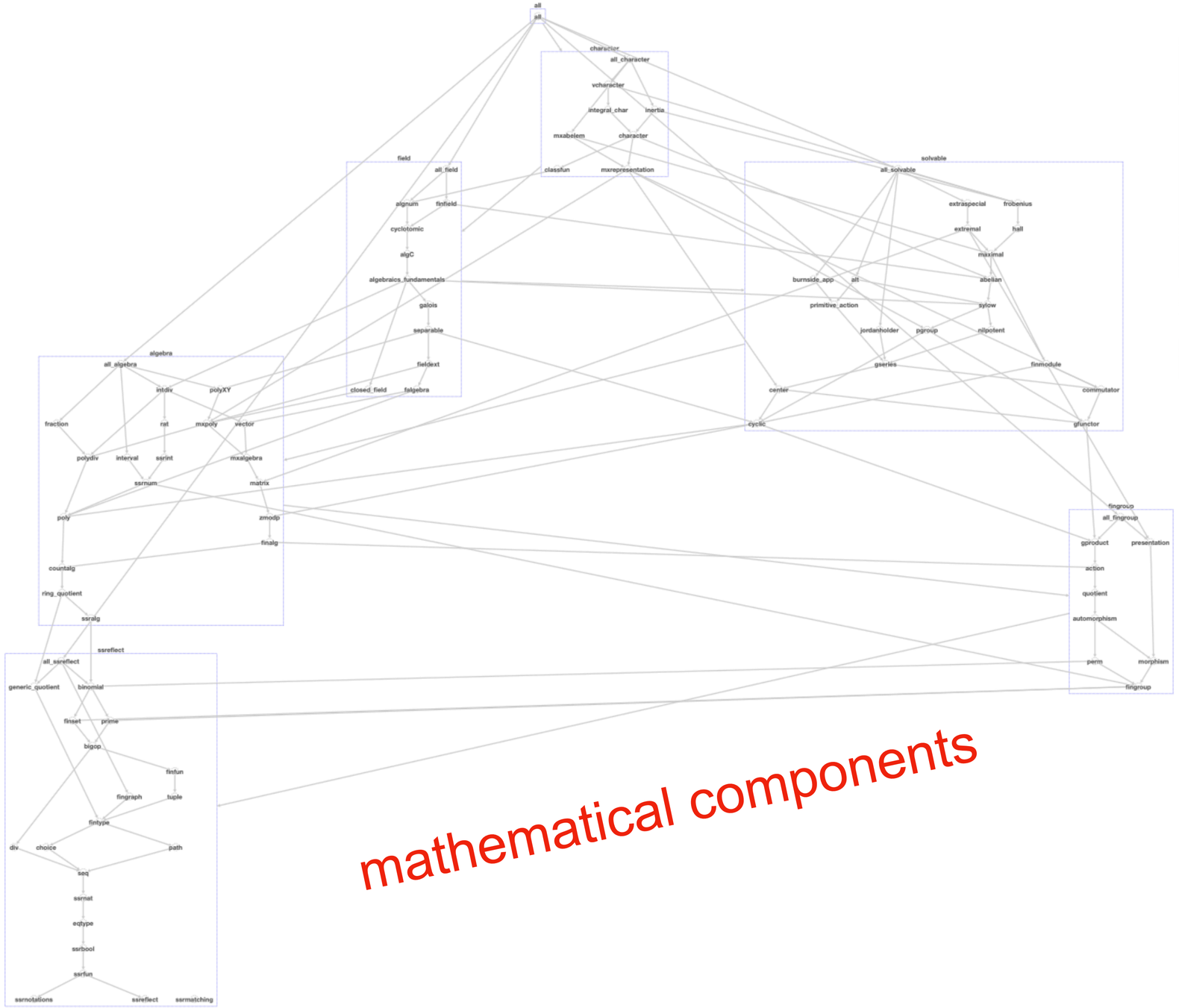
cambridge

joint centre

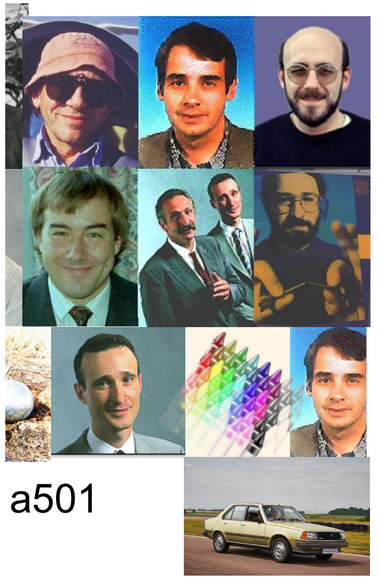
Microsoft Research - Inria
JOINT CENTRE

saclay

?



mathematical components



a501

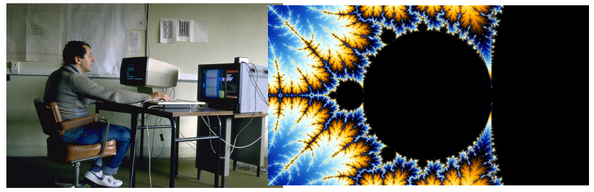
X- pi

X- cc

X-concours

ÉCOLE POLYTECHNIQUE

Plan



colorix ‘86

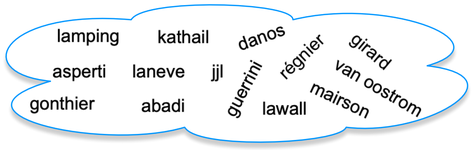
sophia

phD ‘88

[Sémantiques et modèles d'exécution des langages réactifs synchrones](#) :
application à Esterel / Georges Gonthier ;
[sous la direction de Gérard Berry]

bell labs

lambda opt ‘91



rocq

join calculus

The join-calculus is a process calculus developed at INRIA.
with Cédric Fournet

gc tlp ’95

Verifying the Safety of a Practical Concurrent Garbage Collector

Georges Gonthier

INRIA Rocquencourt
78153 LE CHESNAY CEDEX
FRANCE

Portable, unobtrusive garbage collection for multiprocessor systems (POPL'94)

Damien Doligez, Georges Gonthier

a501

cambridge

4 couleurs ‘04

joint centre

Microsoft Research - Inria
JOINT CENTRE

feit thompson ’12

math comp
[ssr, bigops, .20 ans]

mathlib

saclay

?

X- pi

X- cc

X-concours

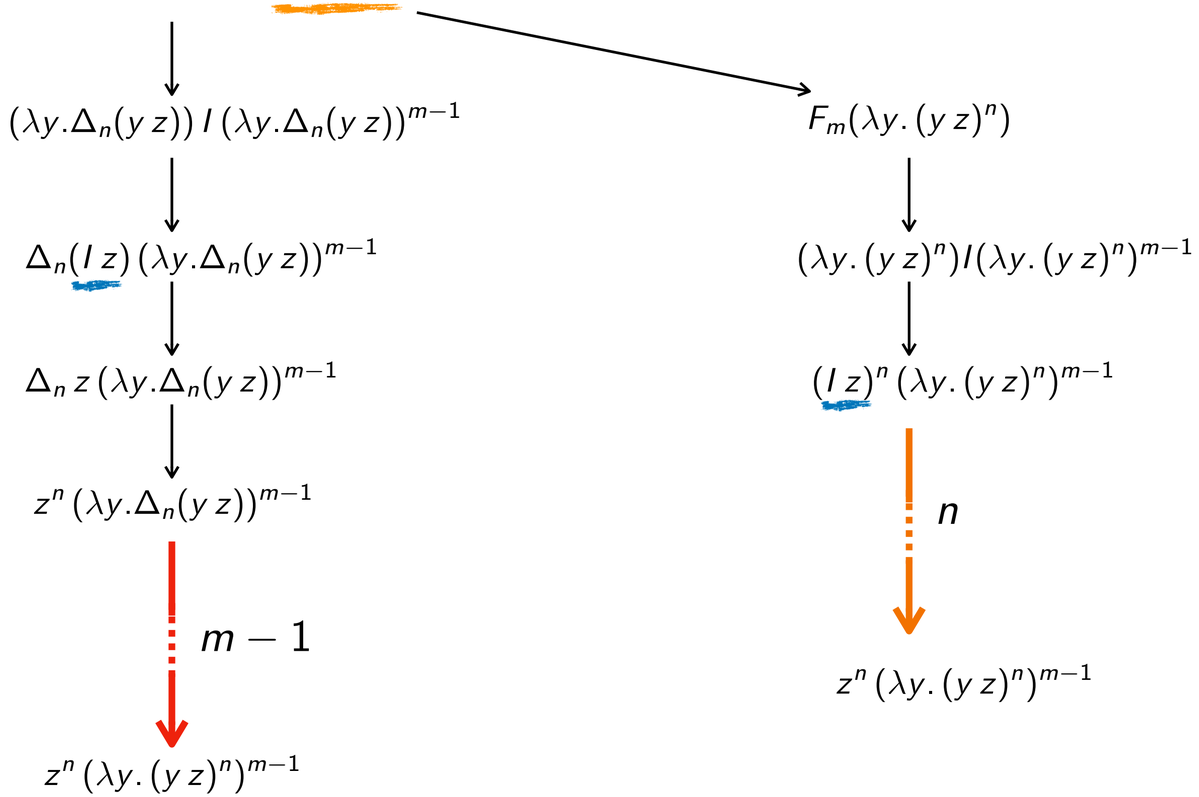
ÉCOLE POLYTECHNIQUE

duplications in the λ-calculus

Initial motivation (side remark)

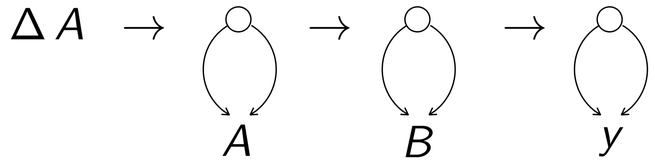
- no single-redex reduction strategy can be optimal

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



Duplication of redexes

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

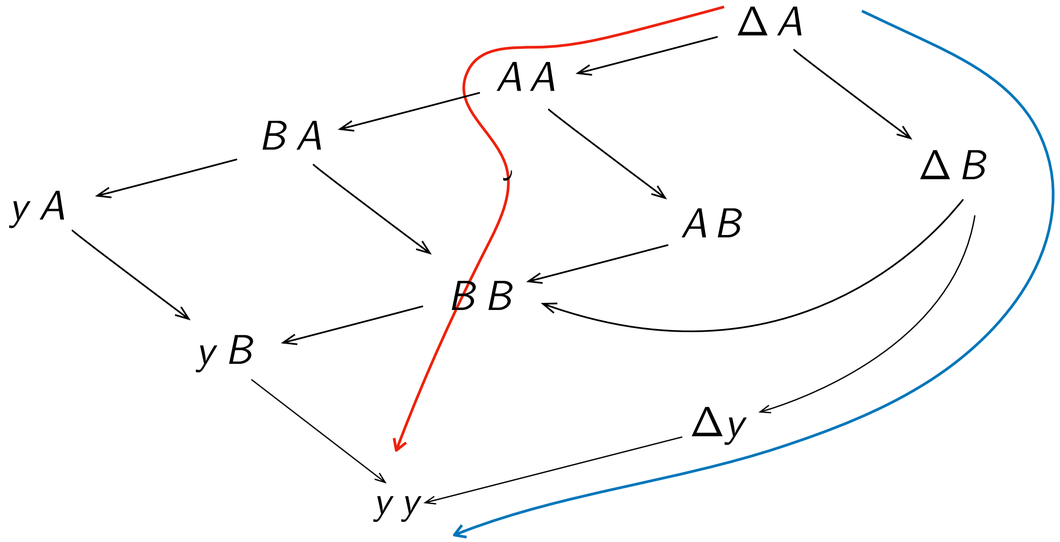


$\Delta = \lambda x.\, x\, x$

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

$I = \lambda x.\, x$

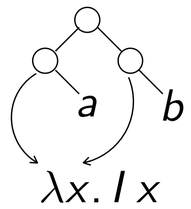
$A = F\, I \;\rightarrow\; I\, y = B \;\rightarrow\; y$

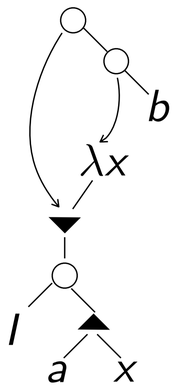


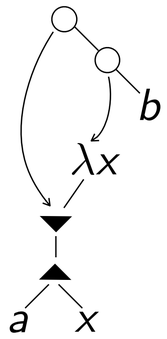
- optimal reductions has to deal with **parallel reductions**

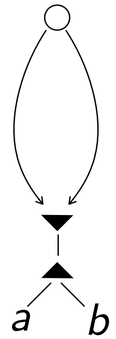
Duplication of redexes

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

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

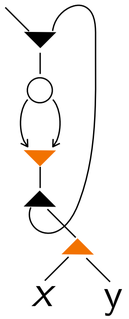
$\rightarrow$ 

$\rightarrow$ 

$\rightarrow$ 

$= \; a\, b$

- possible sharing !!



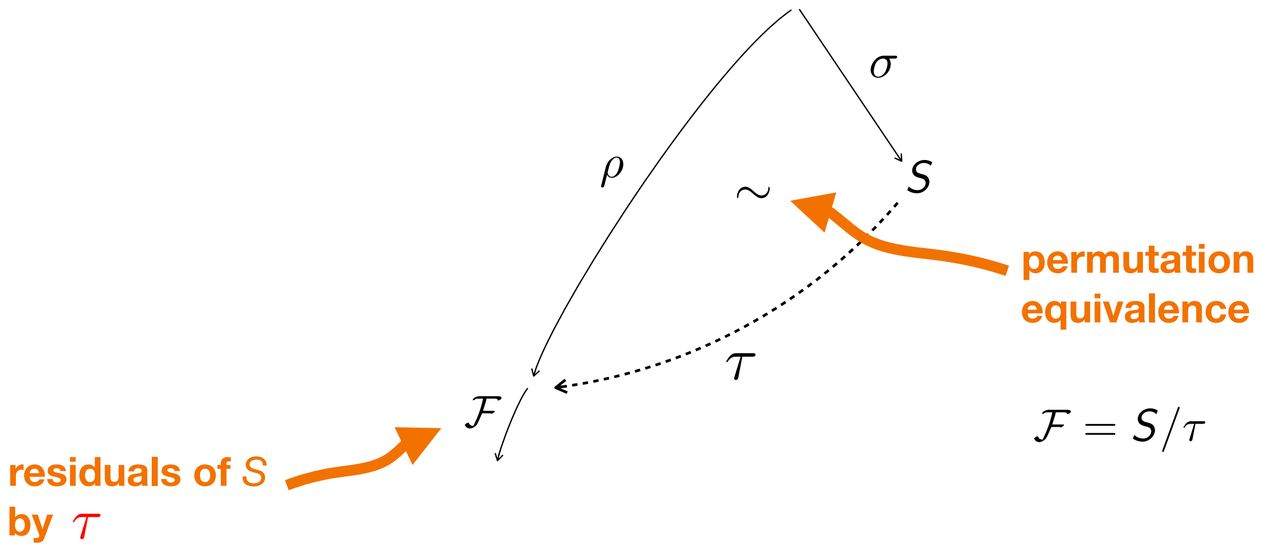
(x y)(x y)
or
(x x)(y y)
??

- sharing contexts ? with boxes and tags ??

- back to easier theory …

Duplication complete reductions

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

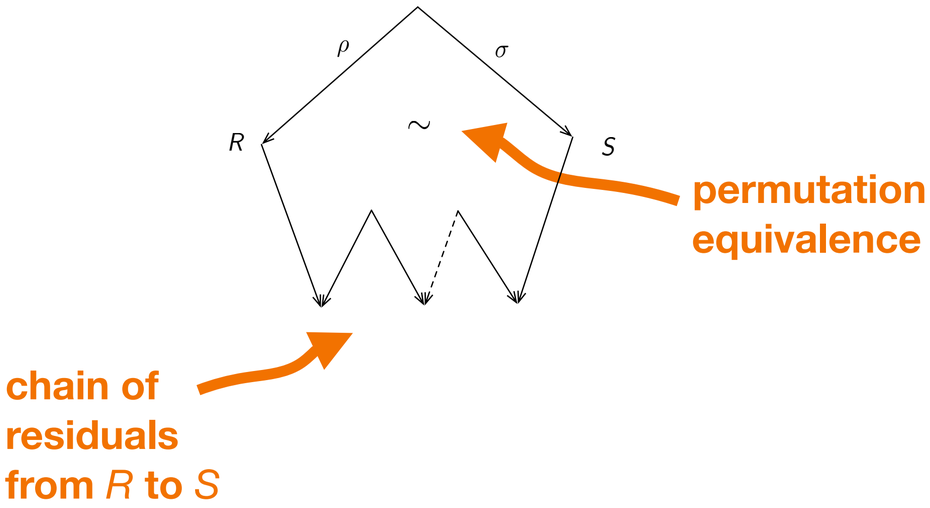


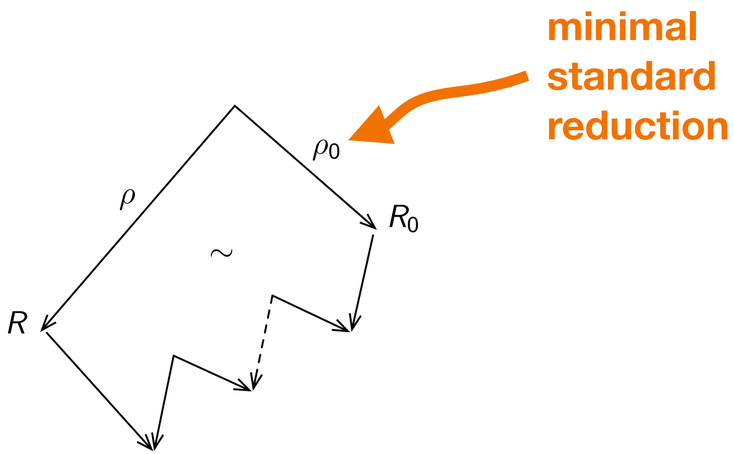
Goal [optimality thm] leftmost-outermost d-complete reductions are optimal in their number of parallel reduction steps

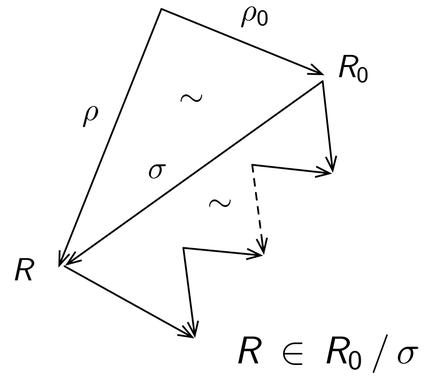
• but how to find $\langle \sigma, S \rangle$?

Redex families

- family of redexes is chain of residuals modulo permutation equivalence







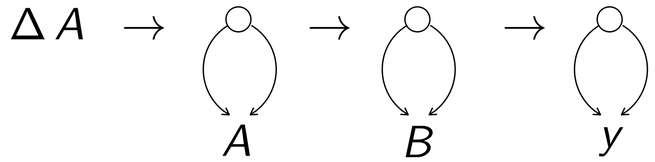
- in each family, canonical representative is unique with standard reduction of minimal length (*stability property*)

- below the canonical representative reduction, family is same as duplication of the canonical representative

Fact [for optimality thm] **d-complete** reductions are same as **family-complete** reductions

Duplication of redexes

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

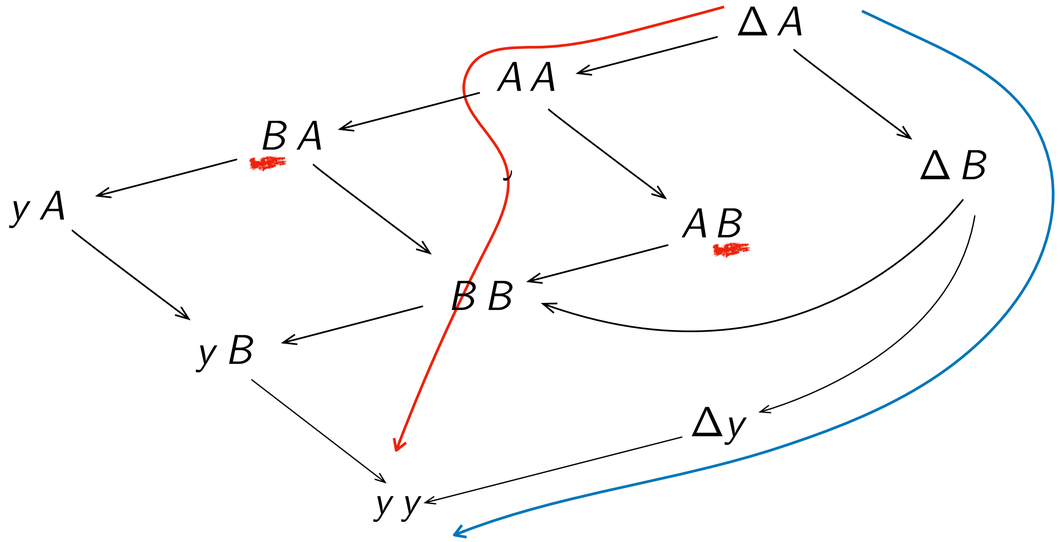


$\Delta = \lambda x.\, x\, x$

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

$I = \lambda x.\, x$

$A = F\, I \;\rightarrow\; I\, y = B \;\rightarrow\; y$



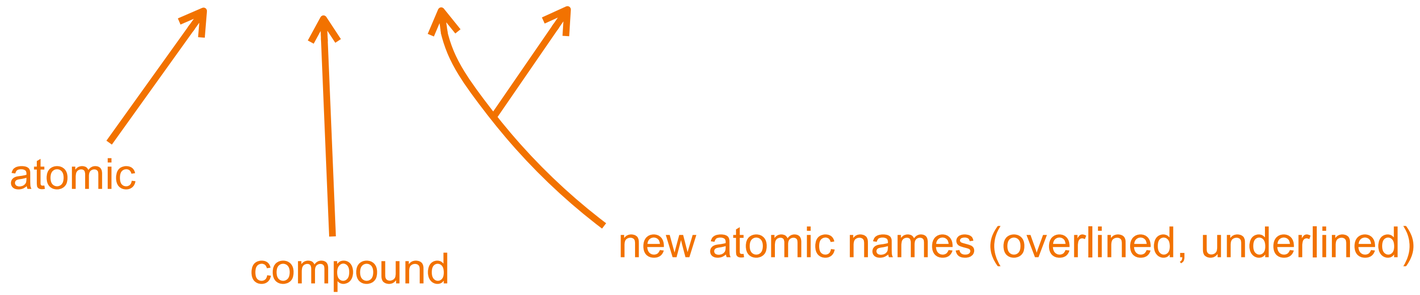
- optimal reductions has to deal with parallel reductions

labeled calculus

The labeled λ-calculus

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

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



- 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]

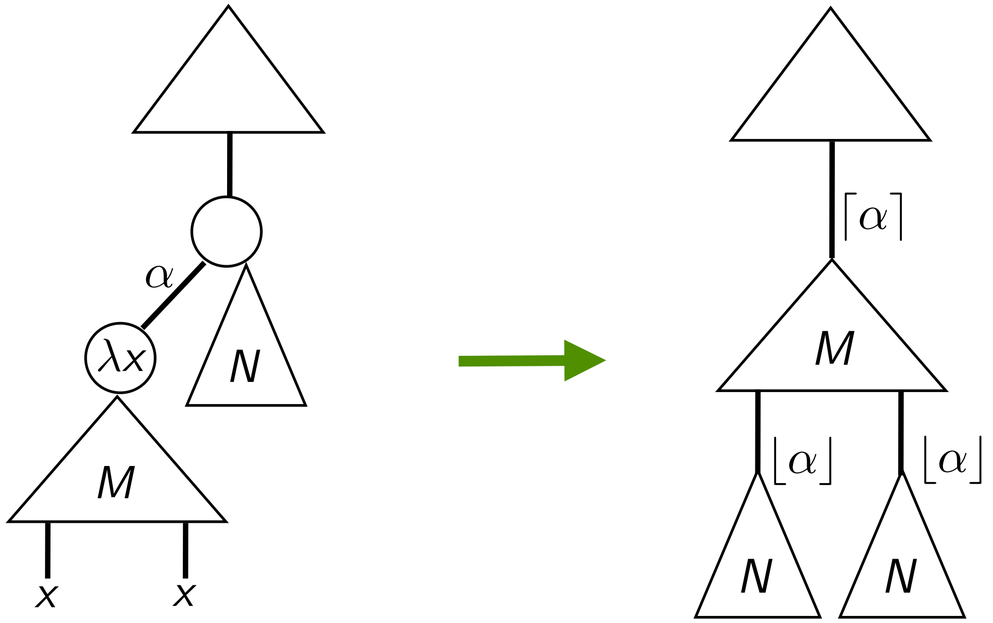
$$M, N, \ldots ::= x \mid MN \mid \lambda x.M \mid a : M$$

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

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

- correspondence with paths in initial term

The labeled λ-calculus



Theorem [optimality] leftmost-outermost **labeled-complete** reductions are optimal in their number of parallel reduction steps

Theorem [labeled family] redexes have same labeled names iff they are in same family

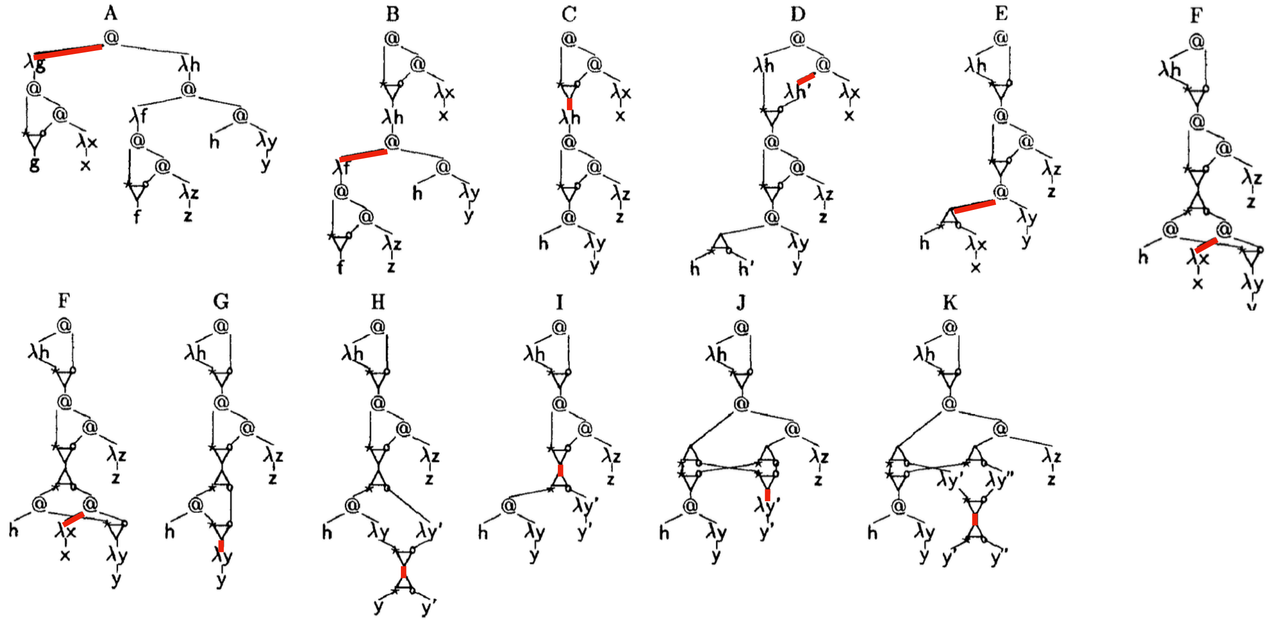
Lamping algorithm

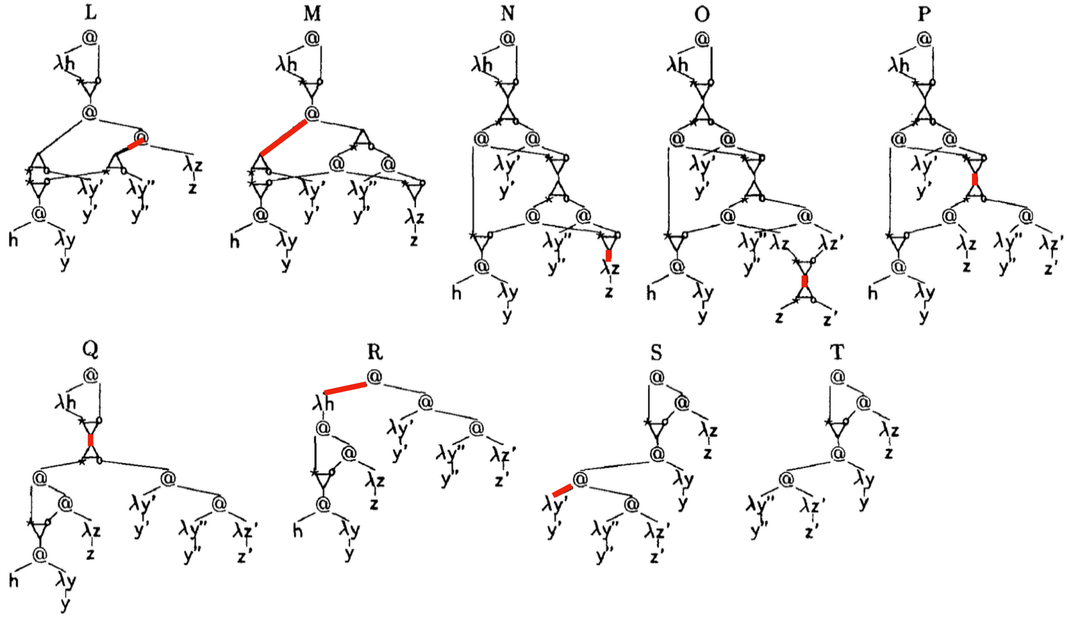
Lamping algorithm

- implements optimal reductions [Lamping 90]

$I = \lambda x.x = \lambda y.y = \lambda z.z$

$A = (\lambda g.\, g(g\, I))\, (\lambda h.\, (\lambda f.\, f(f\, I))\, (h\, I))$





C $\rightarrow$ D : unshare λh and $\lambda h'$

E $\rightarrow$ F : unshare @ by permuting with fan-out

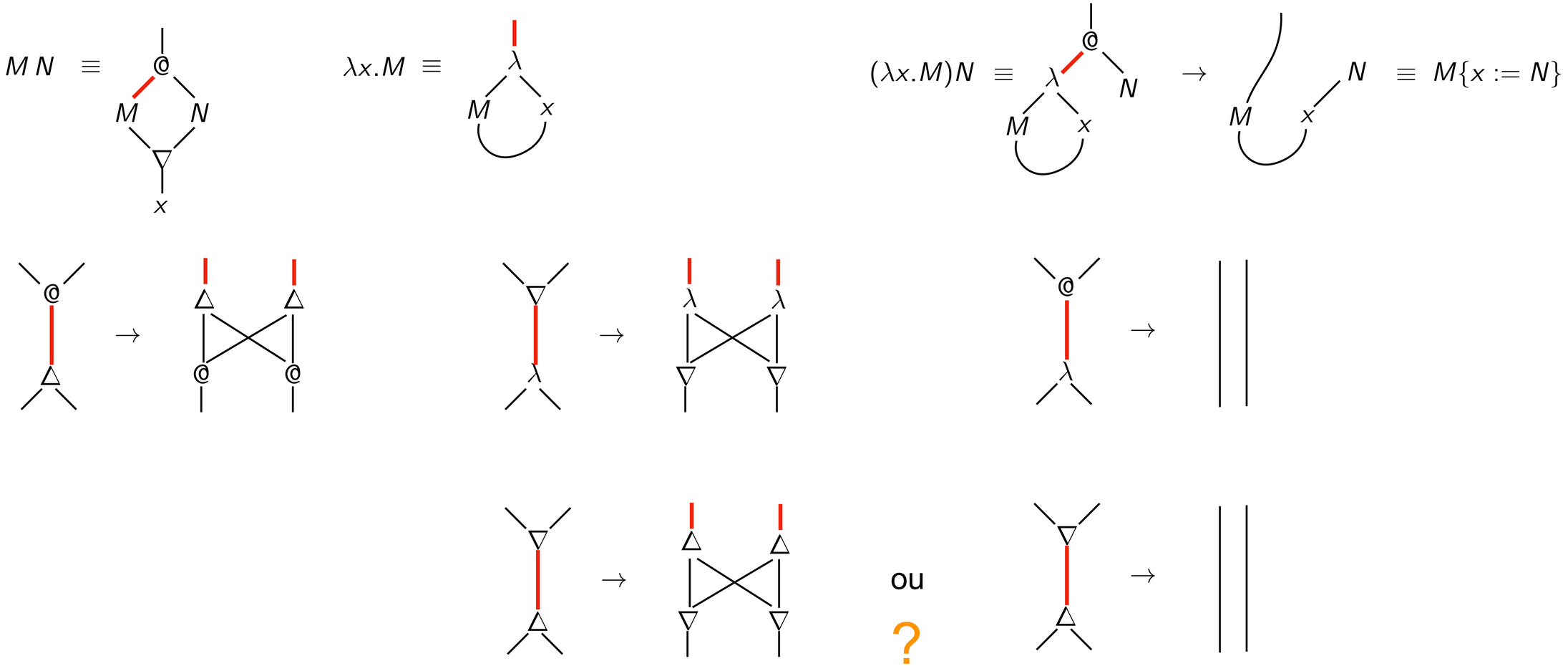
L $\rightarrow$ M $\rightarrow$ N: idem

I $\rightarrow$ J : permute "distinct" fan-in and fan-out

N $\rightarrow$ O: unshare λz and $\lambda z'$

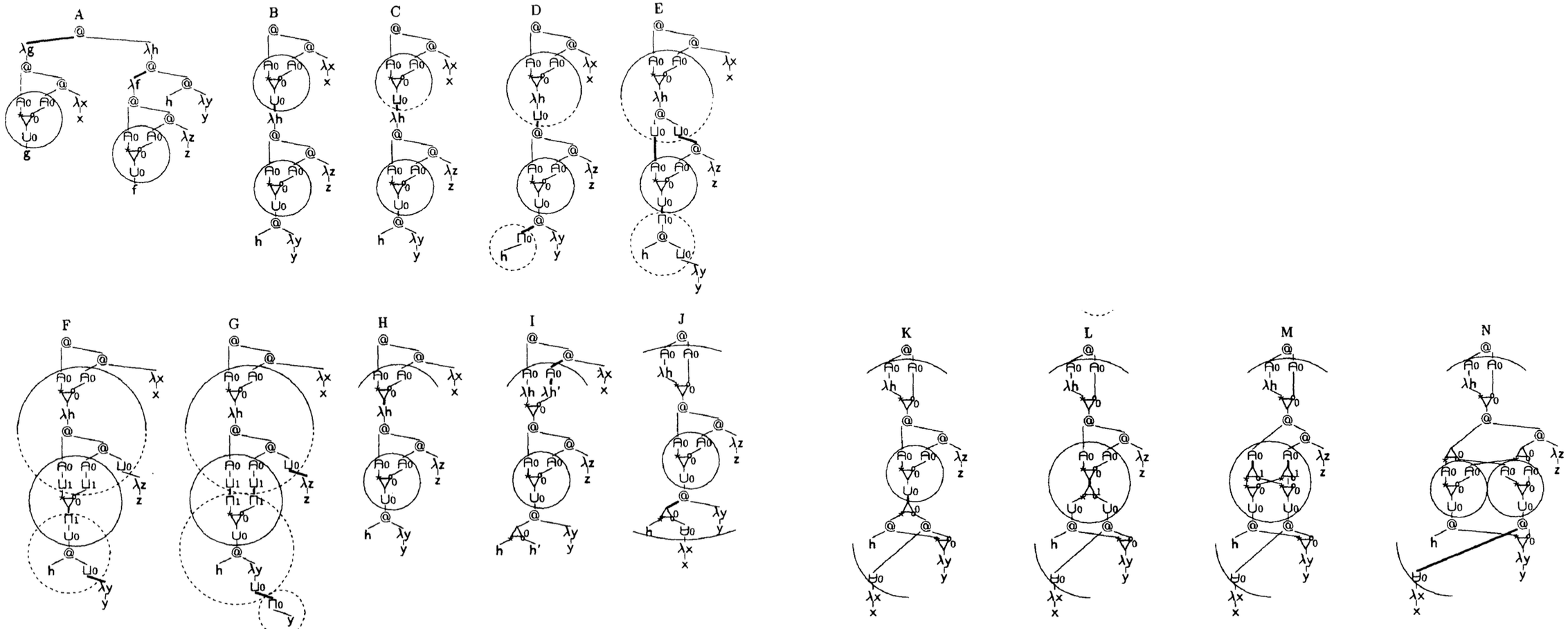
Lamping algorithm

- execution rules [Lamping 90] with interaction nets [Lafont 90]



Lamping algorithm

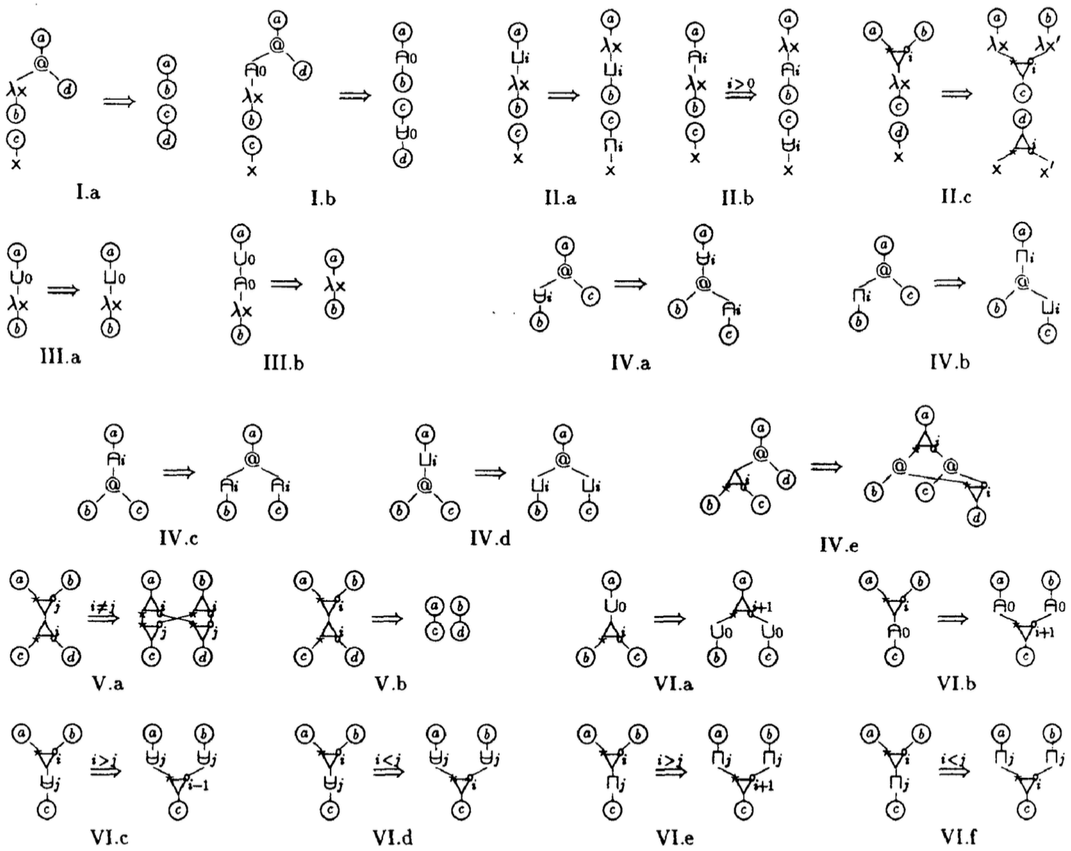
- implements optimal reductions [Lamping 90] with control operators '𝕮' and ']' for fanin's and fanout's enclosures



B $\rightarrow$ C : unconditional $\sqsupset$ becomes conditional $\supset$ allowing non-contiguous enclosures

Lamping algorithm

- execution rules [Lamping 90]



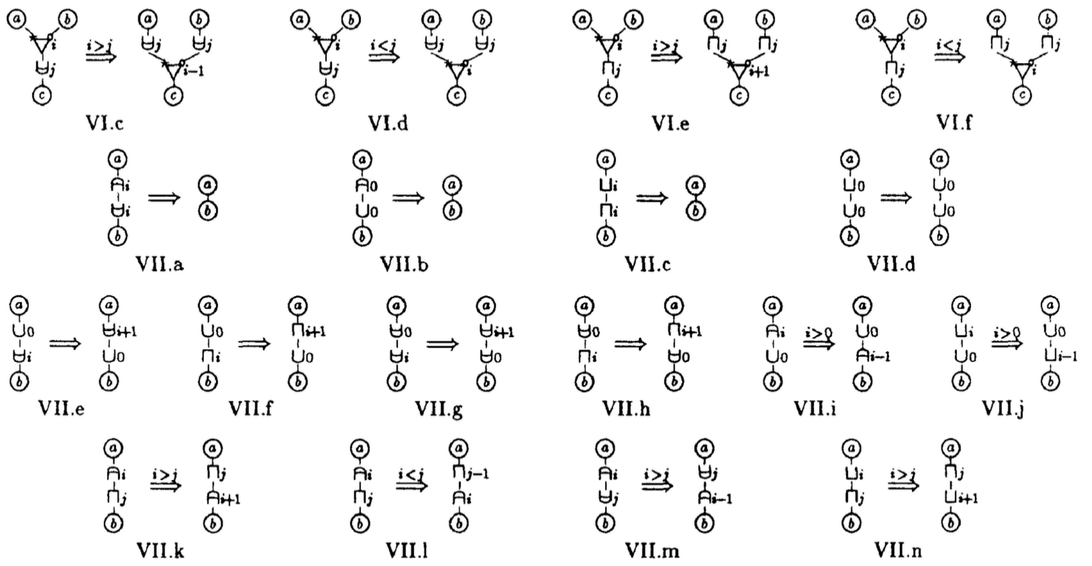
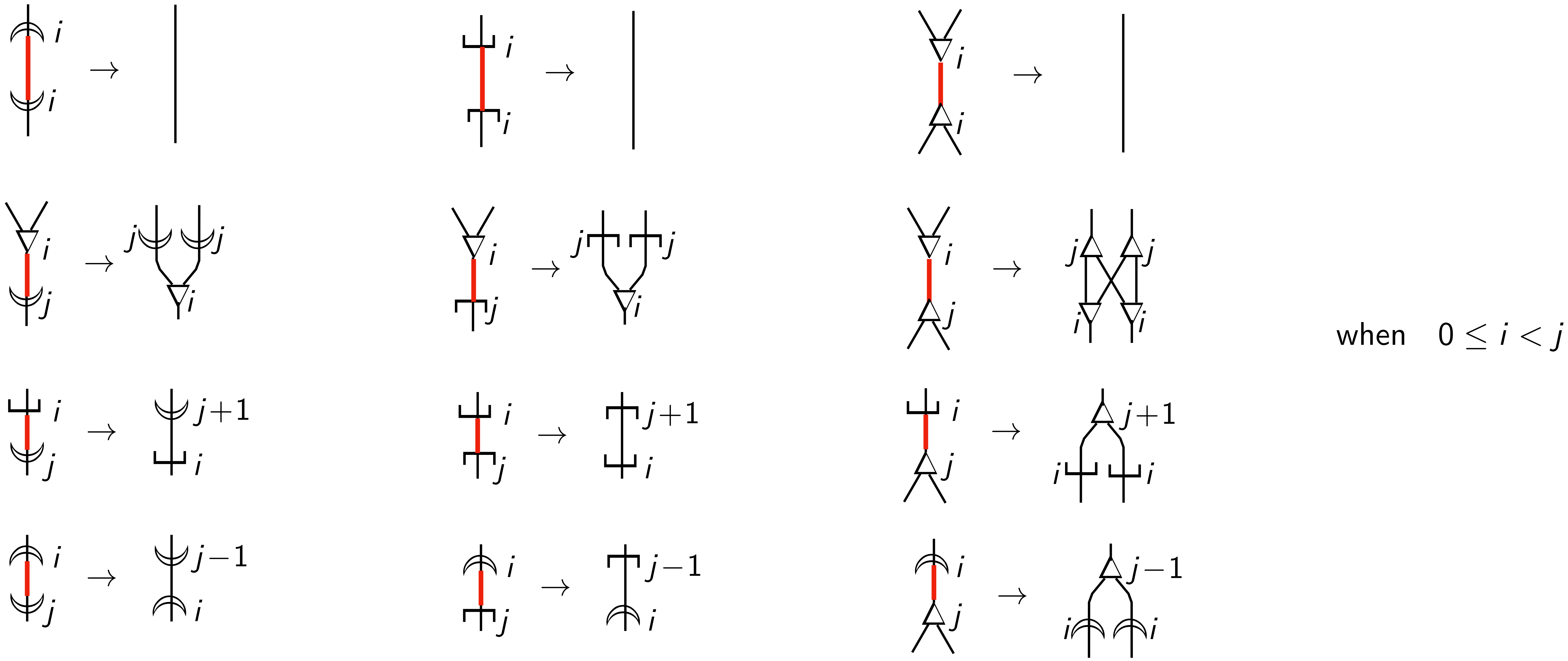


Figure 3: Rules

Gonthier algorithm

Gonthier algorithm

- fanin's ▽ and △ fanout's behave as @ and λ
- simplifying Lamping's enclosures delimited by opening croissants '☾' and closing square brackets ']'
- all operators are indexed (their depth)

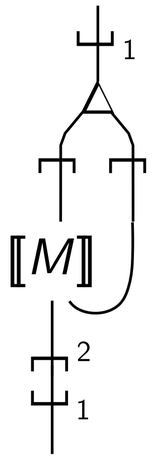


Gonthier algorithm

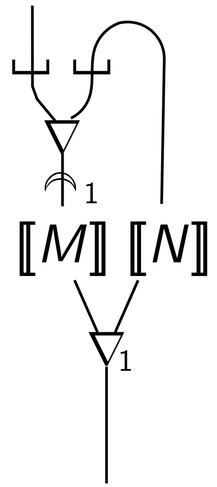
- operators interact with their principal ports (in red) as in linear logic

- coding of λ-expressions

$[\![\lambda x.M]\!] \equiv$



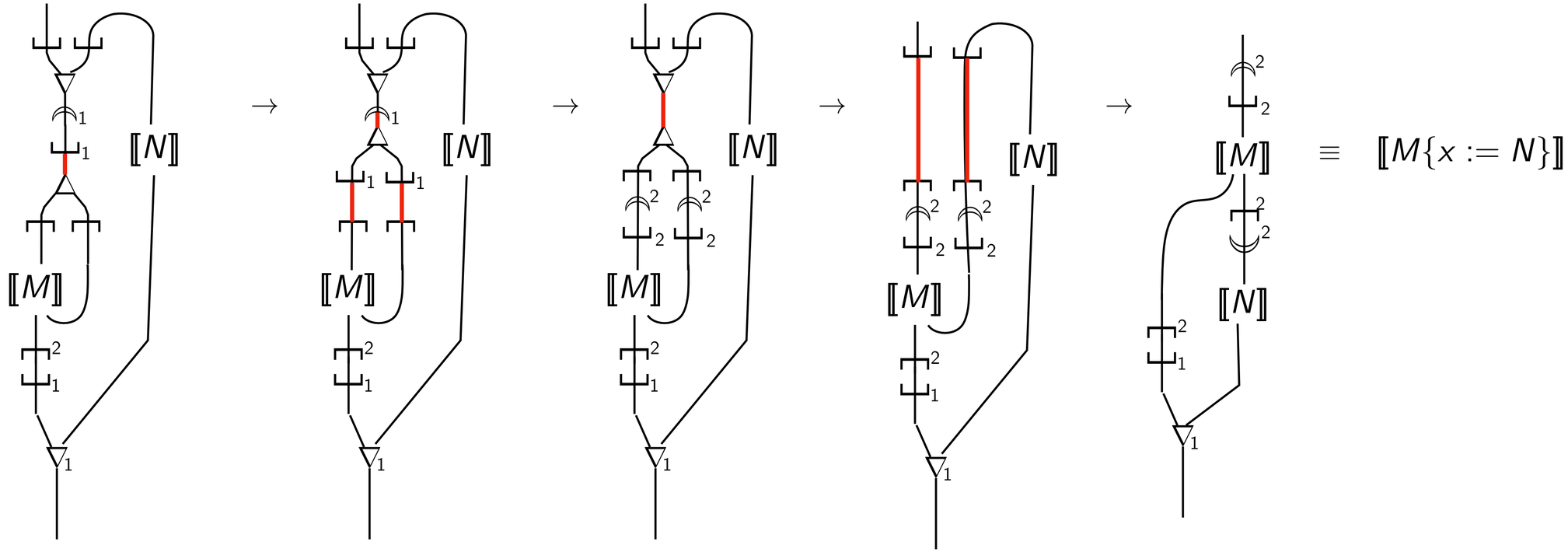
$[\![M\ N]\!] \equiv$



Gonthier algorithm

- beta-reduction step

$[\![(\lambda x.M)\ N]\!] \equiv$

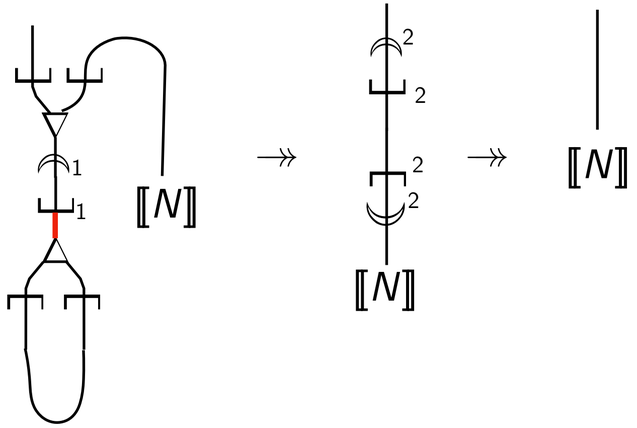


$\equiv \quad [\![M\{x := N\}]\!]$

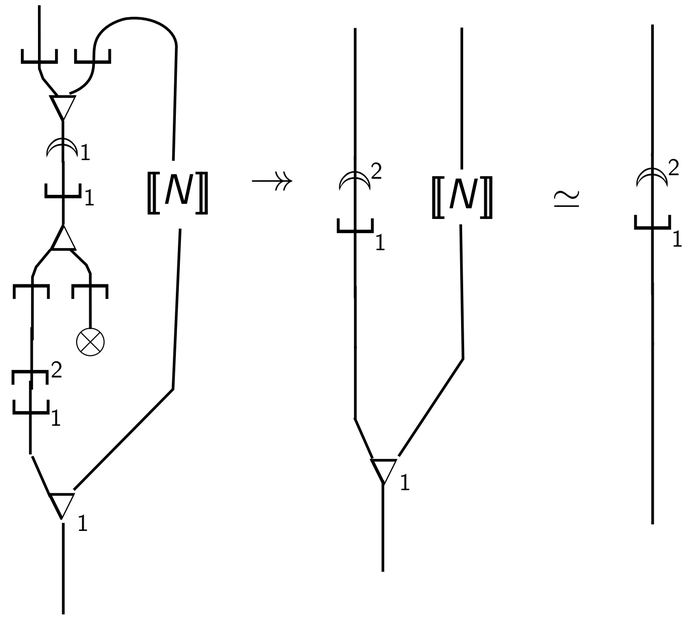
Gonthier algorithm

- examples

$[\![(\lambda x.x)\ N]\!] \equiv$

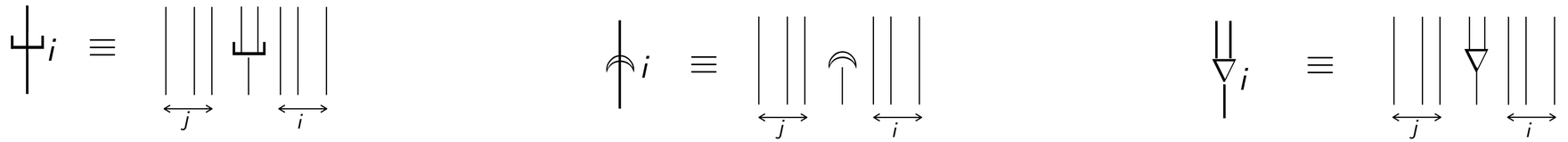


$[\![(\lambda x.y)\ N]\!] \equiv$

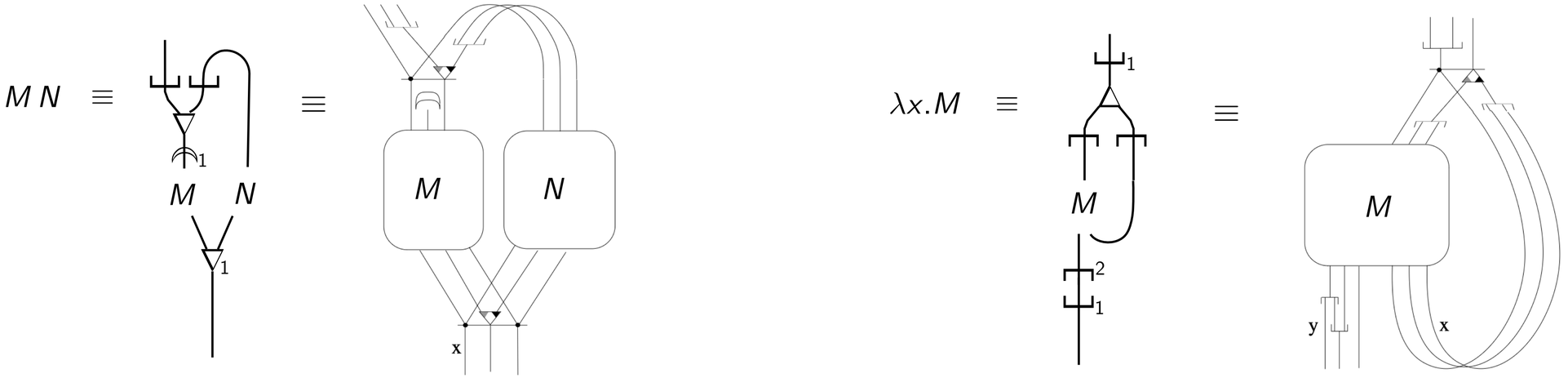


Gonthier algorithm

- operators can be decomposed with a (weird) bus notation

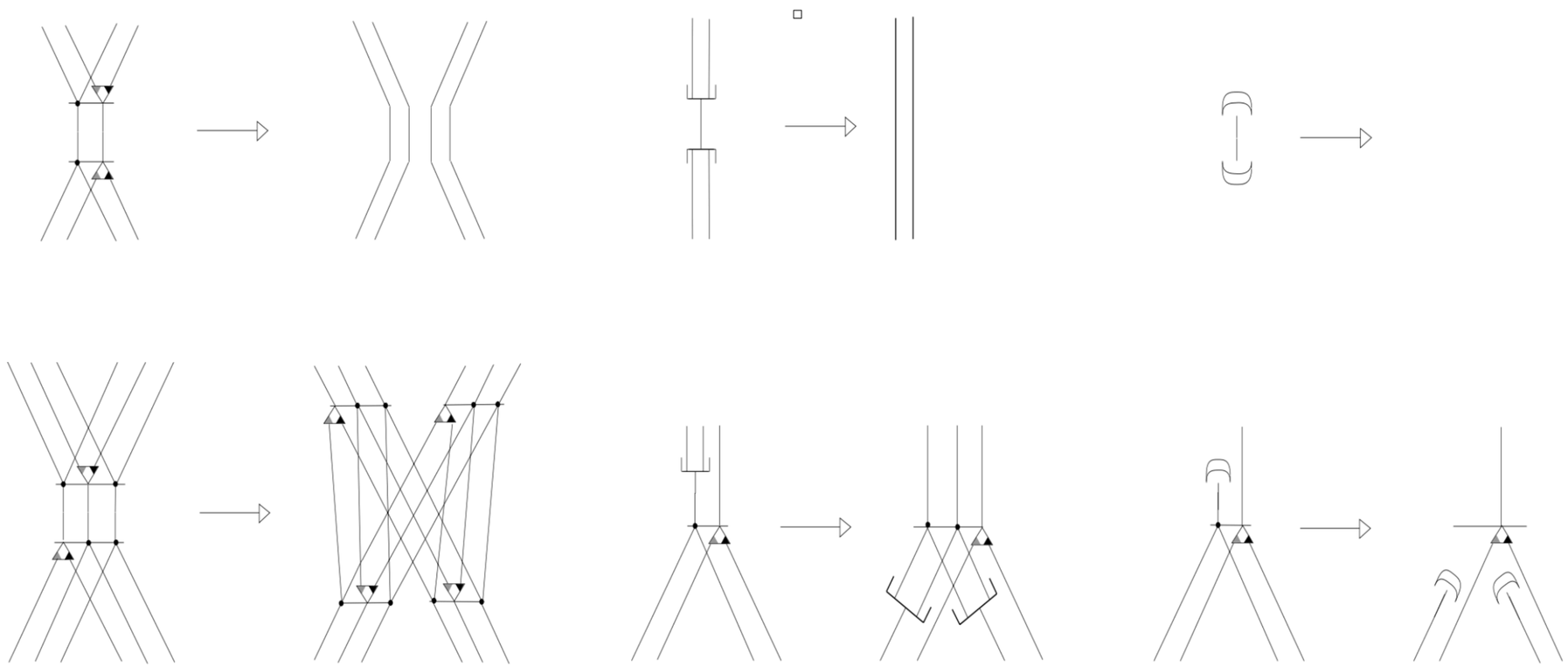


- coding of λ-expressions



Gonthier algorithm

- with the bus notation, interaction rules are ``simpler '' due to geometric properties



Gonthier algorithm

- **semantics** of the sharing nets

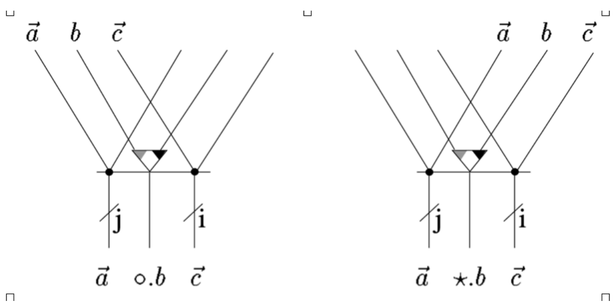
left wires (stack of contexts)

$\langle \dots \langle \langle a_1, a_2 \rangle, a_3 \rangle \dots a_n \rangle$

right wires (list of associations context - operators

$\otimes_1 \cdot \langle a_2, \otimes_2 \rangle \cdot \langle a_3, \otimes_3 \rangle \dots \cdot \langle a_{n-1}, \otimes_{n-1} \rangle \cdot a_n$

- transformation of the sharing contexts



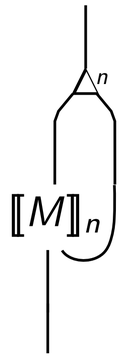
read-back procedure

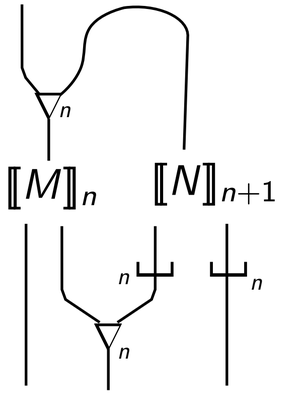
the BOHM machine

Alternatives coding

- `Asperti, Danos, Laneve, Regnier` investigated relations with linear logic
- `Asperti et al` developped the Bologna Higher Order Machine to implement Lamping style machine
- they use the (more natural) ``call-by-need'' coding $D = (!D) \multimap D$ instead of $D = !(D \multimap D)$

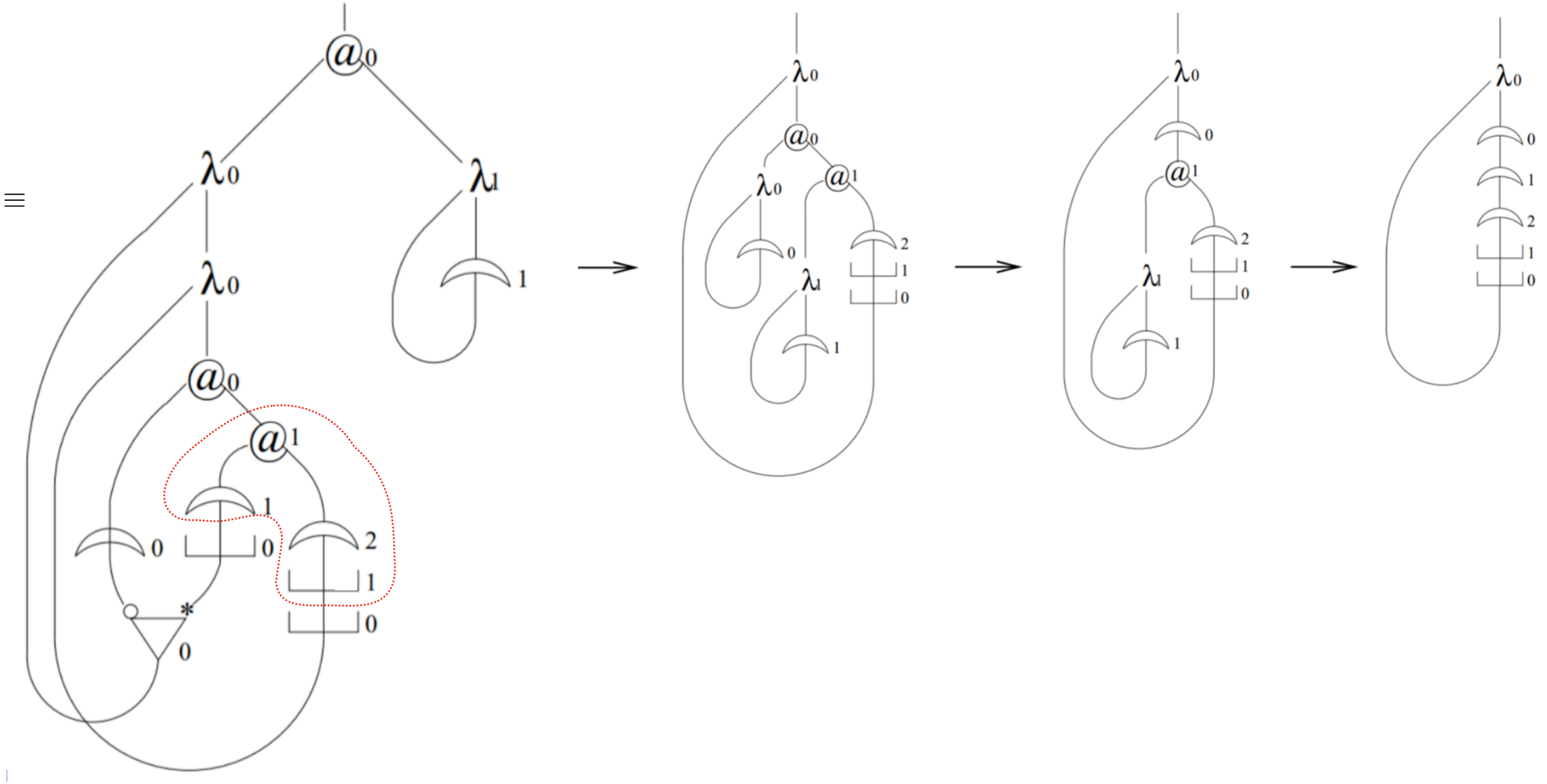
$[\![x]\!]_n \equiv$ 

$[\![\lambda x.M]\!]_n \equiv$ 

$[\![M\ N]\!]_n \equiv$ 

Example

$[\![(\lambda f.\lambda x.f\,(f\,x))\;I]\!]_0 \quad \equiv$



Efficiency

- Statman[79] normalizing simply-typed λ-terms is not elementary recursive in size of term
- Asperti-Mairson[98] the number of simply-typed λ-terms parallel-steps is linear in size of term
- Asperti-Coppola-Martini[04] duplications are not Kalmar elementary recursive
- comments in interesting talk by Laretto[21] :
  - **For pure λ-calculus terms**, BOHM greatly *outperforms* the competition (SML, Caml Light, Yale Haskell), giving polynomial reductions for many exponential cases
  - **For real world functional programs**, BOHM is *one order of magnitude slower* than call-by-value languages (SML, Caml Light), and sometimes only slightly worse than Yale Haskell
- BOHM 1.0 and 1.1 is available on web at https://github.com/asperti/BOHM1.1

[BOHM language is PCF-style extension of untyped -calculus]

Conclusion

- this work established a **deeper** correspondence between λ-calculus and linear logic
- brave student should program the **GOHM** analogous to BOHM
- it works for linear logic without boxes
- and also in any calculi with **bound variables** (interaction systems)
- problem with **no difference** between typed and untyped calculi
- usual functional languages implement **weak reduction** where sharing is easier
- much of this work is due to the clairvoyance of Georges Gonthier

merci

Georges !