Sharing in the weak lambda-calculus (2)

Jean-Jacques Lévy
INRIA

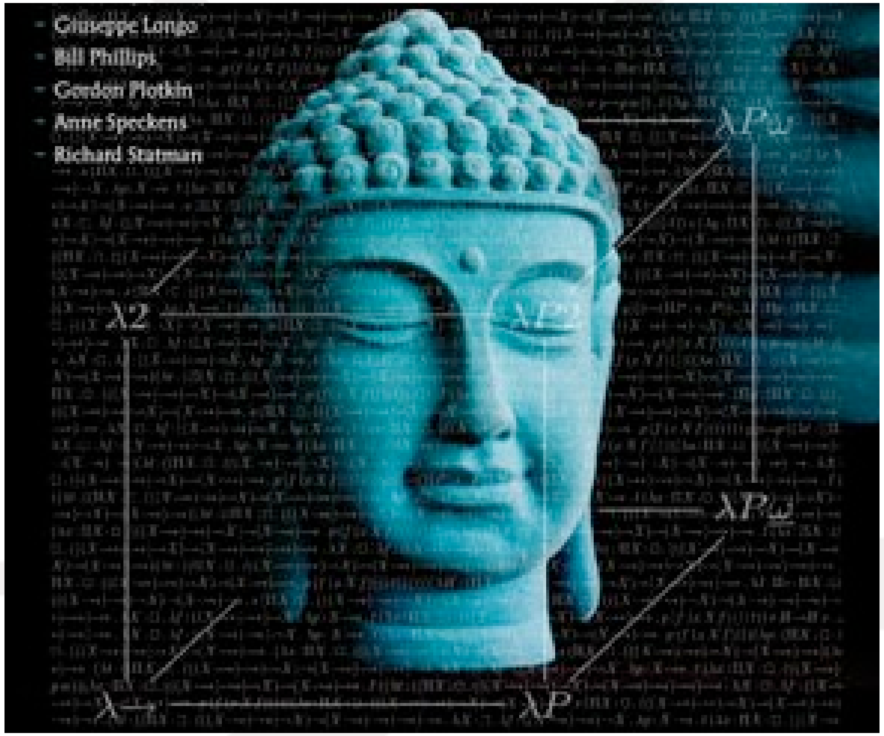
Joint work with Tomasz Blanc and Luc Maranget

CENTRE DE RECHERCHE COMMUN
INRIA MICROSOFT RESEARCH

Happy birthday Henk !

Happy birthday Henk !

Happy birthday Henk !



CENTRE DE RECHERCHE COMMUN

INRIA MICROSOFT RESEARCH

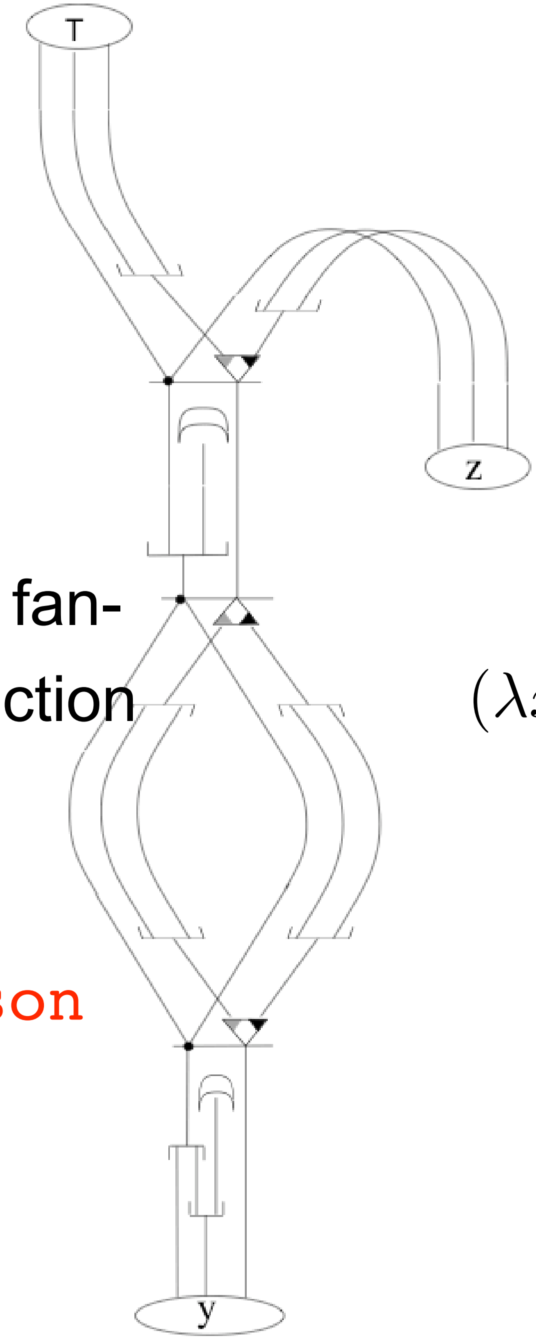
HISTORY

Sharing in the lambda-calculus

- goal:
  - **efficient implementations** of functional languages
  - by “functional languages”, we mean here **logical systems** (Coq, Isabelle, etc)
  - although real functional languages use more environment machines
  - but it could be useful for **partial evaluation**

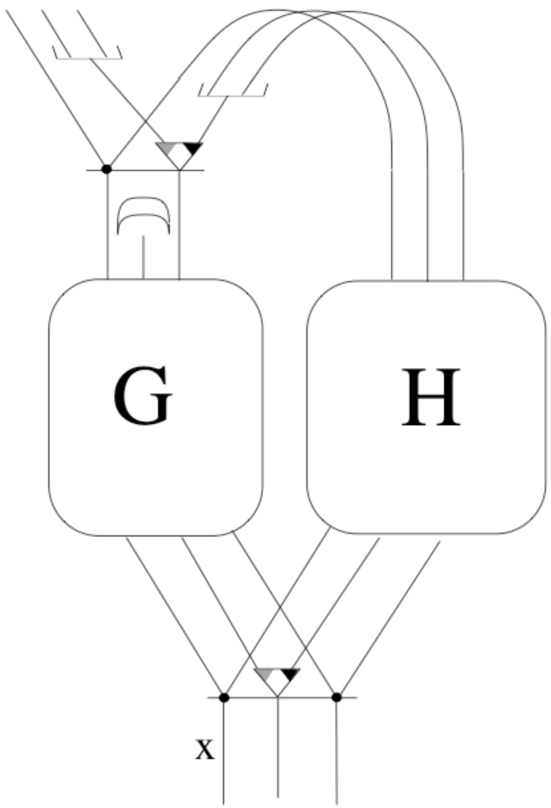
Sharing in the lambda-calculus

- Lamping's algorithm [91]:
  - **optimal** in total number of beta-reductions
  - sharing **contexts**
  - complex treatment of fan-in and fan-out nodes (geometry of interaction [Gonthier 92])
  - **inefficient** in practice (not elementary recursive [Mairson 96])

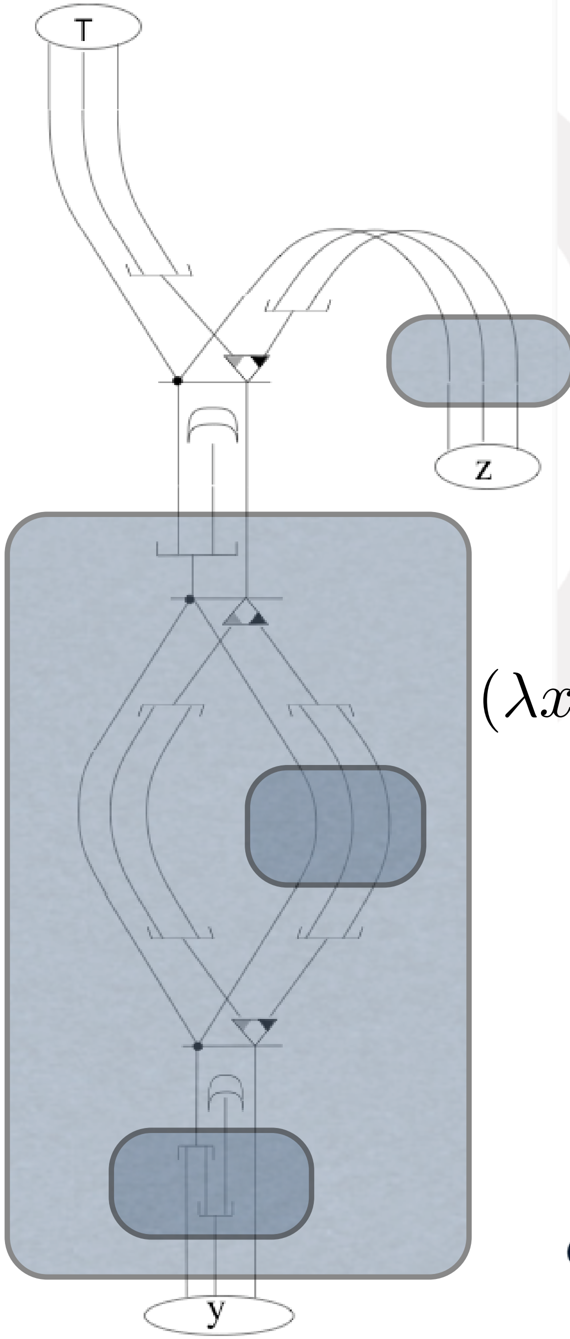


$(\lambda x.yx)z$

Sharing in the lambda-calculus



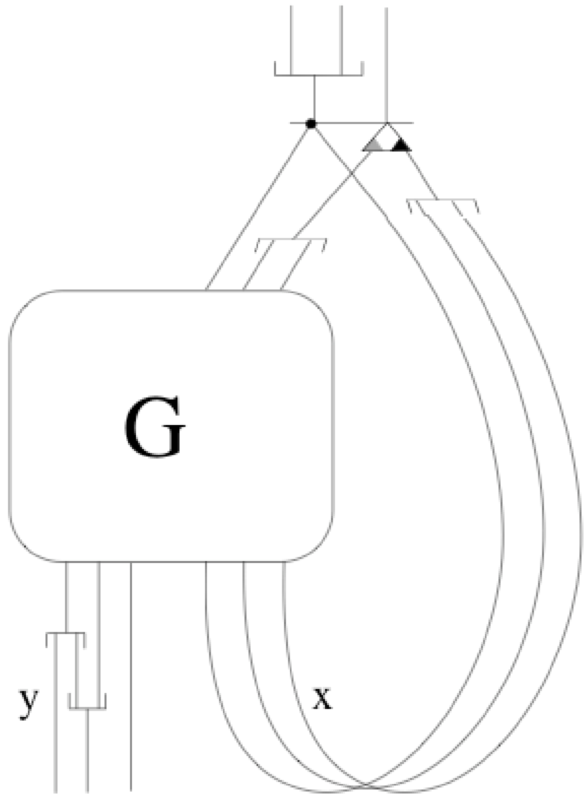
application



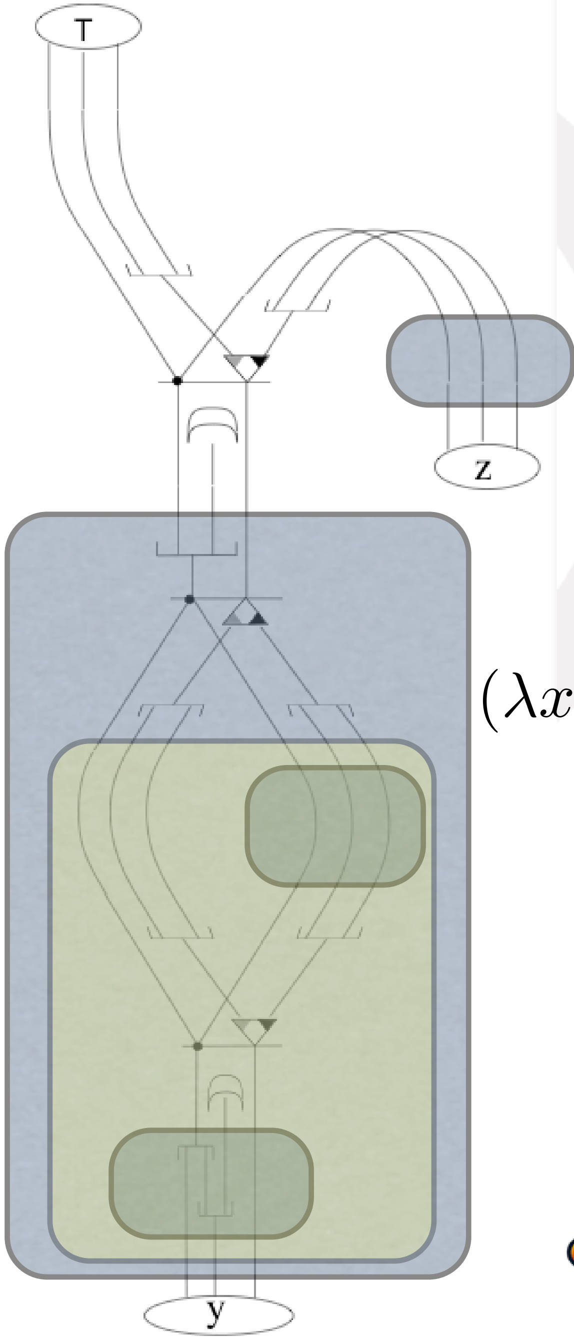
$(\lambda x.yx)z$

INRIA
MICROSOFT RESEARCH

Sharing in the lambda-calculus

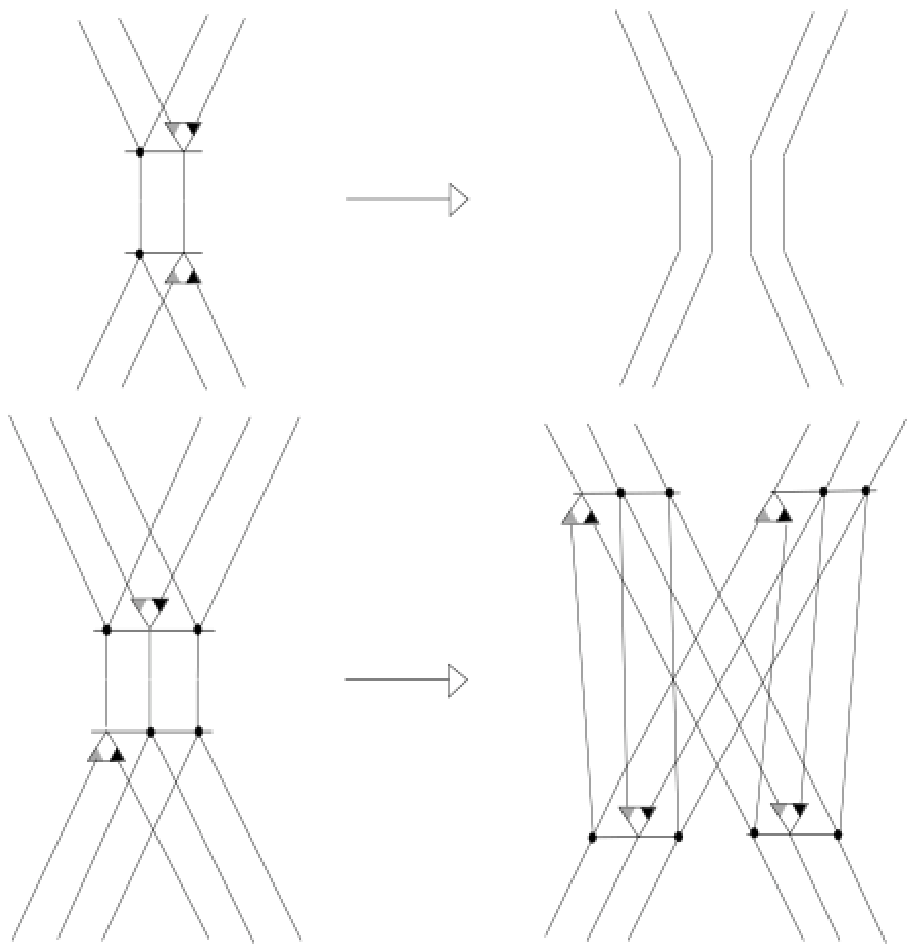


abstraction

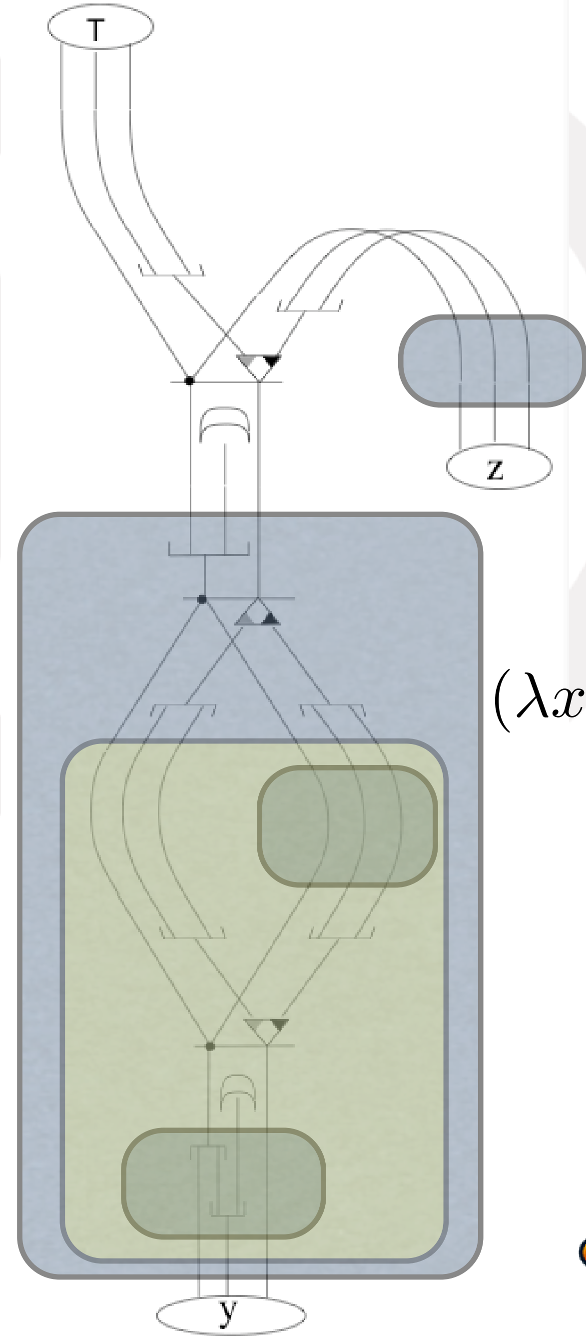


$(\lambda x.yx)z$

Sharing in the lambda-calculus



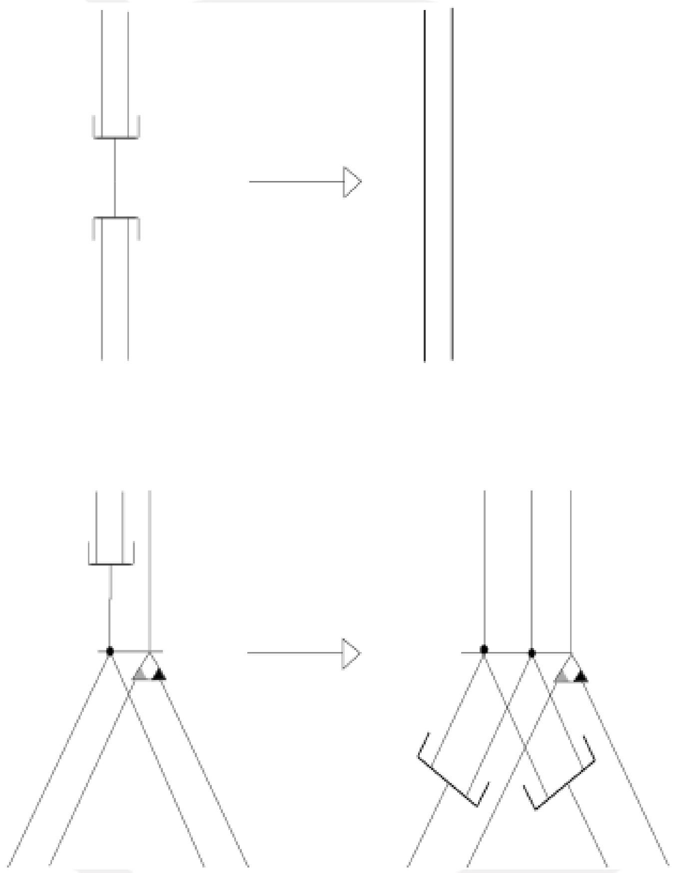
fan rules



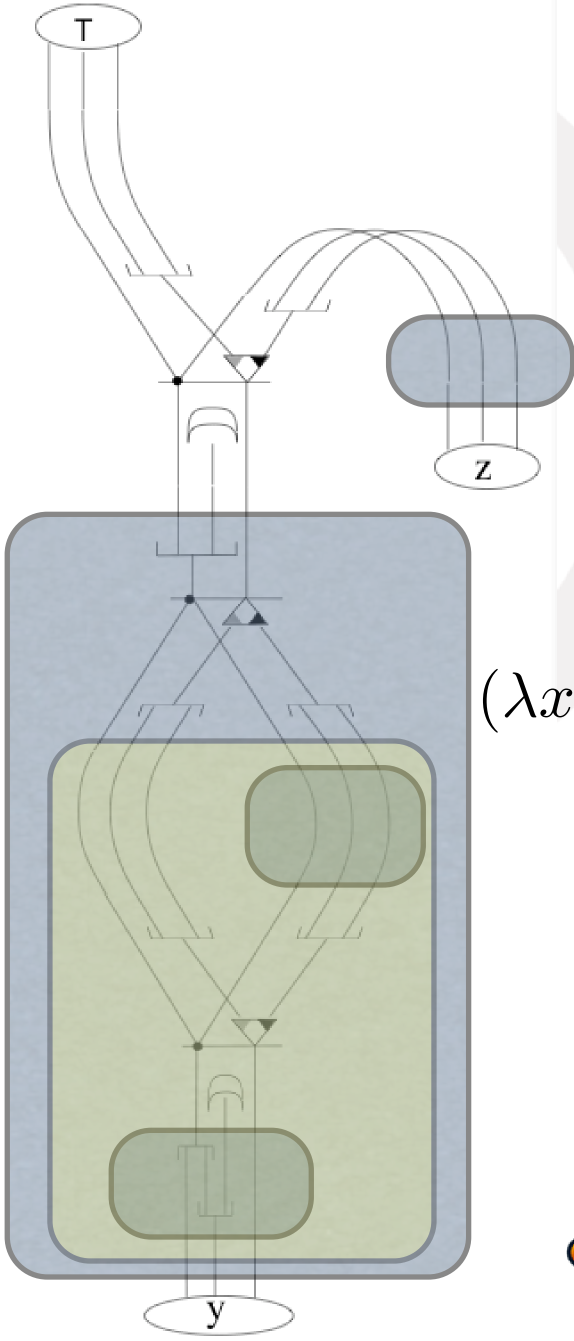
$(\lambda x.yx)z$

INRIA
MICROSOFT RESEARCH

Sharing in the lambda-calculus

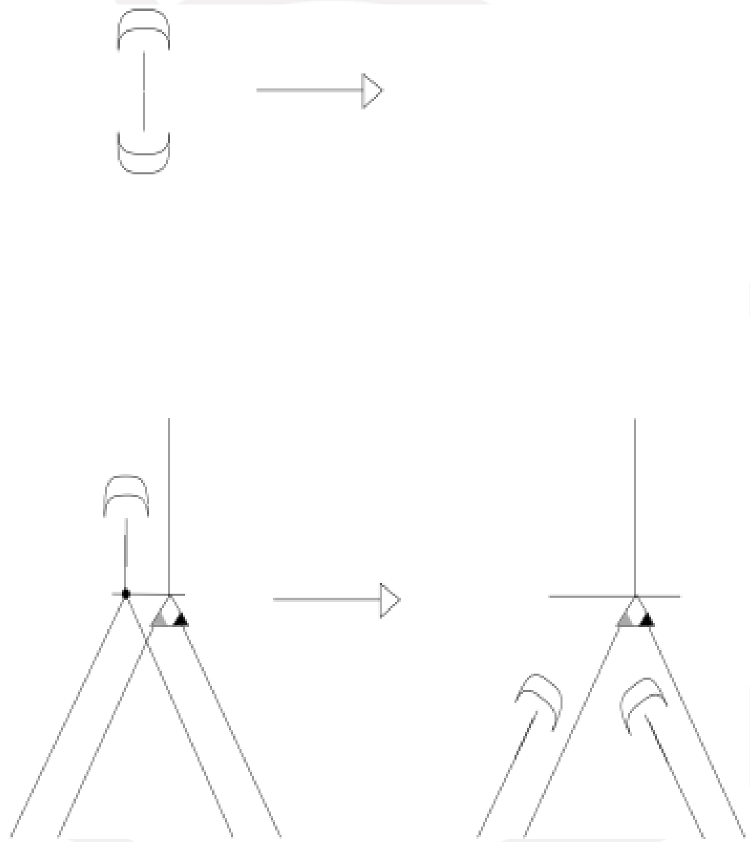


bracket rules

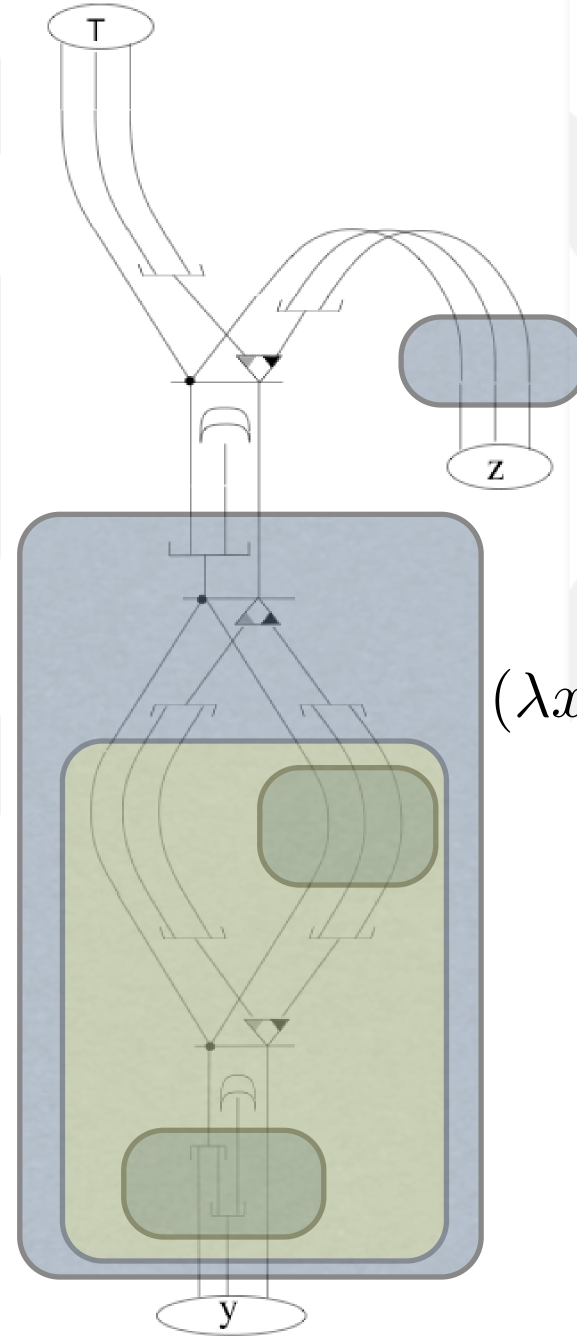


$(\lambda x.yx)z$

Sharing in the lambda-calculus



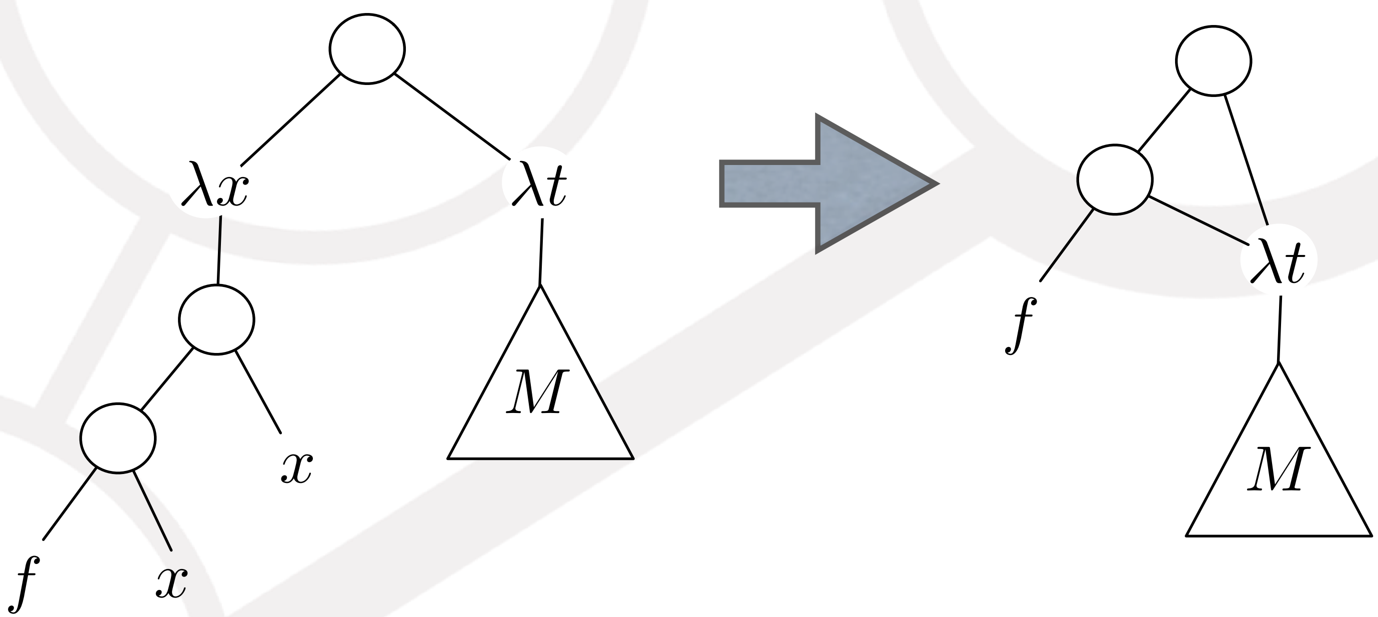
croissant rules



$(\lambda x.yx)z$

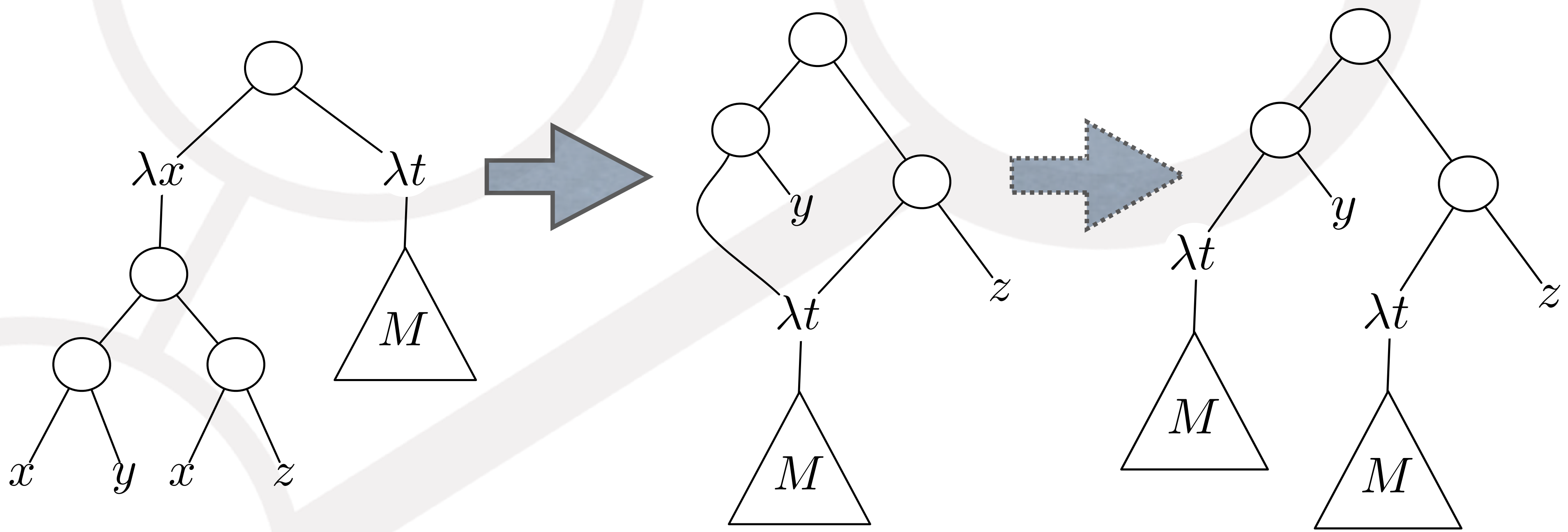
Wadsworth's algorithm

- sharing subterms [Wadsworth 72]:
  - arguments of beta-redexes are shared
  - easy to implement with **dags** (directed acyclic graphs)



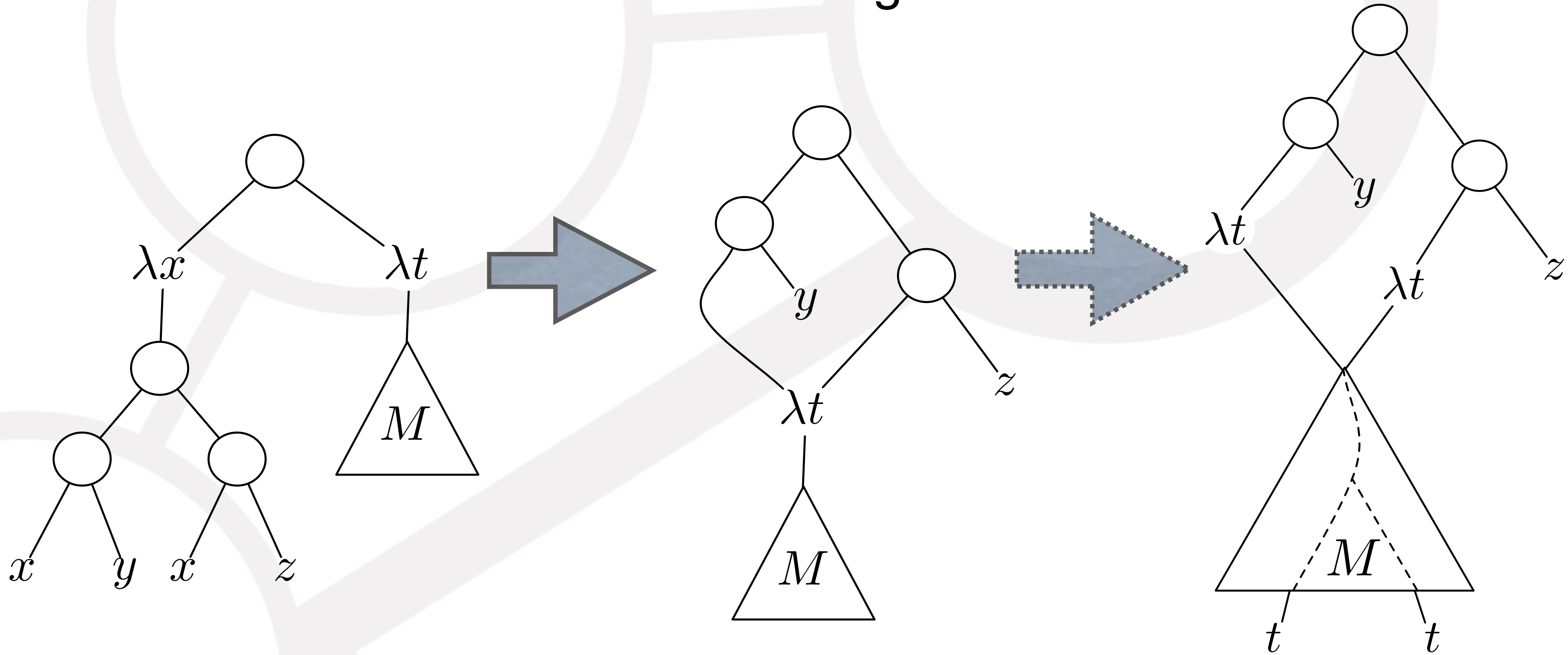
Wadsworth's algorithm

- Algorithm 1:
  - need **duplication steps** (abstractions on left of beta-redexes with reference counter greater than 1)
  - **not** optimal in total number of beta-reductions



Wadsworth's algorithm

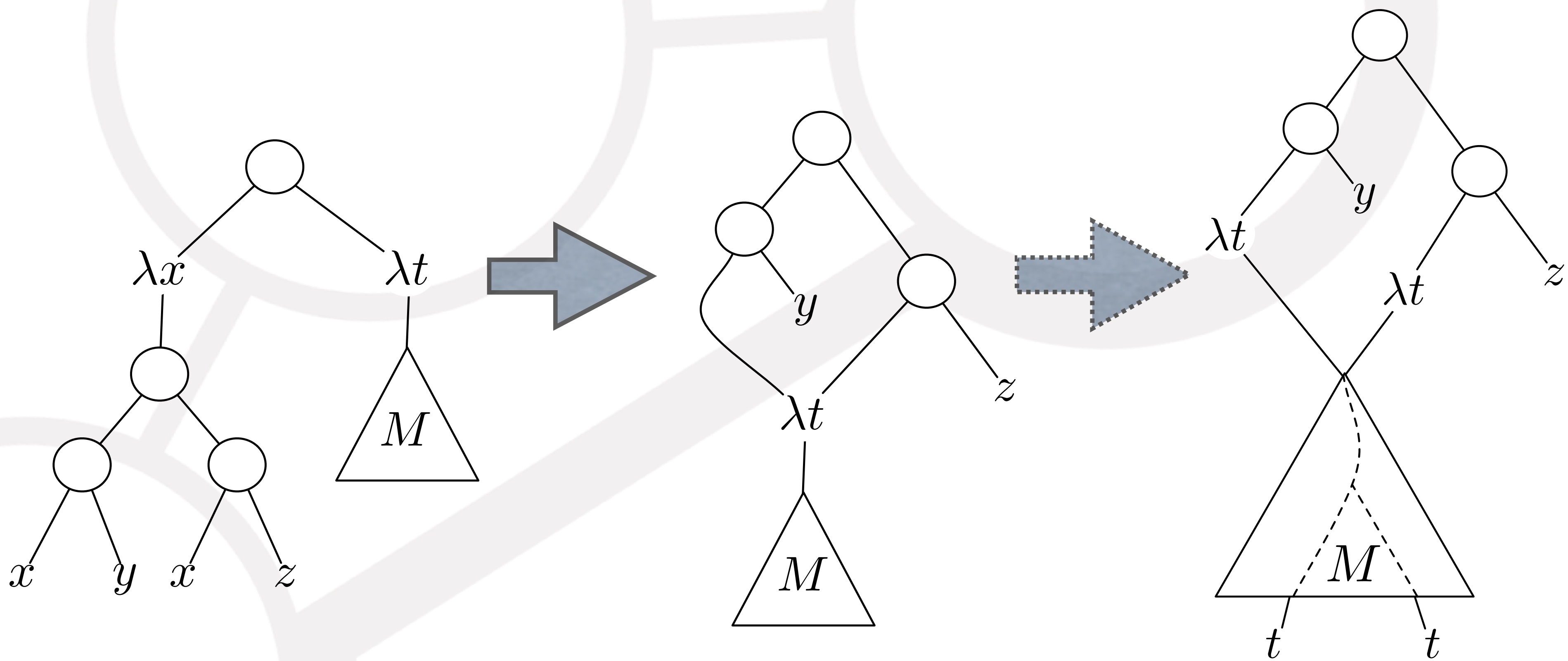
- Algorithm 2:
  - only duplicate **nodes on paths to the bound variable** of abstractions on left of beta-redexes
  - and share subterms not containing the bound variable



CENTRE DE RECHERCHE COMMUN
INRIA MICROSOFT RESEARCH

Wadsworth's algorithm

- Algorithm 2 [Shivers-Wand 04]:
  - bottom-up traversal of abstraction $\lambda t.M$ to find nodes and paths to the bound variable t

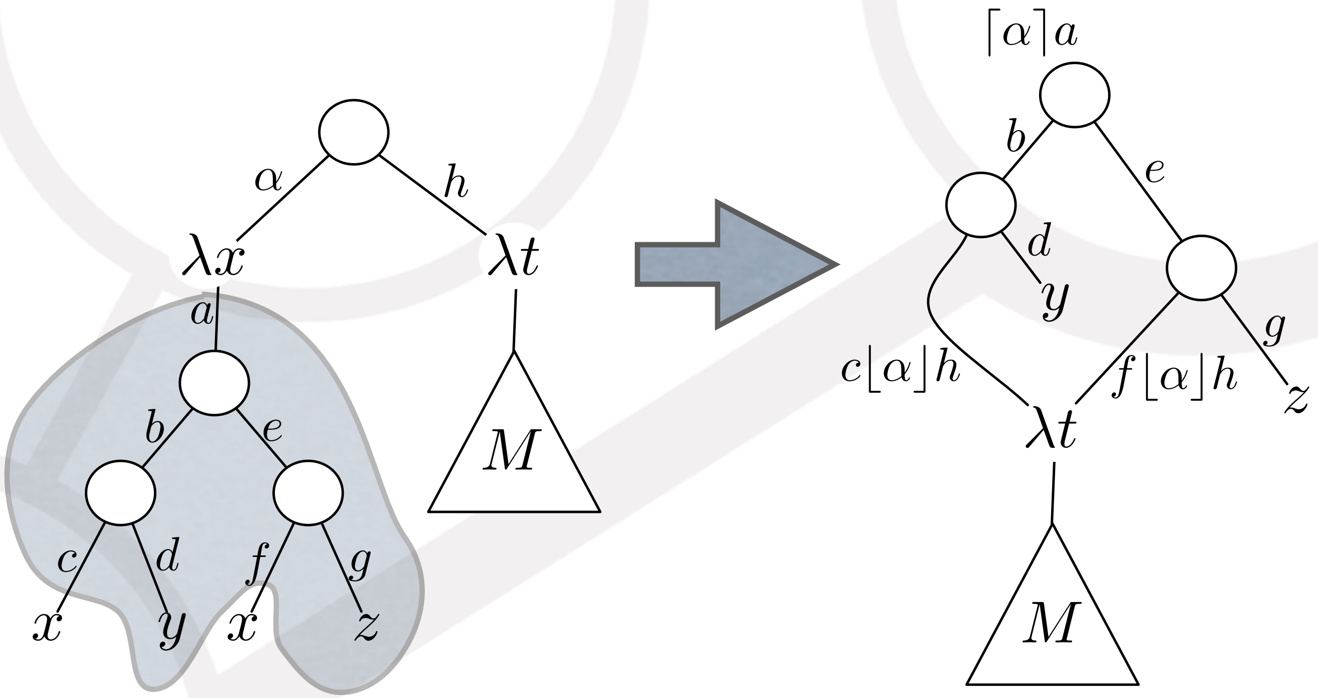


CENTRE DE RECHERCHE COMMUN INRIA MICROSOFT RESEARCH

LABELS

Strong labeled lambda-calculus

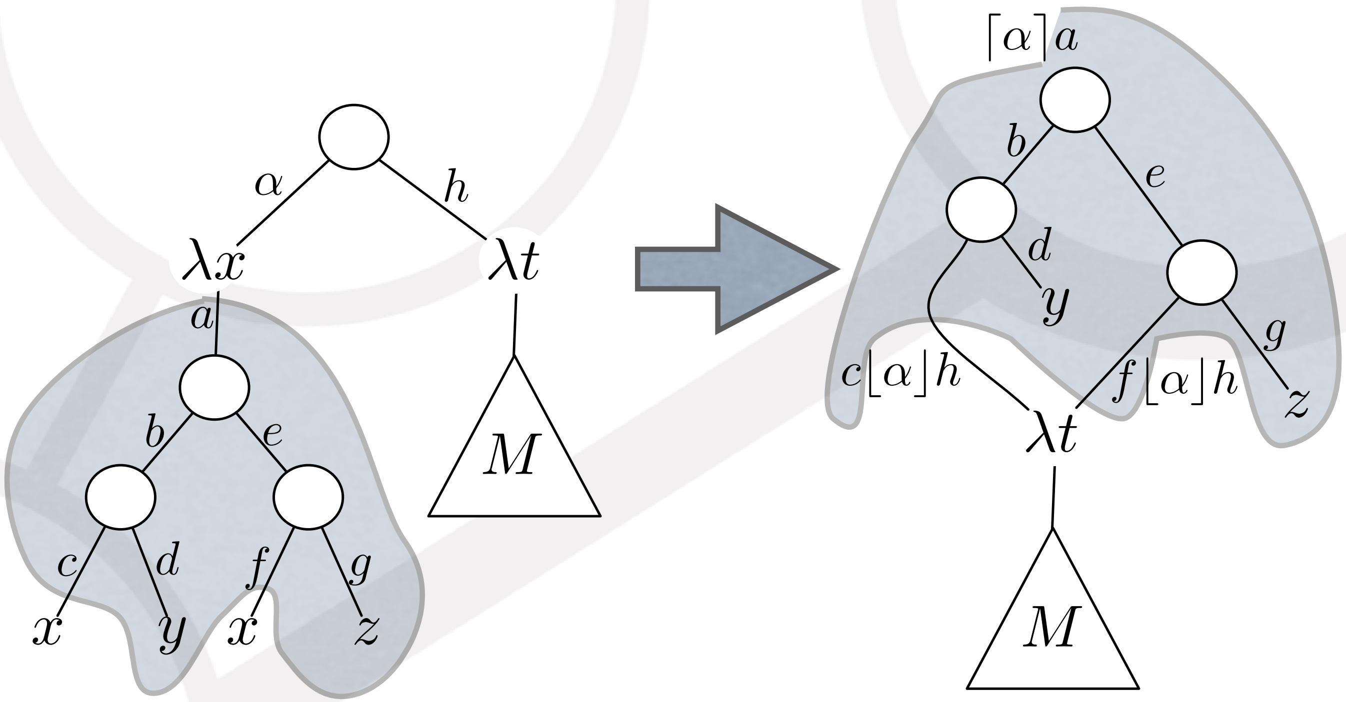
- catch **history** of creations of redexes
- names (labels) of redexes are structured
- **confluent** calculus



CENTRE DE RECHERCHE COMMUN
INRIA MICROSOFT RESEARCH

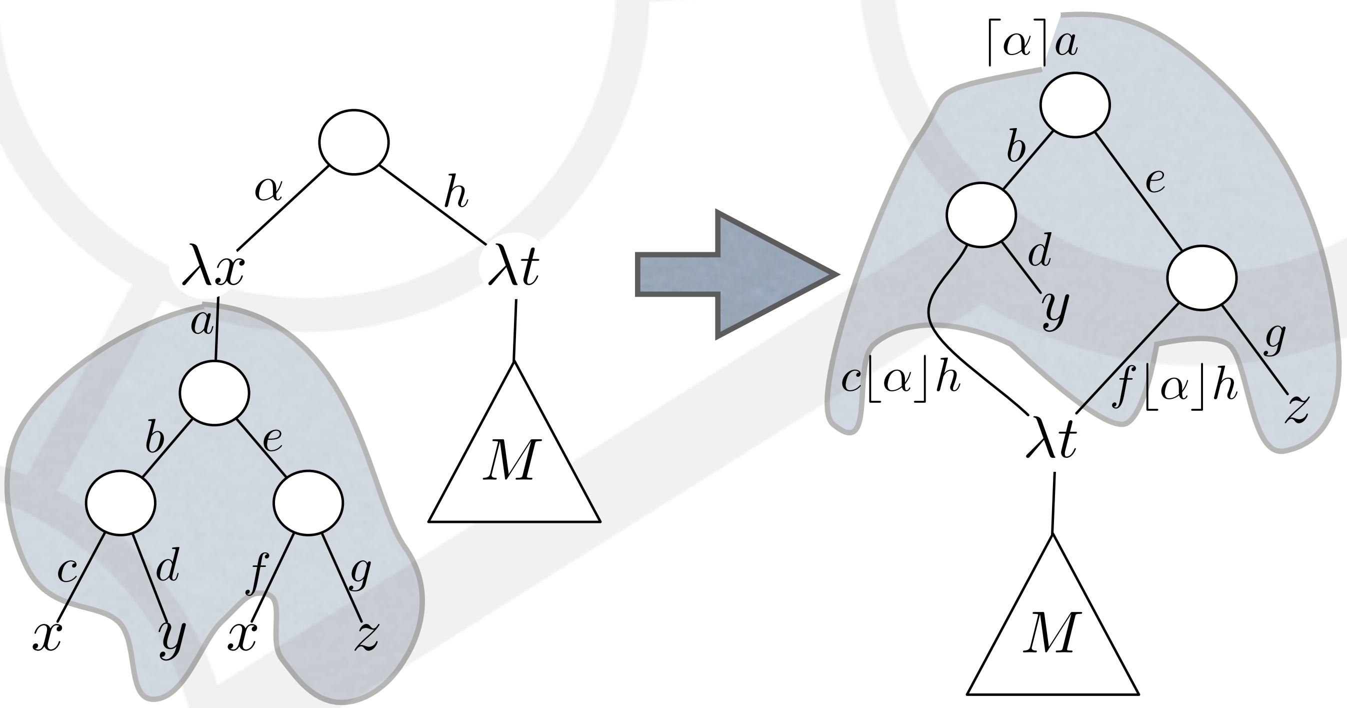
Strong labeled lambda-calculus

- catch **history** of creations of redexes
- names (labels) of redexes are structured
- **confluent** calculus



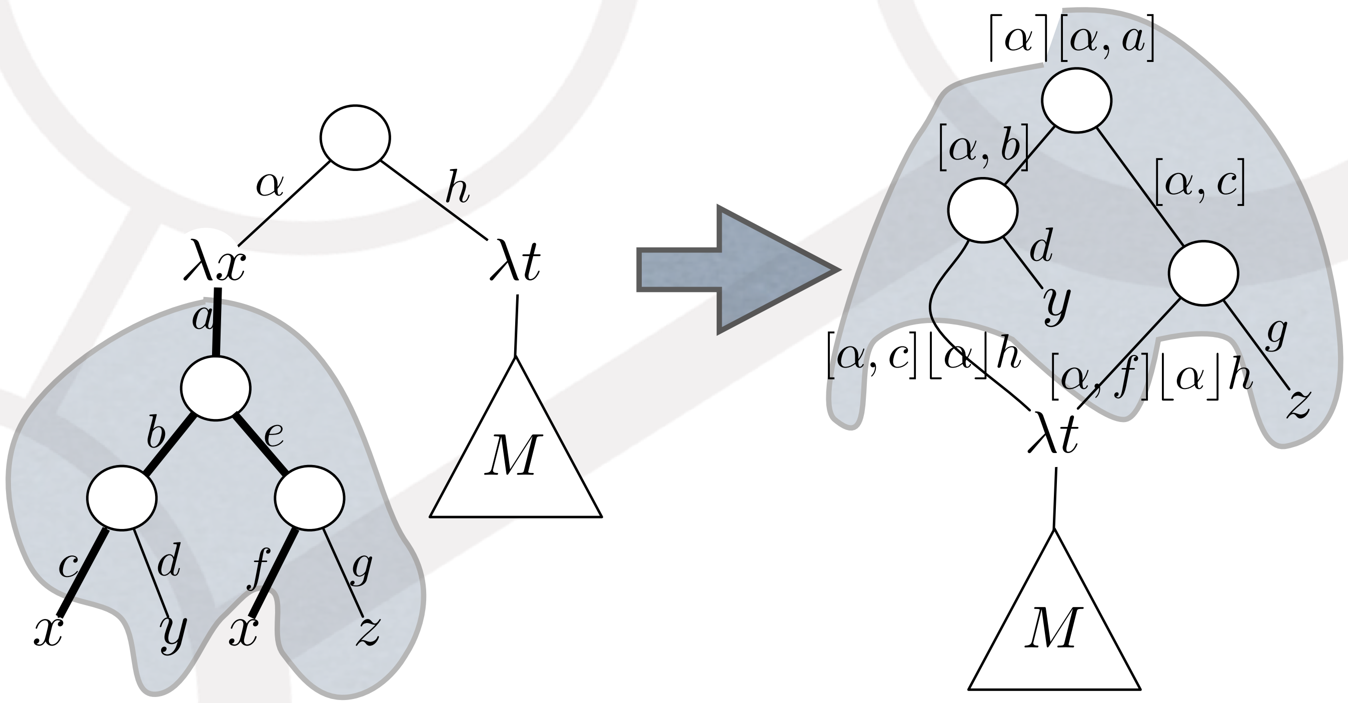
Strong labeled lambda-calculus

- all redexes with same name are contracted in **single** step
- these **complete** normal order reductions are optimal
- theory of **redex families**



Weak labeled lambda-calculus

- under lambdas compute subterms with **no occurence** of bound variable
- strong labeled theory + **tagging paths** to occurences of the bound variable
- confluent calculus

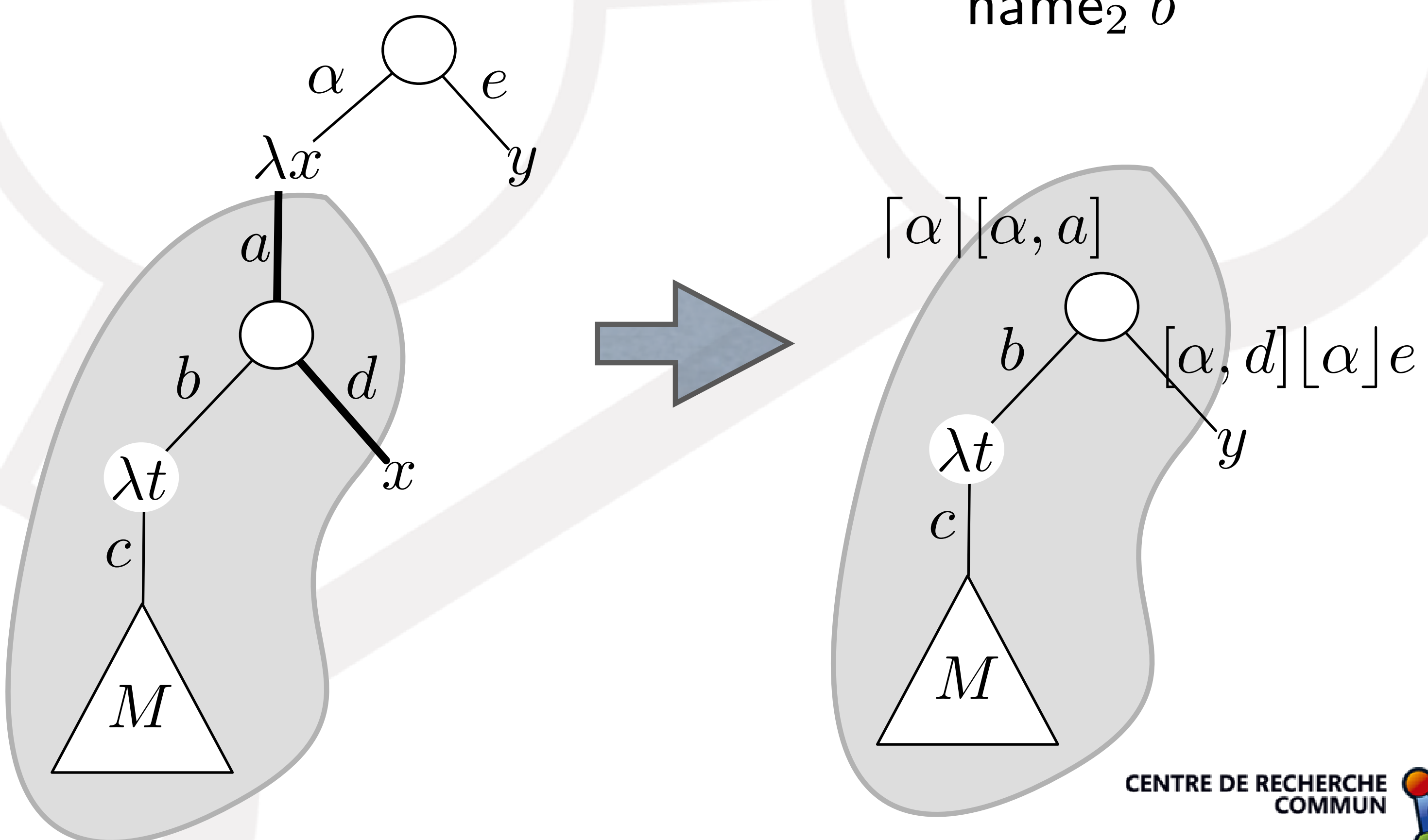


Weak labeled lambda-calculus

- problem with K-terms.
- name of creating redex does not appear in name of created redex

name$_1$ α

name$_2$ b

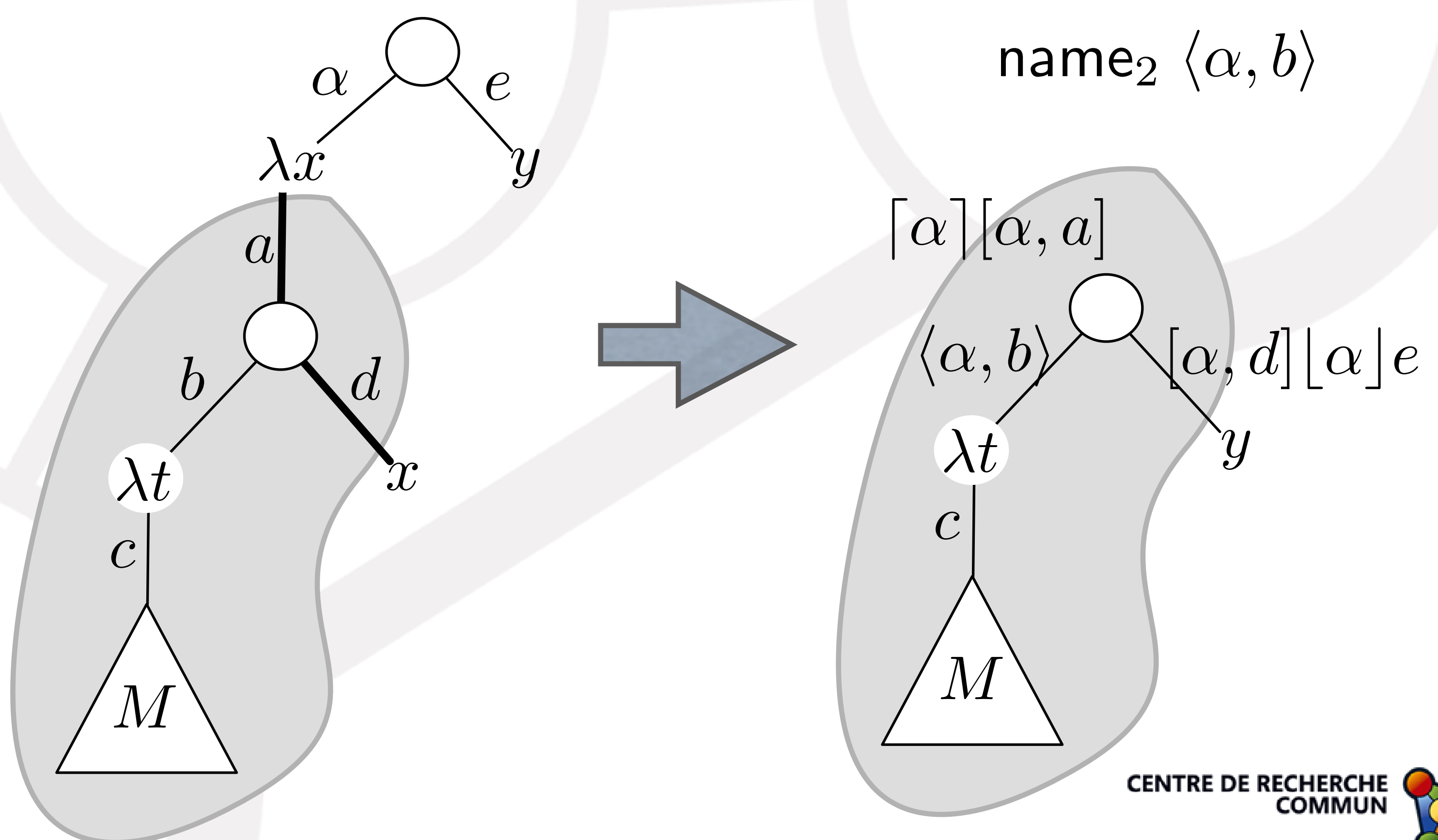


Weak labeled lambda-calculus(1)

- [Klop 60]
  - tag lefts of application nodes when K-terms
  - **special tag** since corresponding node is not duplicated

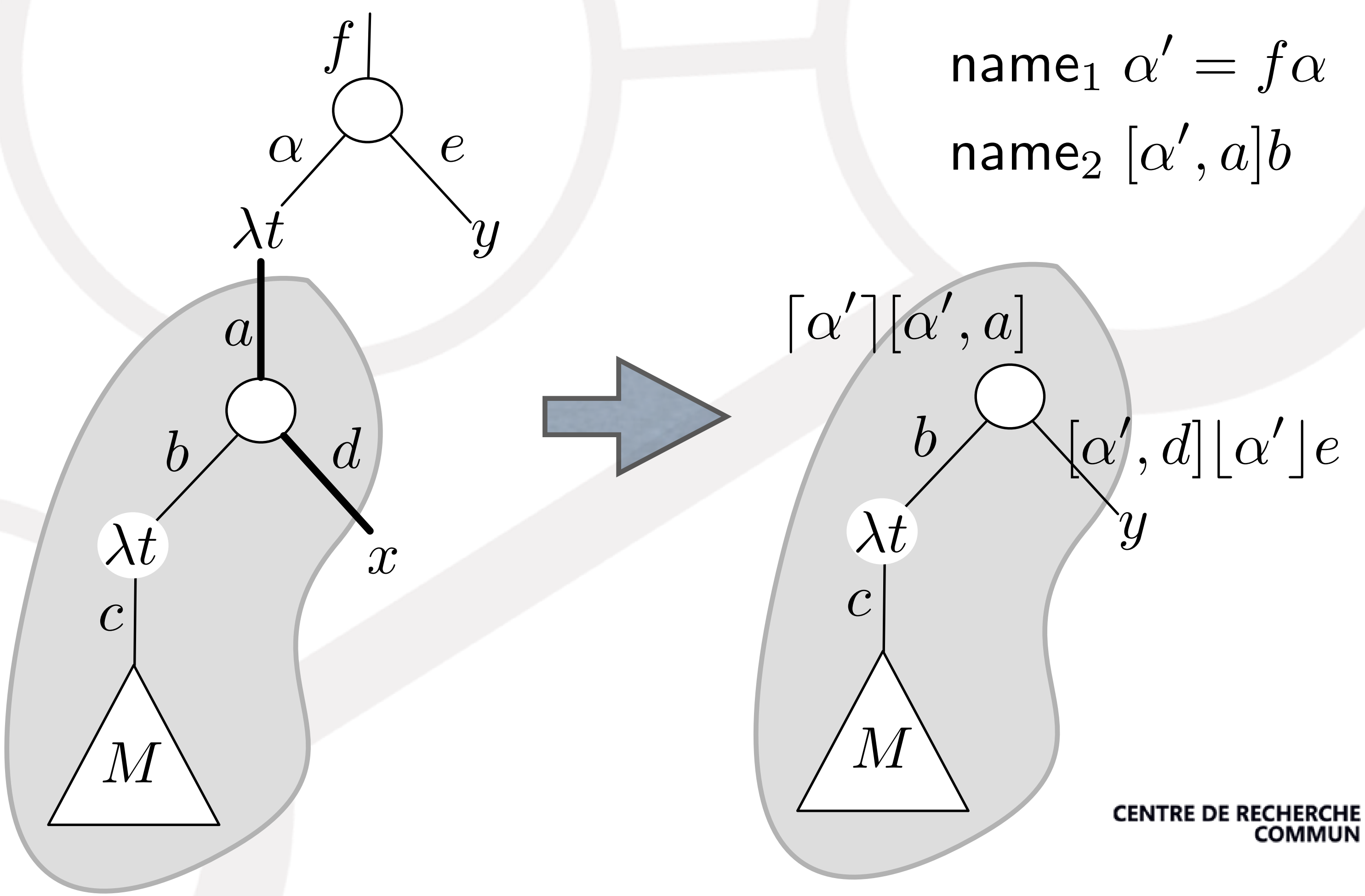
name$_1$ α

name$_2$ $\langle \alpha, b \rangle$



Weak labeled lambda-calculus (2)

- [Barendregt 60]
  - **add** the atomic label of applications **to the names** of redexes.
  - and redo all theory of (weak) labelled calculus

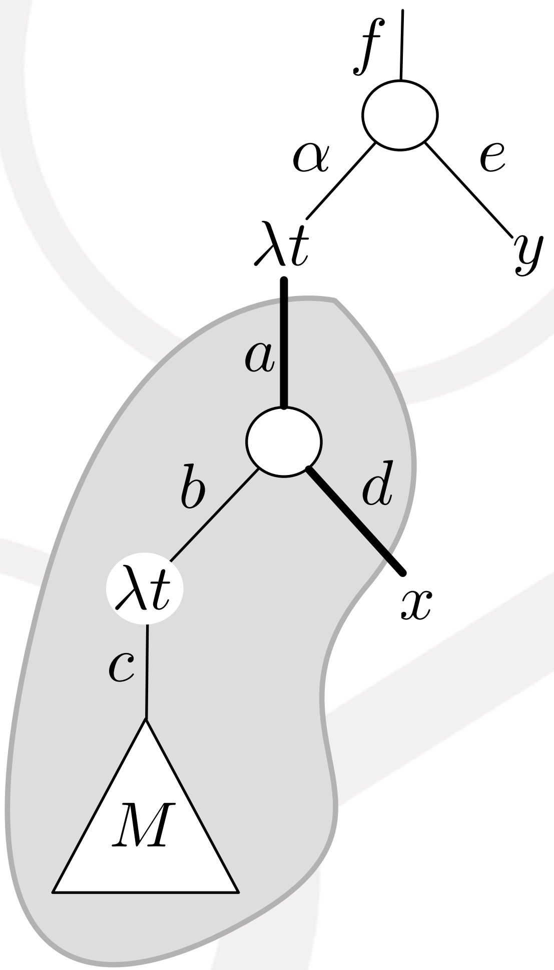


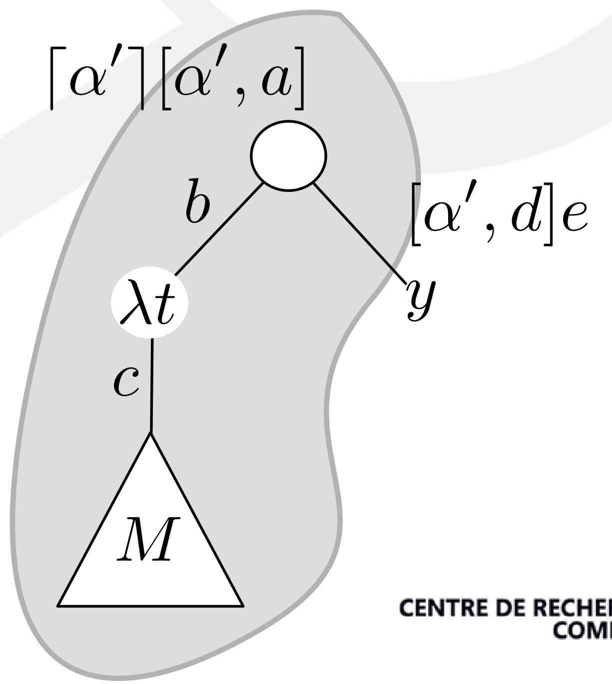
name$_1$ $\alpha' = f\alpha$

name$_2$ $[\alpha', a]b$

Weak labeled lambda-calculus (3)

- [Geuvers 50 ?]
  - keep $\lceil \alpha \rceil$ because of K-redexes
  - $\lfloor \alpha \rfloor$ are useless





$\mathsf{name}_1\ \alpha' = f\alpha$

$\mathsf{name}_2\ [\alpha', a]b$

THEOREMS

Weak lambda-calculus

- Terms

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

- Rules

$$(\beta)\quad R = (\lambda x.M)N \xrightarrow{R} M[\![x\backslash N]\!] \qquad\qquad (\nu)\ \frac{M \xrightarrow{R} M'}{MN \xrightarrow{R} M'N}$$

$$(\xi')\ \frac{M \xrightarrow{R} M' \quad x \notin R}{\lambda x.M \xrightarrow{R} \lambda x.M'} \qquad\qquad (\mu)\ \frac{N \xrightarrow{R} N'}{MN \xrightarrow{R} MN'}$$

$$(w)\ \frac{M \xrightarrow{R} N}{M \to N}$$

Weak labeled lambda-calculus

- Terms

$$
\begin{array}{rcll}
U, V & ::= & \alpha : X & \text{labeled terms} \\
X, Y & ::= & S \mid U & \text{clipped or labeled terms} \\
S, T & ::= & x \mid UV \mid \lambda x.U & \text{clipped terms}
\end{array}
$$

- Labels

$$
\begin{array}{rcll}
\alpha, \beta & ::= & a \mid \lceil \alpha' \rceil \mid \lfloor \alpha' \rfloor \mid [\alpha', \beta] & \text{labels} \\
\alpha', \beta' & ::= & \alpha_1 \alpha_2 \cdots \alpha_n \quad (n \geq 1) & \text{compound labels}
\end{array}
$$

- Rules

$$
(\ell) \qquad R = \beta : ((\alpha' \cdot \lambda x.U)V) \xrightarrow{R} \lceil \beta\alpha' \rceil : (\beta\alpha' \circledast U)[\![x \backslash \lfloor \beta\alpha' \rfloor : V]\!]
$$

$$
\alpha_1 \alpha_2 \cdots \alpha_n \cdot X = \alpha_1 : \alpha_2 : \cdots \alpha_n : X
$$

Weak labeled lambda-calculus

- Diffusion

$$
\begin{array}{rcl}
\alpha' \textcircled{x} X & = & X \ \text{ if } x \notin X \\
\alpha' \textcircled{x} x & = & x \\
\alpha' \textcircled{x} UV & = & (\alpha' \textcircled{x} U \ \ \alpha' \textcircled{x} V) \ \text{ if } x \in UV \\
\alpha' \textcircled{x} \lambda y.U & = & \lambda y.\, \alpha' \textcircled{x} U \ \text{ if } x \in \lambda y.U \\
\alpha' \textcircled{x} \beta : X & = & [\alpha', \beta] : \alpha' \textcircled{x} X \ \text{ if } x \in X
\end{array}
$$

- Substitution

$$
\begin{array}{rcl}
x[[x\backslash W]] & = & W \\
y[[x\backslash W]] & = & y \\
(UV)[[x\backslash W]] & = & U[[x\backslash W]]\, V[[x\backslash W]] \\
(\lambda y.U)[[x\backslash W]] & = & \lambda y.U[[x\backslash W]] \\
(\beta : X)[[x\backslash W]] & = & \beta : X[[x\backslash W]]
\end{array}
$$

Weak labeled lambda-calculus

- Labels containment:

$$\alpha' \prec \lfloor \alpha' \rfloor$$

$$\alpha' \prec \lceil \alpha' \rceil$$

$$\alpha' \prec [\alpha', \beta]$$

$$\alpha' \prec \beta_i \;\Rightarrow\; \alpha' \prec \beta_1 \cdots \beta_n$$

$$\alpha' \prec \beta' \prec \gamma' \;\Rightarrow\; \alpha' \prec \gamma'$$

Weak labeled lambda-calculus

- Maximality invariant

  $\mathcal{Q}(W)$ ::= we have $\alpha' \not\leq \beta$ for every redex R with name α' and any subterm $\beta : X$ in W.

- Lemma 1

  If $\mathcal{Q}(W)$ and $W \overset{\gamma'}{\Longrightarrow} W'$, then $\mathcal{Q}(W')$.

Weak labeled lambda-calculus

- Maximality invariant

  $\mathcal{Q}(W)$ ::= we have $\alpha' \not\leq \beta$ for every redex R with name α' and any subterm $\beta : X$ in W.

- Lemma 1

  If $\mathcal{Q}(W)$ and $W \stackrel{\gamma'}{\Longrightarrow} W'$, then $\mathcal{Q}(W')$.

- Lexical scope invariant

  $\mathcal{R}(W)$ ::= for any pair of subterms $\alpha : x$ and $\alpha : y$ in W, we have x free in W iff y free in W.

- Lemma 2

  If $\mathcal{R}(W)$ and $W \to W'$, then $\mathcal{R}(W')$

Weak labeled lambda-calculus

- Maximality invariant

  $\mathcal{P}(W)$ ::= for any pair of subterms $\alpha : X$ and $\alpha : Y$ in W, we have $X = Y$.

- Sharing lemma

  If $\mathcal{P}(W) \wedge \mathcal{Q}(W) \wedge \mathcal{R}(W)$ and $W \stackrel{\gamma'}{\Longrightarrow} W'$, then $\mathcal{P}(W')$.

- Sharing theorem

  $Init(U)$ ::= every subterm of U is labeled with a distinct letter.

  Let $Init(U)$ and $U \Longrightarrow V$, then $\mathcal{P}(V)$.

CONCLUSION

Conclusion

- weak lambda calculus implemented with **dags**
- **useful** for programming languages ?
- do theory for weak labeled lambda calculus (3)
- and if **explicit substitutions** ?
- do theory as particular case of **term rewriting systems**
- big difference between weak and strong calculus (POPL mark)

Conclusion

- weak lambda calculus implemented with **dags**
- **useful** for programming languages ?
- do theory for weak labeled lambda calculus (3)
- and if **explicit substitutions** ?
- do theory as particular case of **term rewriting systems**
- big difference between weak and strong calculus (POPL mark)

Rendez-vous in 2017...

CENTRE DE RECHERCHE
COMMUN

INRIA
MICROSOFT RESEARCH