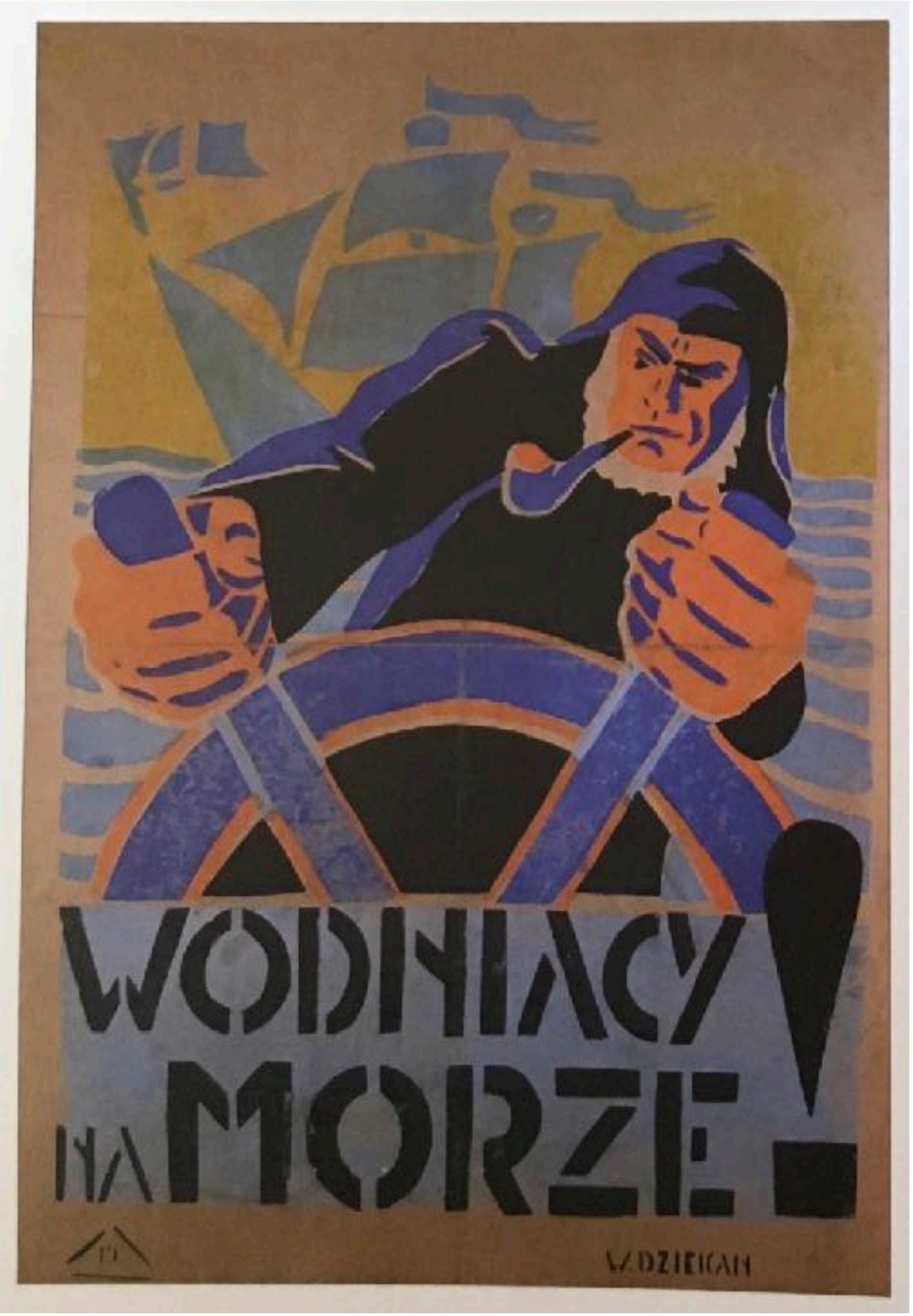
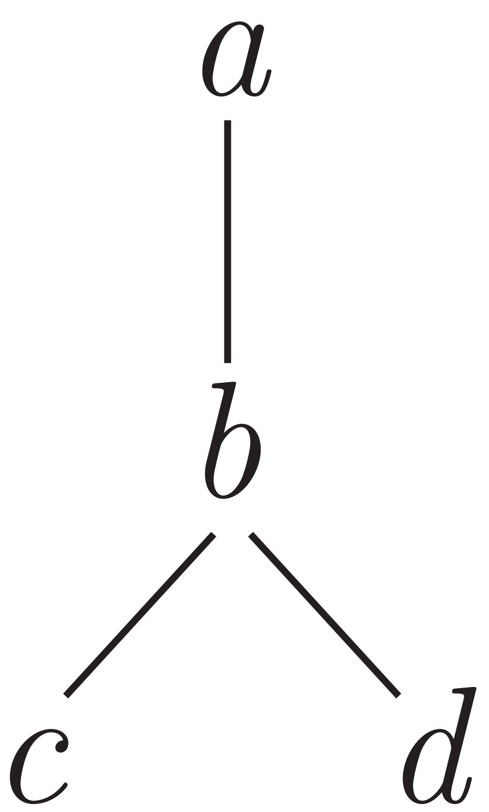
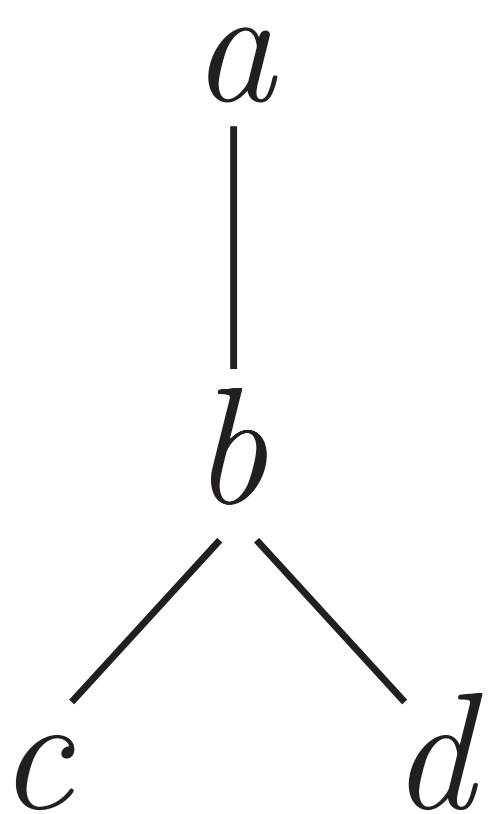
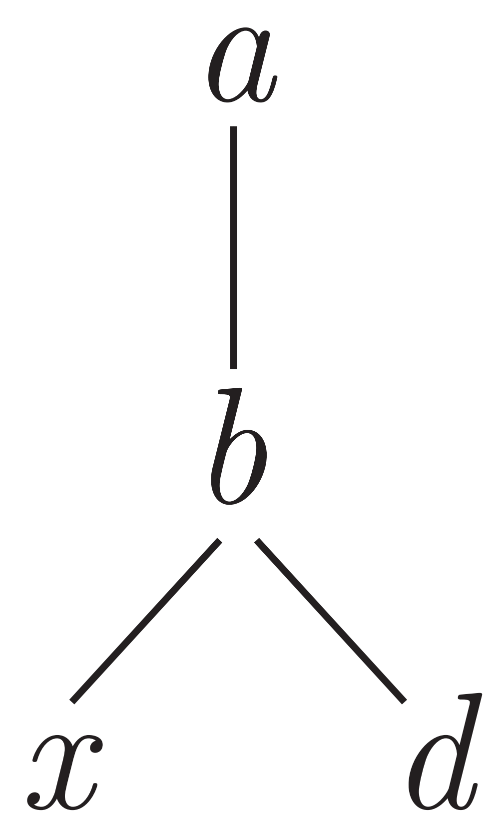


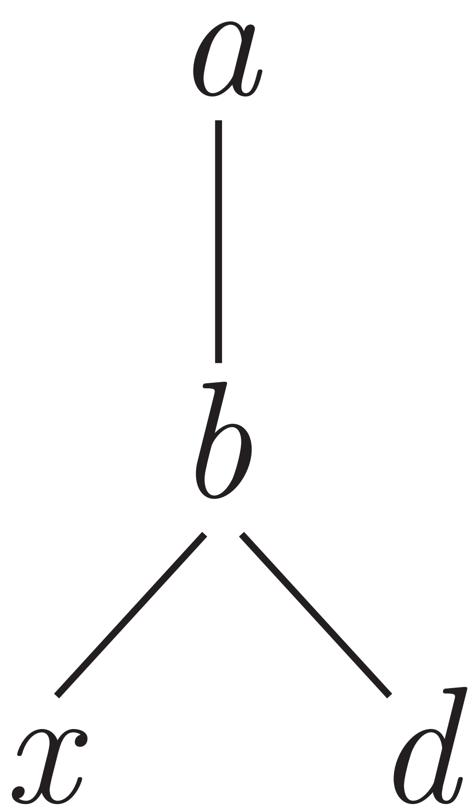
Lambda Y
calculus

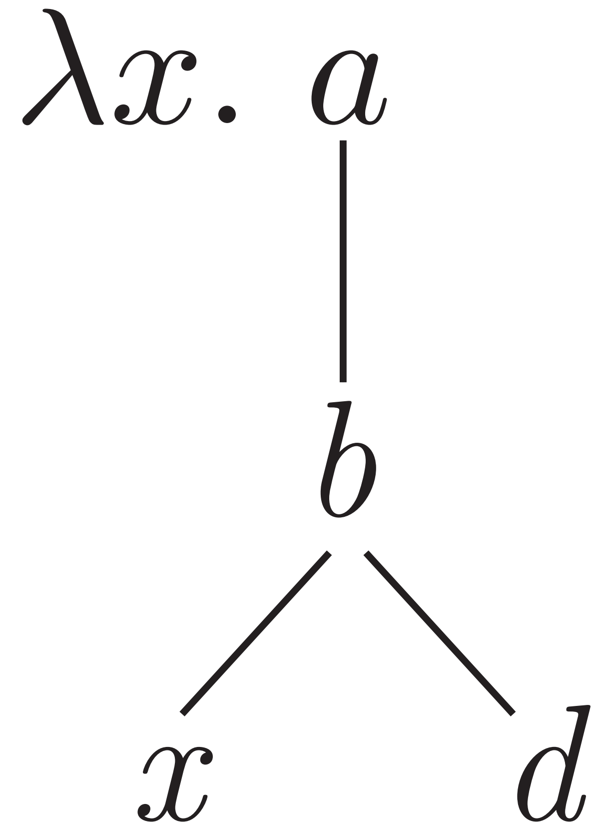

a
b
c
d



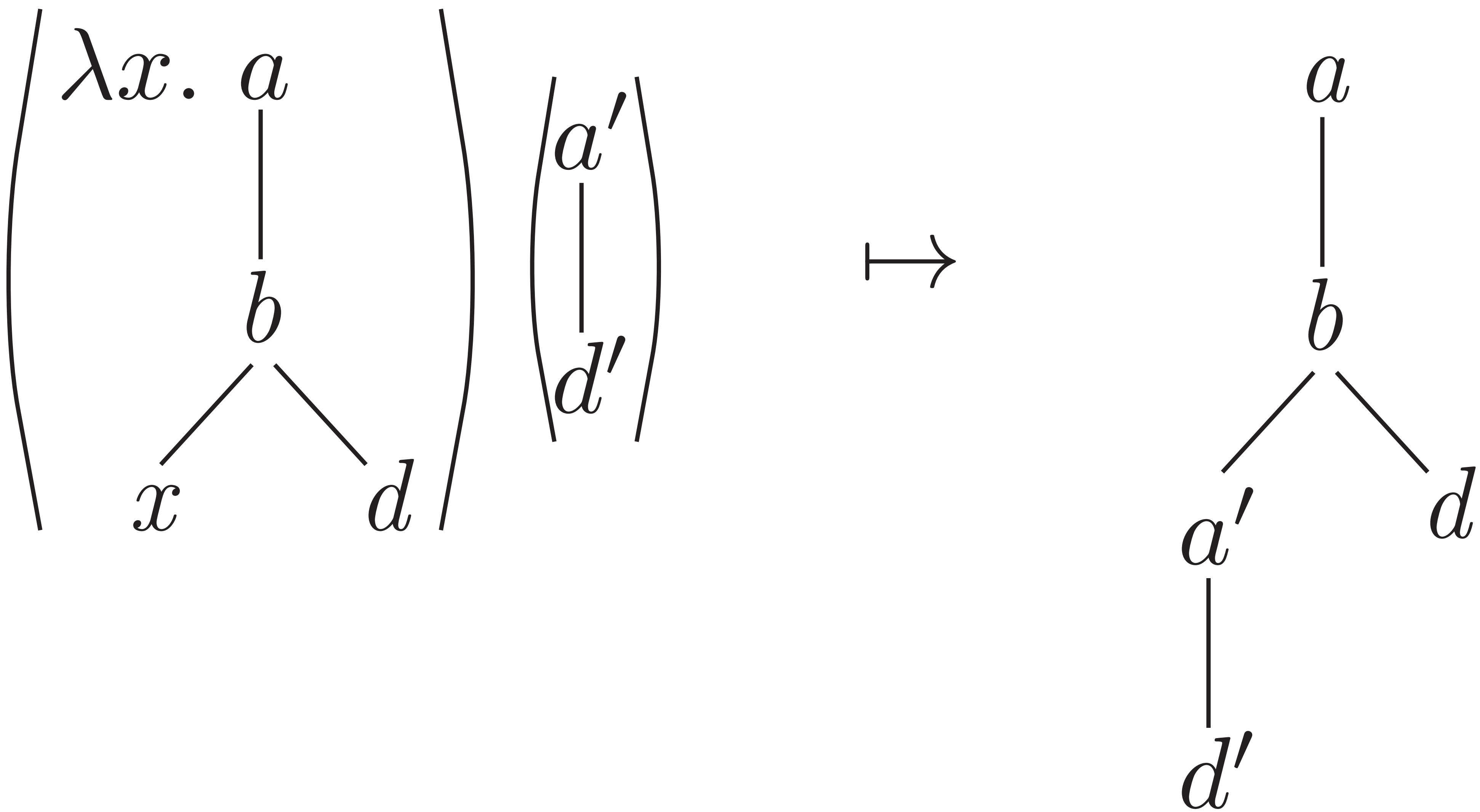


x a variable

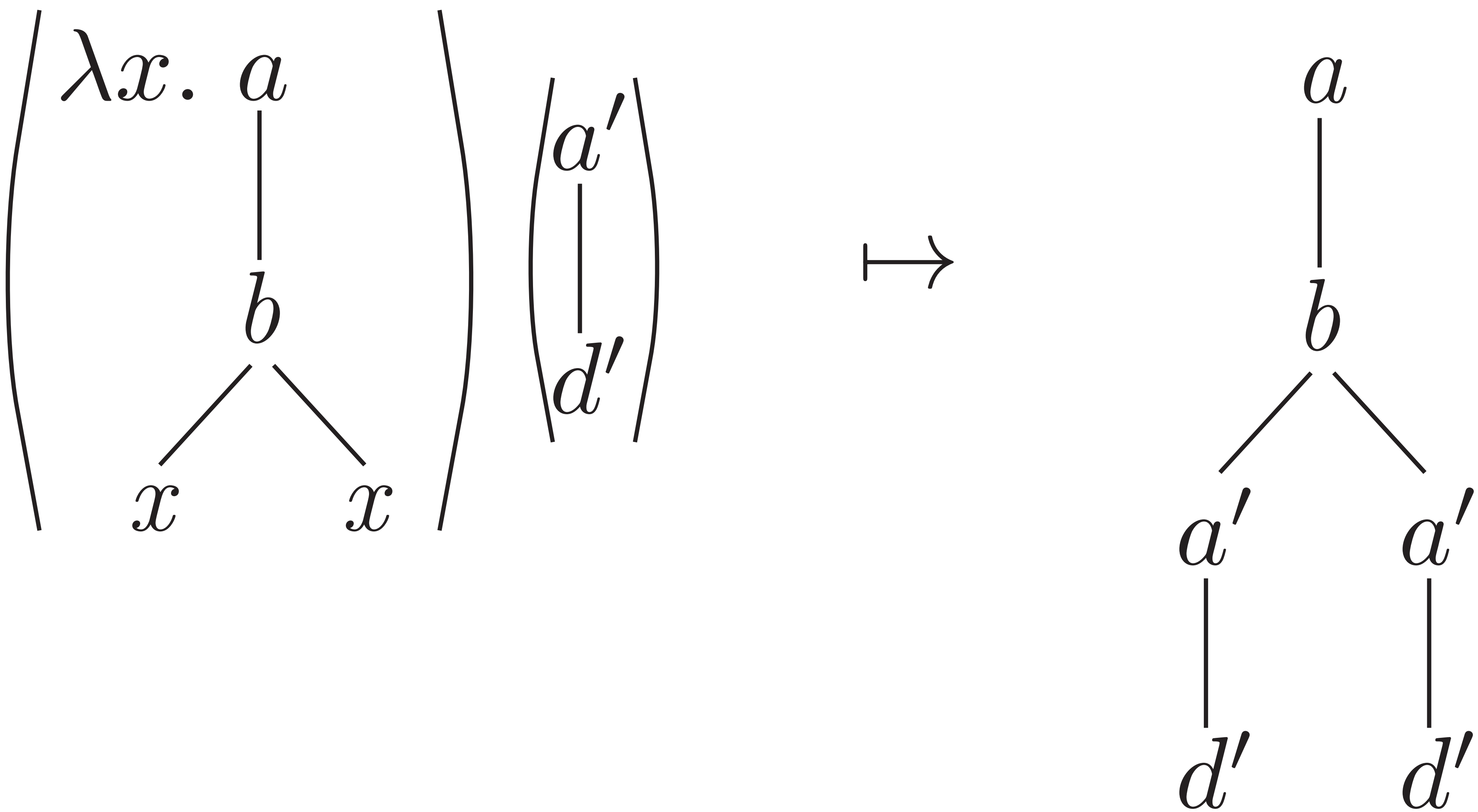




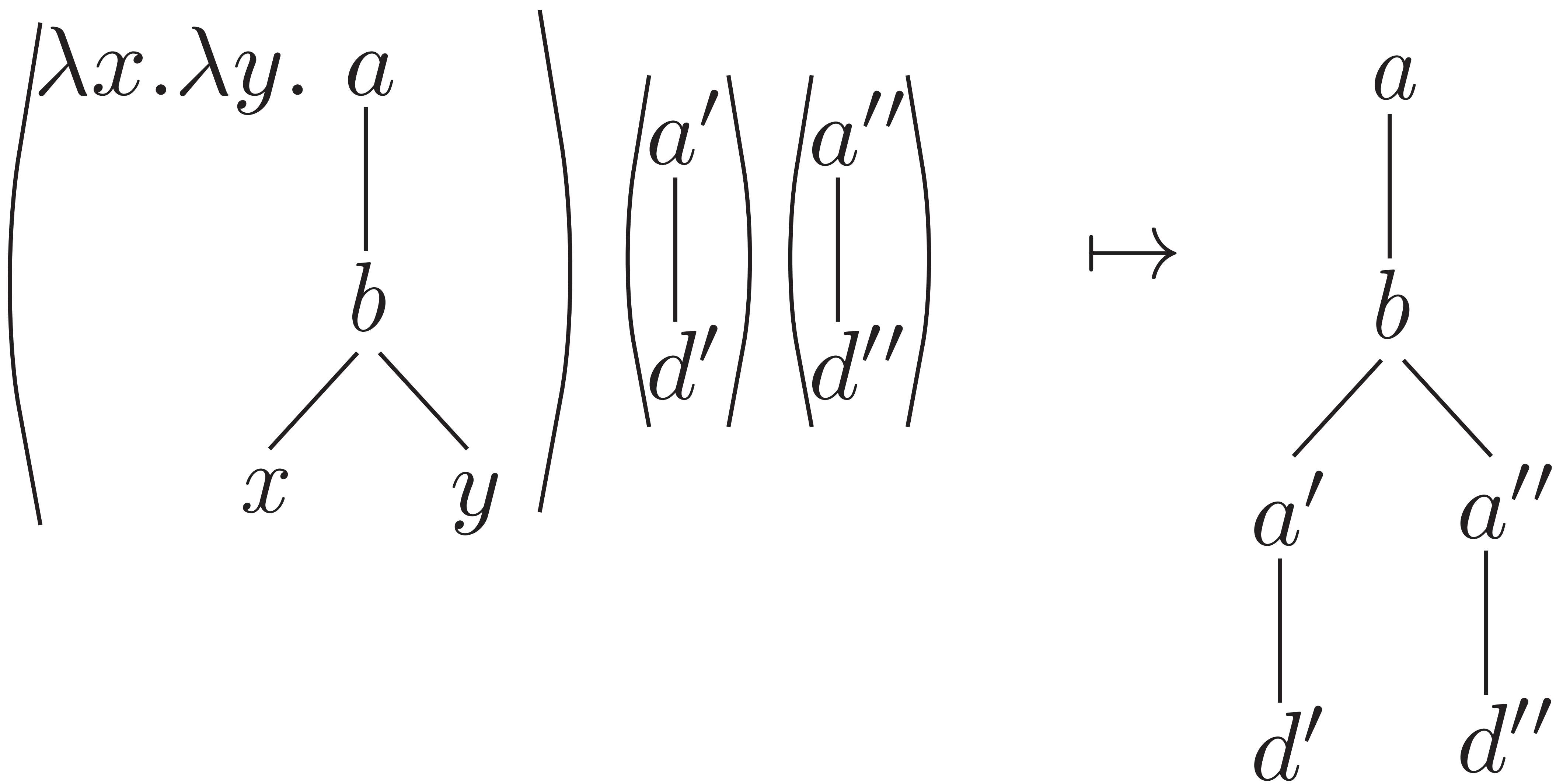
A context



Application of a context to a tree

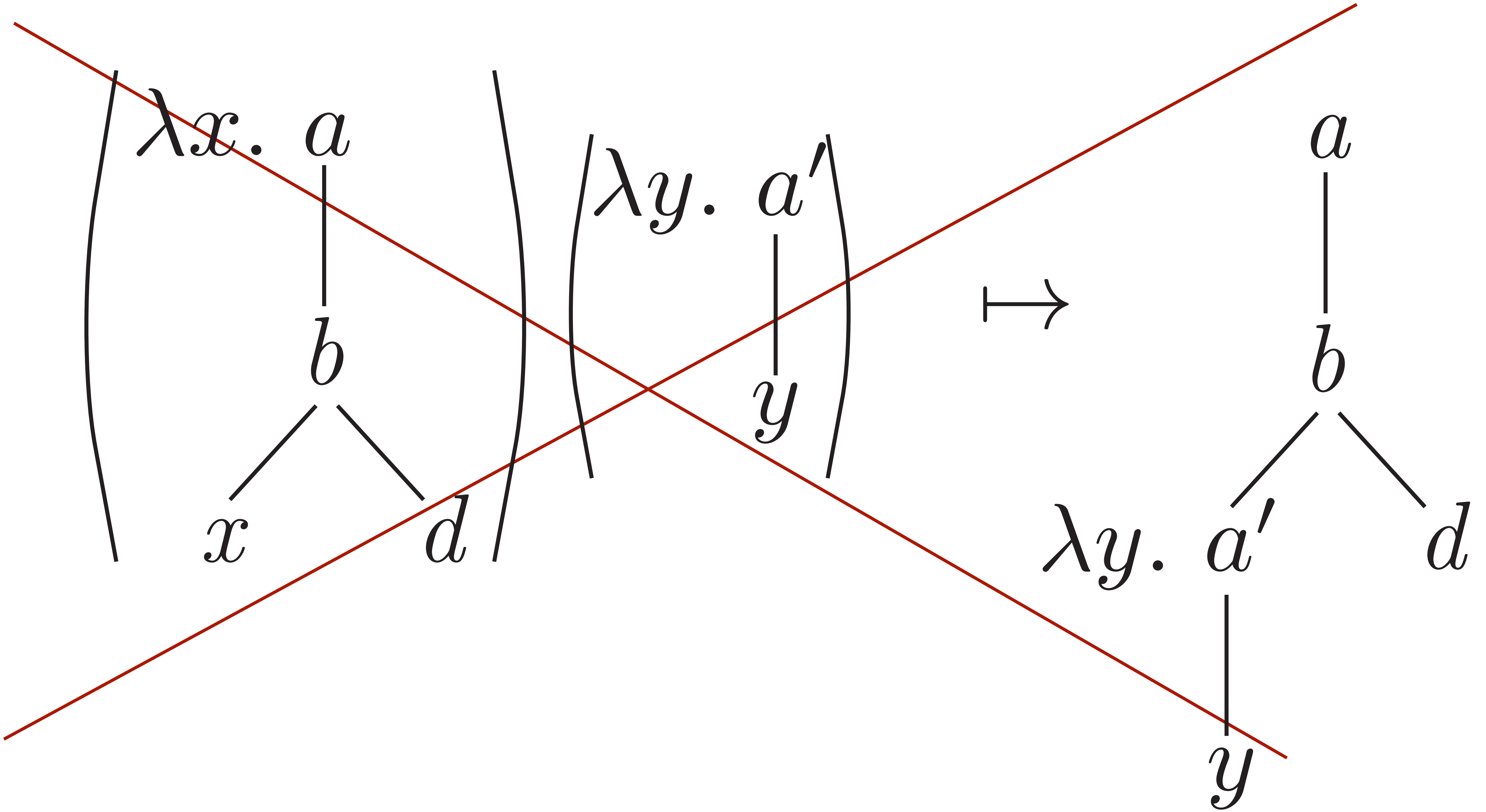


Application of a context with two holes



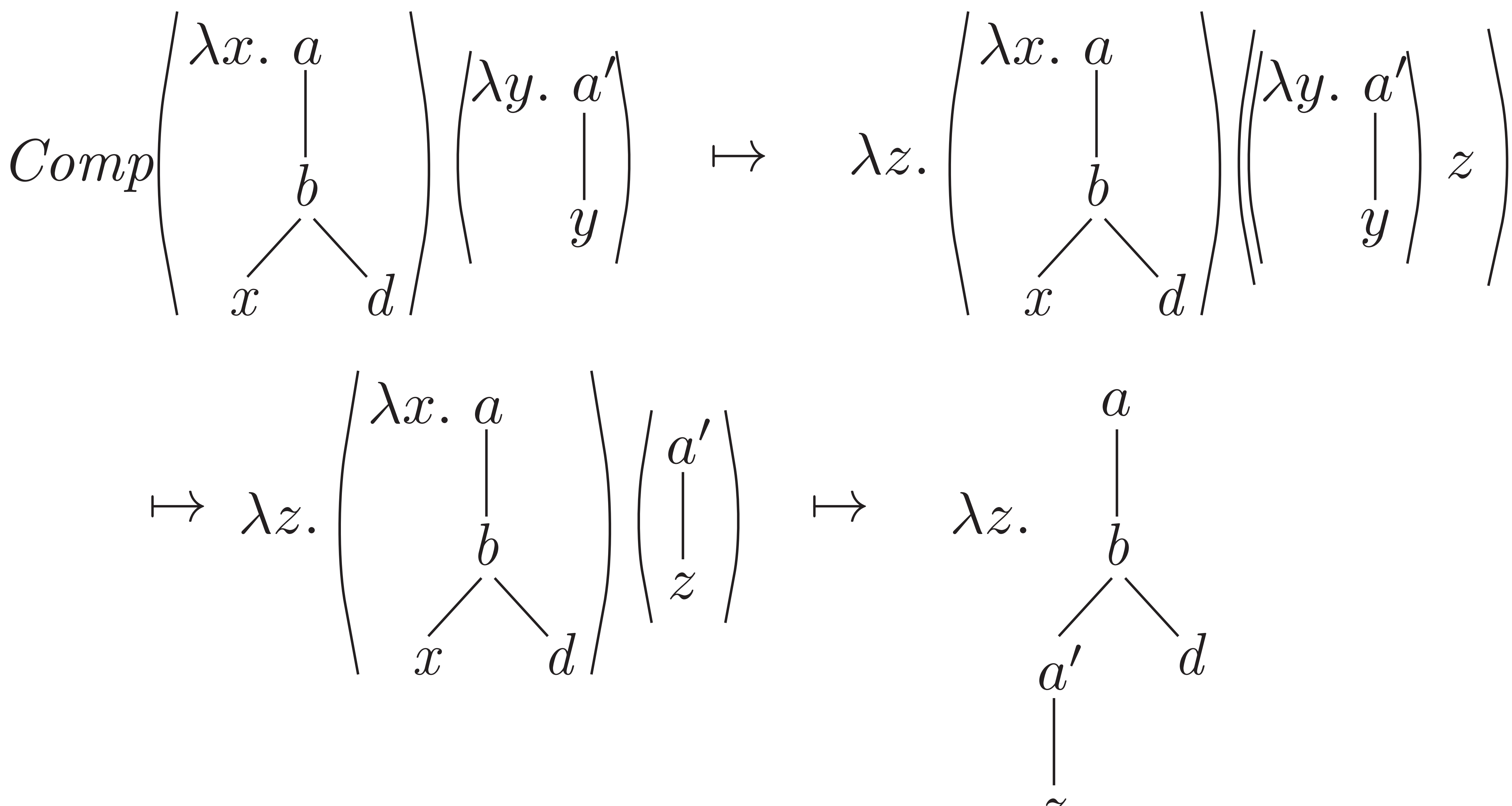
Application of a context with two distinct holes

How to compose contexts?



How to compose contexts?

$$Comp \equiv \lambda p.\lambda q.\lambda z.\ p(q(z))$$



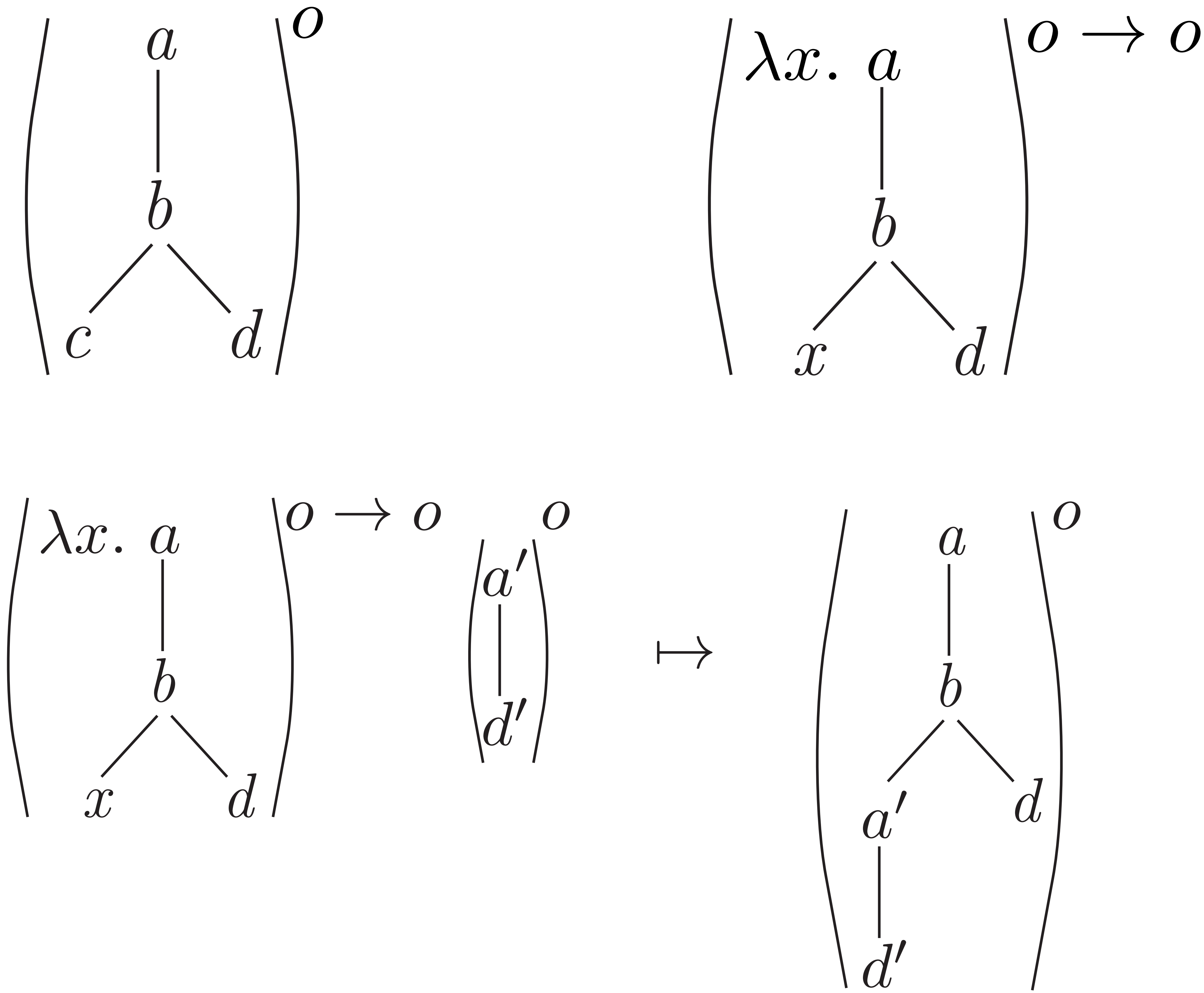
Can we apply anything to anything?
Or, shall we distinguish between trees and contexts?

Does $\lambda x.\ x$ | x make sense?

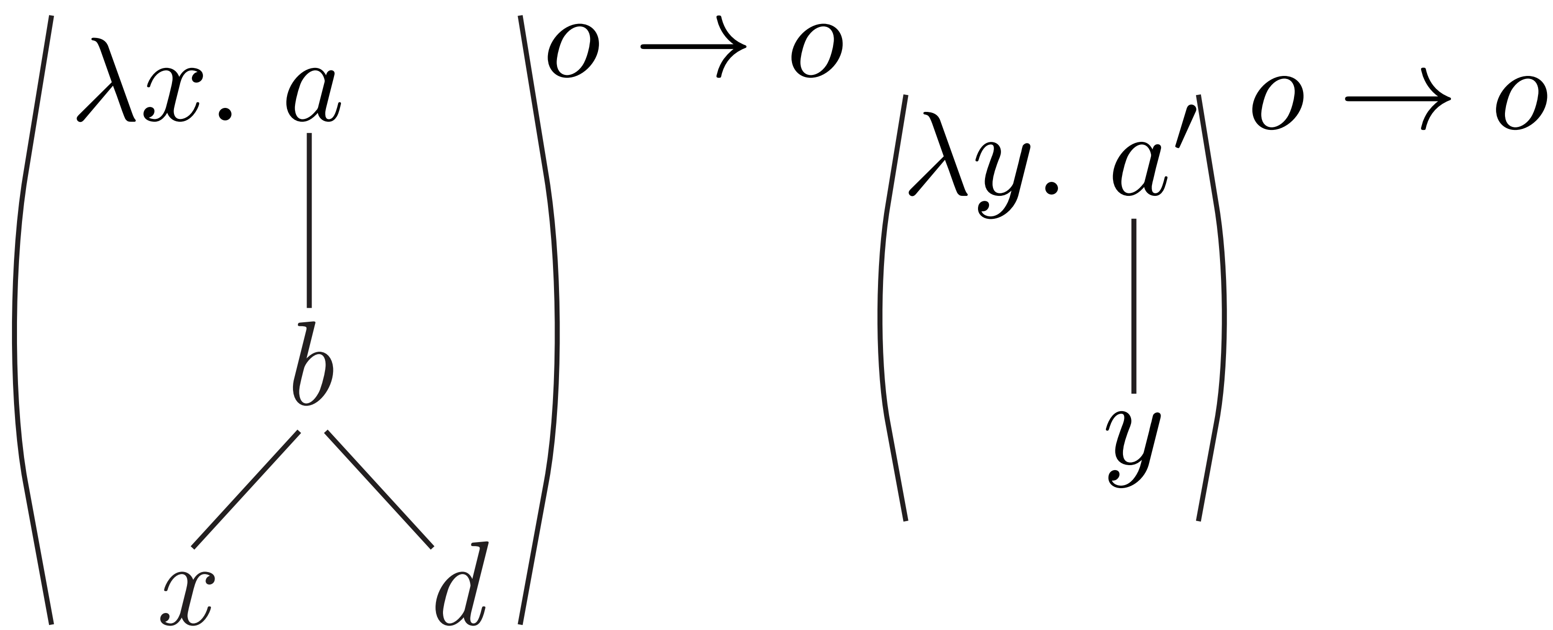
yes → Turing complete

no → **Types**

Types (simple types)



The problematic application is not well-typed:



Typing the composition of contexts:

$$Comp \equiv \lambda p^{o\to o}.\lambda q^{o\to o}.\lambda z^{o}.\ p(q(z)) : (o \to o) \to (o \to o) \to o \to o$$

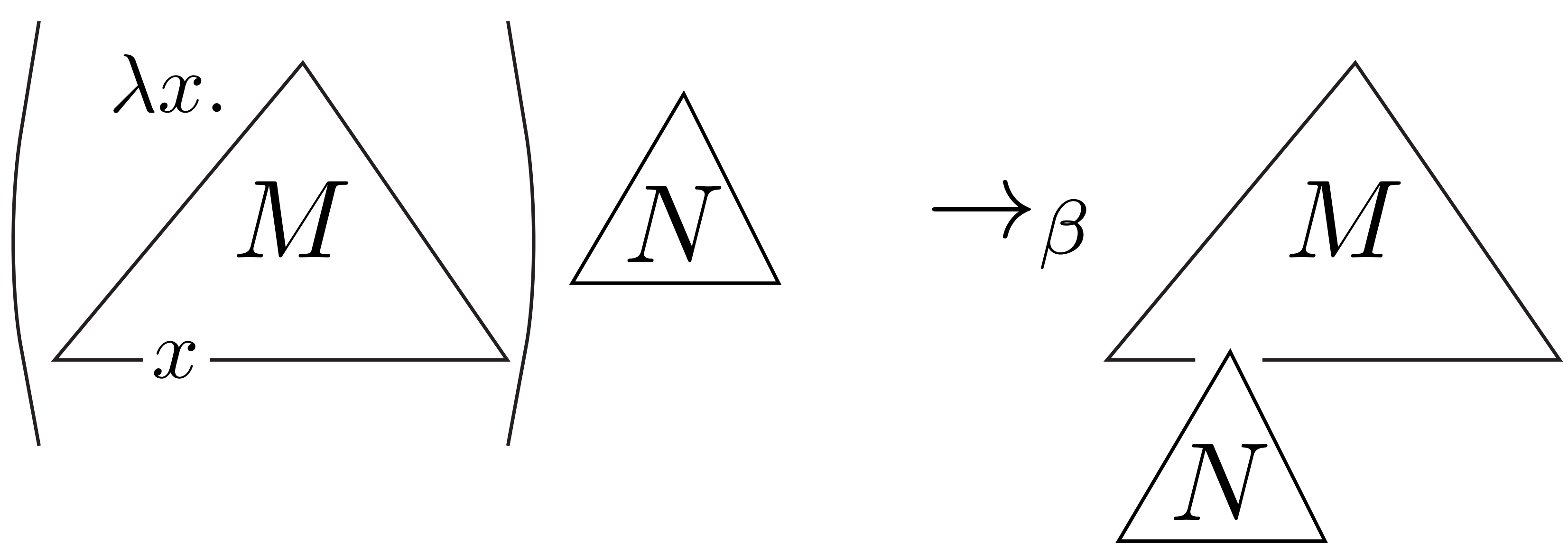
λ-calculus (simply typed)

Types: $o,\ A \to B$

Terms: $c,\ x,\ MN,\ \lambda x.M$

Typed terms: $c^A,\ x^A,\ (M^{(A\to B)}N^A)^B,\ (\lambda x^A.M^B)^{A\to B}$

β-reduction: $(\lambda x.M)N \to_\beta M[N/x]$



β**-reduction:** $(\lambda x.M)N \to_\beta M[N/x]$

Example:

$$\big(\lambda f^{o\to o}\lambda x^o.f(fx)\big)ad \;\to_\beta\; \big(\lambda x^o.a(ax)\big)d \;\to_\beta\; a(a(d))$$

$\boldsymbol{\beta}$**-reduction:** $(\lambda x.M)N \to_\beta M[N/x]$

Example:

$$\big(\lambda f^{o\to o}\lambda x^o.f(fx)\big)ad \ \to_\beta \ \big(\lambda x^o.a(ax)\big)d \ \to_\beta \ a(a(d))$$

Substitution should avoid variable capture (as in logic) :

$$(\lambda x.\lambda y.\ x)y \ \to_\beta \ \lambda z.y$$

and not $\lambda y.y$

Example (QBF)

- $\mathsf{tt} = \lambda xy.\ x$, $\quad \mathsf{ff} = \lambda xy.\ y$, $\quad$ They are of type $0 \to 0 \to 0$.
- $and = \lambda b_1 b_2 xy.\ b_1(b_2 xy)y$, $\quad or = \lambda b_1 b_2 xy.\ b_1 x(b_2 xy)$,
- $\mathsf{neg} = \lambda bxy.\ byx$
- $\mathsf{All} = \lambda f.\ and(f\,\mathsf{tt})(f\,\mathsf{ff})$, $\quad \mathsf{Exists} = \lambda f.\ or(f\,\mathsf{tt})(f\,\mathsf{ff})$.

QBF to terms Every QBF formula α can be translated to a term M_α:

$$\forall x.\exists y.\ x \wedge \neg y \quad \mapsto \quad \mathsf{All}(\lambda x.\ \mathsf{Exists}(\lambda y.\ and\ x\ (\mathsf{neg}\ y)))$$

Fact For every QBF sentence α:

$$\alpha \text{ is true} \quad \text{iff} \quad M_\alpha \text{ evaluates to } \mathsf{tt}.$$

- Early beginning with Frege (1893) and Schönfinkel (1924).
- Conceived by Church (1932-1933) as part of a general theory of functions and logic.
- General theory shown inconsistent by Kleene & Rosner (1936), but the functional part has become successful.
- All computable functions are representable in lambda-calculus Kleene & Rosner (1936), Turing (1937).
  Equivalence of two lambda-terms is the first known undecidable problem.
- Typed version has been introduced by Curry (1936), and Church (1940).
- In the 60-ties Scott gives mathematical semantics to the calculus.
- Applications to functional languages, and to linguistics start.

- Early beginning with Frege (1893) and Schönfinkel (1924).
- Conceived by Church (1932-1933) as part of a general theory of functions and logic.
- General theory shown inconsistent by Kleene & Rosner (1936), but the functional part has become successful.
- All computable functions are representable in lambda-calculus Kleene & Rosner (1936), Turing (1937). Equivalence of two lambda-terms is the first known undecidable problem.
- Typed version has been introduced by Curry (1936), and Church (1940).
- In the 60-ties Scott gives mathematical semantics to the calculus.
- Applications to functional languages, and to linguistics start.

« The past is never dead.
It's not even past »

- Early beginning with Frege (1893) and Schönfinkel (1924).
- Conceived by Church (1932-1933) as part of a general theory of functions and logic.
- General theory shown inconsistent by Kleene & Rosner (1936), but the functional part has become successful.
- All computable functions are representable in lambda-calculus Kleene & Rosner (1936), Turing (1937).
  Equivalence of two lambda-terms is the first known undecidable problem.
- Typed version has been introduced by Curry (1936), and Church (1940).
- In the 60-ties Scott gives mathematical semantics to the calculus.
- Applications to functional languages, and to linguistics start.

« The past is never dead.
It's not even past »

« The past is a foreign country:
they do things differently there »

Every term computes to a unique result

β**-reduction:** $(\lambda x.M)N \to_\beta M[N/x]$

Df: A term is in β-*normal form* if it does not have β-redexes.

$\lambda f.\lambda x.\ f(fx)$ is in the normal from.

$\lambda f.\ (\lambda x.\ f(fx))y$ is not.

Reduction preserves typing

Every term computes to a unique result

β**-reduction:** $(\lambda x.M)N \to_\beta M[N/x]$

Df: A term is in β-*normal form* if it does not have β-redexes.

Thm [Curry'36, Church'40, Turing, Tait'67]:
Suppose $M \to_\beta^* N$ then:

- N has the same type as M (type preservation),
- if N is in the normal form then N uniquely determined (confluence),
- for every M there is N in the normal form with $M \to_\beta^* N$ (strong normalisation).

A reduction sequence can be long

$$D \;\equiv\; \lambda f^{o\to o}\lambda x^{o}.f(fx) : (o\to o)\to o\to o$$

$$D(D(Da))d \;\to_\beta\; D(Da^2)d \;\to_\beta\; D(a^4)d \;\to_\beta\; a^8 d$$

Or even very long

Let $\tau_1 \equiv o \to o$.

$$D_2 \;\equiv\; \lambda f^{\tau_1\to\tau_1}\lambda x^{\tau_1}.f(fx) : (\tau_1\to\tau_1)\to\tau_1\to\tau_1$$

Let $\tau_k \equiv \tau_{k-1} \to \tau_{k-1}$.

$$D_{k+1} \;\equiv\; \lambda f^{\tau_k\to\tau_k}\lambda x^{\tau_k}.f(fx) : (\tau_k\to\tau_k)\to\tau_k\to\tau_k$$

$$(((\ldots((D_{k+1}D_k)D_{k-1})\ldots)D_1)a)d \;\to^*\; a^{\mathit{Tower}(k+1)}c$$

How difficult it is to get the normal form?

Consider $bool \equiv o \to o \to o$.

We have $true \equiv \lambda x.\lambda y.x : bool$, and $false \equiv \lambda x.\lambda y.y : bool$

Order of a type:

$$Ord(o) = 0 \qquad Ord(A \to B) = \max(Ord(A) + 1, Ord(B)).$$

Order of a term: maximal order of a type of a sub-term.

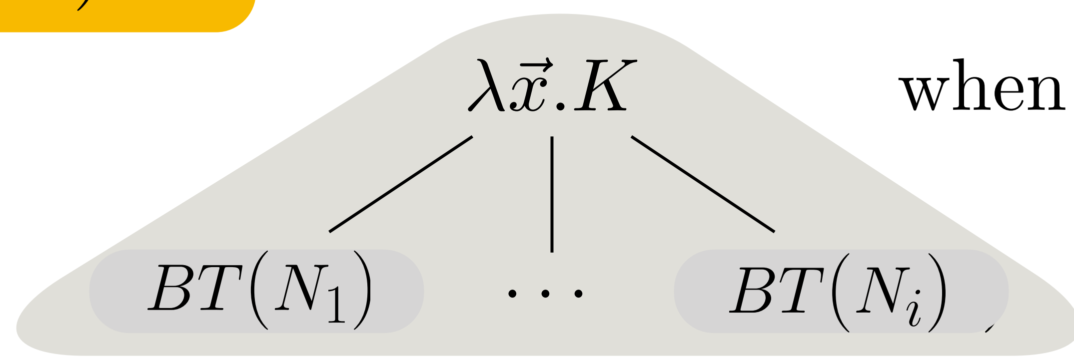
Bool-red(r): Given $M : bool$ of order r decide if $M \to_\beta^* true$.

Thm [Terui]:
Problem Bool-red($2r+2$) is r-EXPTIME-complete.
Problem Bool-red($2r+3$) is r-EXPSPACE-complete.

Böhm tree of a term

(evaluation of a term to a normal form)

$BT(M)$ is

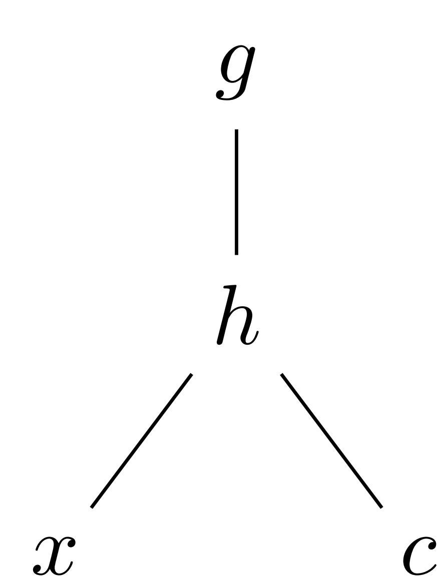


when $M \to_\beta^* \lambda\vec{x}.K N_1 \ldots N_i$

constant or variable

head normal form

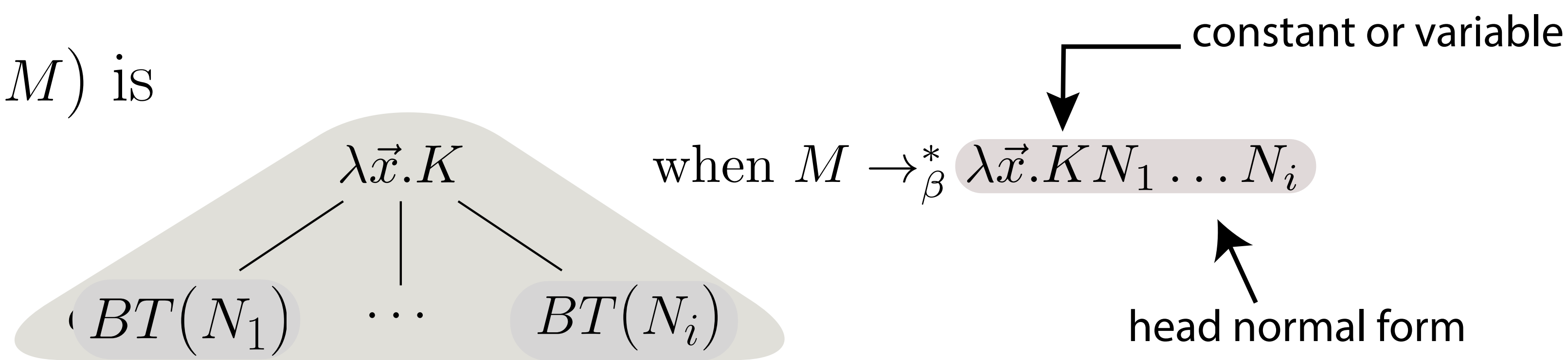
Evaluation tree of $(\lambda y.\ g\ (hxy))\ c$ is

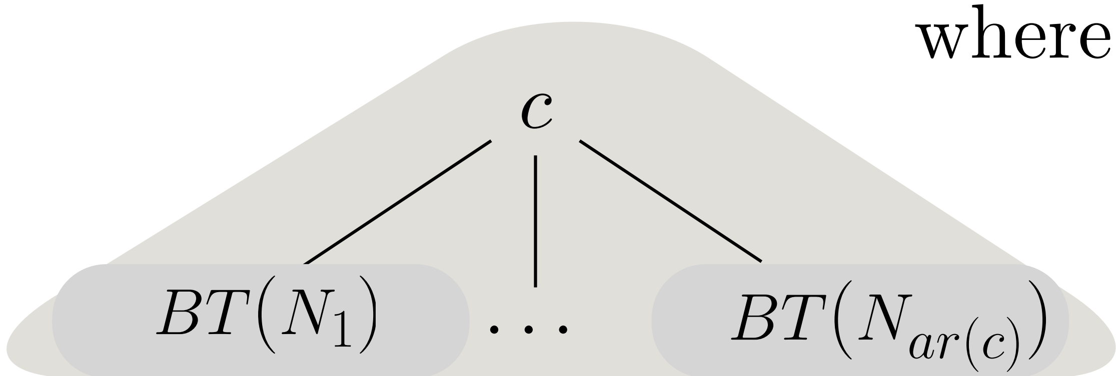


The unique result is (in some cases) a ranked tree

Tree signature:
All constants of have type of the form $o \to \cdots \to o \to o$, or just o.

$BT(M)$ is



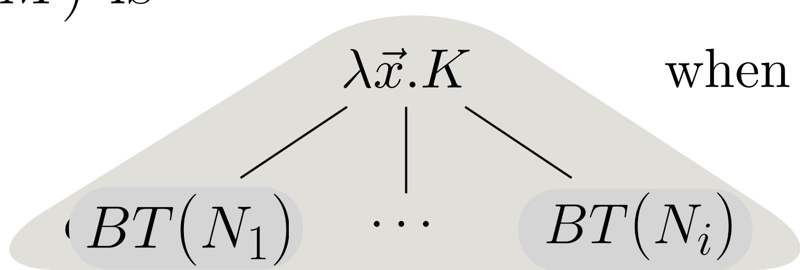


where $M \to_\beta^* cN_1 \ldots N_{ar(c)}$

The unique result is (in some cases) a ranked tree

Tree signature:
All constants of have type of the form $o \to \cdots \to o \to o$, or just o.

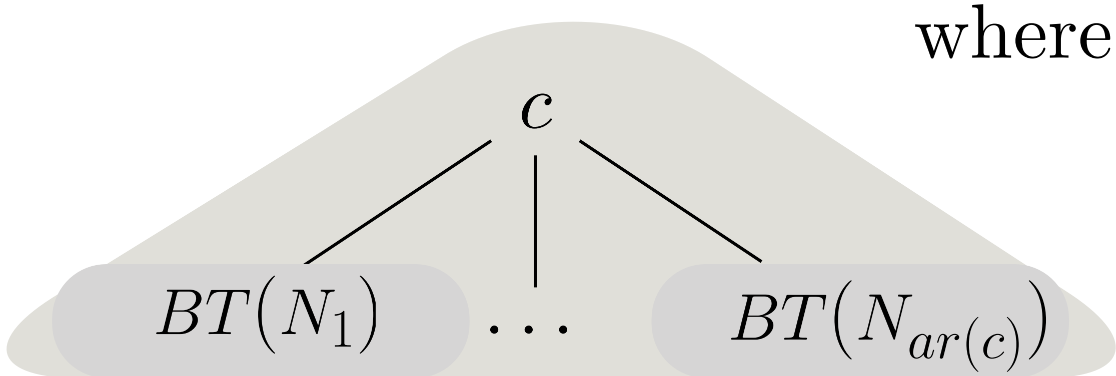
$BT(M)$ is



when $M \to_\beta^* \lambda\vec{x}.K N_1 \dots N_i$

constant or variable

head normal form



where $M \to_\beta^* c N_1 \dots N_{ar(c)}$

Cor: A normal form of a term $M : 0$ over a tree signature is a finite ranked tree.

A simply-typed **λ**-term evaluates to a finite ranked tree

$$M \to_\beta^* BT(M)$$

Infinite computations are obtained by adding a fix-point operator

λY-calculus (simply typed)

Types: $0,\ \alpha \to \beta$

Typed tems: $c^A,\ x^A,\ (M^{(A\to B)}N^A)^B,\ (\lambda x^A.M^B)^{A\to B},\ (Yx^A.M^A)^A$

δ-reduction: $(Yx.M) \to_\delta M[Yx.M/x]$

For example:

$$Yx.a(x) \quad \to_\delta \quad a(Yx.a(x)) \quad \to_\delta \quad aa(Yx.a(x)) \quad \to_\delta \cdots$$

The Böhm tree of $Yx.ax$ is the infinite sequence $aa\ldots$

The Böhm tree of $Yx.ax$ is the infinite sequence $aa\ldots$

$$Yx.a(x) \quad \to_\delta \quad a(Yx.a(x)) \quad \to_\delta \quad aa(Yx.a(x)) \quad \to_\delta \cdots$$

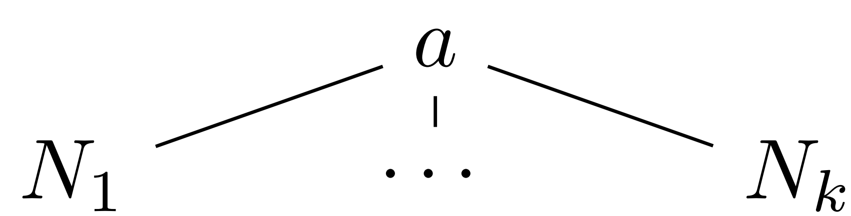
What is the Böhm tree of $Yx.x$?

$$Yx.x \quad \to_\delta \quad Yx.x \quad \to_\delta \cdots$$

By convention we say that it is Ω.

Evaluation tree of a term (Böhm tree)

- If $M \rightarrow^*_{\beta\delta} a N_1 \dots N_k$ then BT $(M) =$

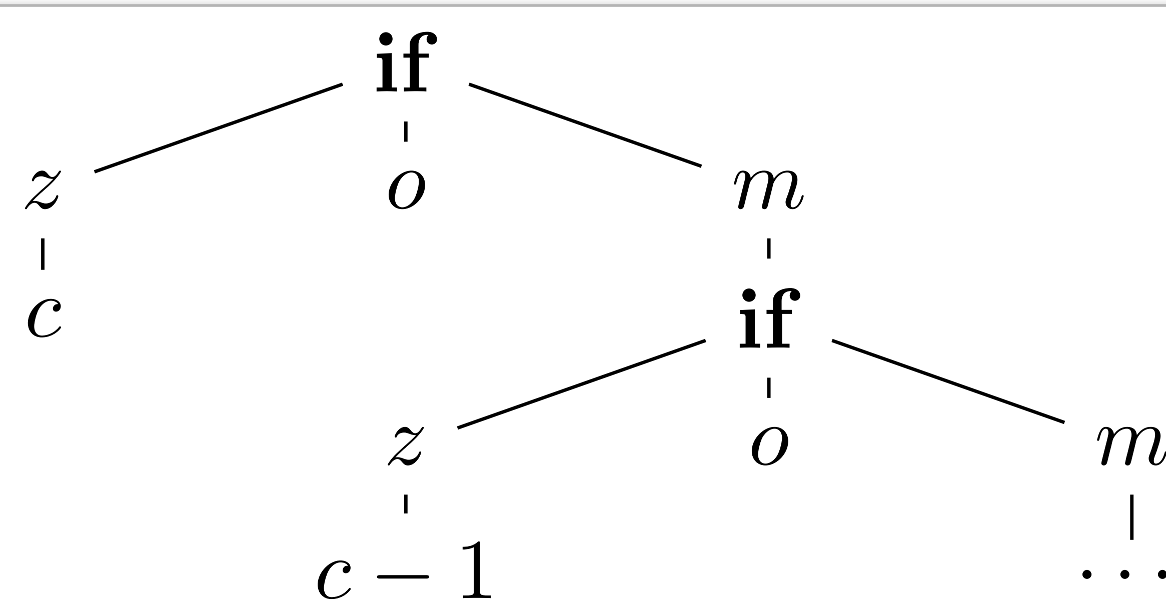


- otherwise $eval(M) = \Omega$.

$$Fct(x) \;\equiv\; \textbf{if } x = 0 \textbf{ then } 1 \textbf{ else } Fct(x-1) \cdot x \;.$$

$$Fct \equiv Y F.\ \lambda x.\ \textbf{if} - \textbf{then} - \textbf{else}(z(x), o, m(F(x-1), x))$$

$eval(Fct(c))$ is



Recursive schemes

Hierarchical equations

$$X_n =_\nu \alpha_n(\vec{X})$$

$$\vdots$$

$$X_1 =_\mu \alpha_1(\vec{X})$$

Recursive schemes

$$F_n = \lambda \vec{x}_n . M_n$$

$$\vdots$$

$$F_1 = \lambda \vec{x}_1 . M_1$$

M_i has no λ, is of type o, and contains only variables $\vec{x}_i \cup \{F_1, \ldots, F_n\}$.

Computation rule

$$F_i \vec{N} \to M_i[\vec{N}/x_i]$$