universität innsbruck

α-Avoidance

FSCD 2023, July 5 — Rome, Italy

Samuel Frontull, Georg Moser, Vincent van Oostrom

www.tcs-informatik.uibk.ac.at

Overview

1. **Motivation**
2. **α-Paths**
3. **α-Avoidance in different calculi**
4. **Soundness and Undecidability**
5. **Conclusion and Future Work**

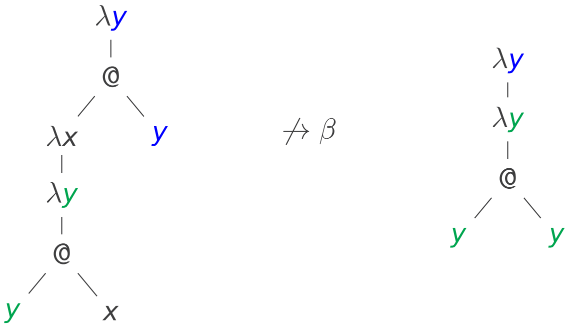
Overview

1. **Motivation**
2. **α-Paths**
3. **α-Avoidance in different calculi**
4. **Soundness and Undecidability**
5. **Conclusion and Future Work**

Substitution and bindings

β-reduction in the λ-calculus

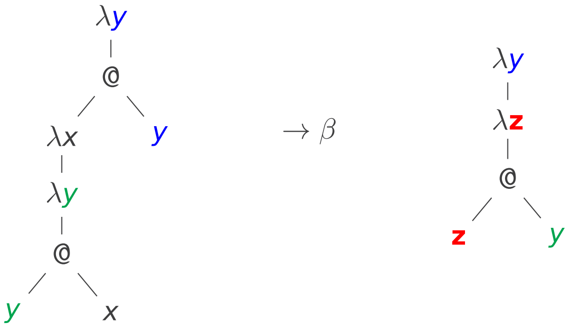
A variable capture may lead to inconsistent results.



Substitution and bindings

β-reduction in the λ-calculus

A variable capture may lead to inconsistent results.



α-Avoidance

$$M \quad \to_\beta \quad N_1 \quad \to_\beta \quad \dots \quad \to_\beta \quad N_k$$

α-Avoidance

$$M \quad \rightarrow_\beta \quad N_1 \quad \rightarrow_\beta \quad \dots \quad \rightarrow_\beta \quad N_k$$

$\rightarrow_\beta$: ordinary β-step where we may (need to) apply α.

α-Avoidance

Question

Can α-conversion steps be avoided for a λ-term M

$$M \quad \rightarrow_\beta \quad N_1 \quad \rightarrow_\beta \quad \dots \quad \rightarrow_\beta \quad N_k$$

$\rightarrow_\beta$: ordinary β-step where we may (need to) apply α.

α-Avoidance

Question

Can α-conversion steps be avoided for a λ-term M , by suitably α-converting it up front, say to a term M'

$$\begin{array}{ccccccccc} M & \to_\beta & N_1 & \to_\beta & \ldots & \to_\beta & N_k \\ \equiv_\alpha \\ M' \end{array}$$

$\to_\beta$: ordinary β-step where we may (need to) apply α.

α-Avoidance

Question

Can α-conversion steps be avoided for a λ-term M, by suitably α-converting it up front, say to a term M' such that no α-conversion step needs to be invoked along any reduction from M'.

$$
\begin{array}{ccccccccc}
M & \to_\beta & N_1 & \to_\beta & \dots & \to_\beta & N_k \\
\equiv_\alpha & & & & & & \\
M' & \to_{\beta_{naive}} & N'_1 & \to_{\beta_{naive}} & \dots & \to_{\beta_{naive}} & N'_k
\end{array}
$$

$\to_\beta$: ordinary β-step where we may (need to) apply α.

α-Avoidance

Question

Can α-conversion steps be avoided for a λ-term M, by suitably α-converting it up front, say to a term M' such that no α-conversion step needs to be invoked along any reduction from M'.

$$
\begin{array}{ccccccccc}
M & \to_\beta & N_1 & \to_\beta & \cdots & \to_\beta & N_k \\
\equiv_\alpha & & & & & & \\
M' & \color{red}{\to_{\beta_{naive}}} & N'_1 & \color{red}{\to_{\beta_{naive}}} & \cdots & \color{red}{\to_{\beta_{naive}}} & N'_k
\end{array}
$$

$\to_\beta$: ordinary β-step where we may (need to) apply α.

$\to_{\beta_{naive}}$: naïve β-step with naïve substitution (no α).

α-Avoidance

Question

Can α-conversion steps be avoided for a λ-term M , by suitably α-converting it up front, say to a term M' such that no α-conversion step needs to be invoked along any reduction from M'.

$$
\begin{array}{ccccccc}
M & \to_\beta & N_1 & \to_\beta & \dots & \to_\beta & N_k \\
\equiv_\alpha & & {\color{red}\equiv_\alpha} & & & & {\color{red}\equiv_\alpha} \\
M' & \to_{\beta_{naive}} & N'_1 & \to_{\beta_{naive}} & \dots & \to_{\beta_{naive}} & N'_k
\end{array}
$$

$\to_\beta$: ordinary β-step where we may (need to) apply α.

$\to_{\beta_{naive}}$: naïve β-step with naïve substitution (no α).

Question

Can α-conversion steps be avoided for a λ-term M, by suitably α-converting it up front, say to a term M' such that no α-conversion step needs to be invoked along any reduction from M'.

$$
\begin{array}{cccccccc}
 & M & \rightarrow_{\beta} & N_1 & \rightarrow_{\beta} & \dots & \rightarrow_{\beta} & N_k \\
\text{vaccination} & & & \equiv_{\alpha} & & & & \equiv_{\alpha} \\
 & M' & \rightarrow_{\beta_{naive}} & N'_1 & \rightarrow_{\beta_{naive}} & \dots & \rightarrow_{\beta_{naive}} & N'_k
\end{array}
$$

$\rightarrow_{\beta}$: ordinary β-step where we may (need to) apply α.

$\rightarrow_{\beta_{naive}}$: naïve β-step with naïve substitution (no α).

Variable capture

A naïve substitution leads to a variable capture whenever we:

Variable capture

A naïve substitution leads to a variable capture whenever we:

1. naïvely contract a redex $(\lambda x.M)\ N$ where

Variable capture

A naïve substitution leads to a variable capture whenever we:

1. naïvely contract a redex $(\lambda x.M)\ N$ where

2. where some variable y occurs free in N

Variable capture

A naïve substitution leads to a variable capture whenever we:

1. naïvely contract a redex $(\lambda x.M)\ N$ where
2. where some variable y occurs free in N
3. is moved into M, where some x that is free

Variable capture

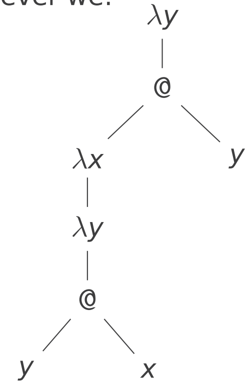
A naïve substitution leads to a variable capture whenever we:

1. naïvely contract a redex $(\lambda x.M)\ N$ where
2. where some variable y occurs free in N
3. is moved into M, where some x that is free
4. is in the scope of a λy

Variable capture

A naïve substitution leads to a variable capture whenever we:

1. naïvely contract a redex $(\lambda x.M)\, N$ where
2. where some variable y occurs free in N
3. is moved into M, where some x that is free
4. is in the scope of a λy

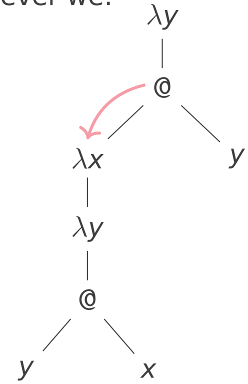


$AST(\lambda y.(\lambda xy.y\, x)\, y)$

Variable capture

A naïve substitution leads to a variable capture whenever we:

1. naïvely contract a redex $(\lambda x.M)\ N$ where (r-edge ⟶)
2. where some variable y occurs free in N
3. is moved into M, where some x that is free
4. is in the scope of a λy

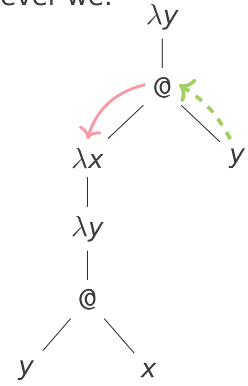


$AST(\lambda y.(\lambda xy.y\,x)\,y)$

Variable capture

A naïve substitution leads to a variable capture whenever we:

1. naïvely contract a redex $(\lambda x.M)\ N$ where (r-edge ⟶)
2. where some variable y occurs free in N (a-edge ⇢)
3. is moved into M, where some x that is free
4. is in the scope of a λy

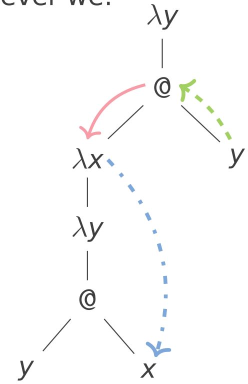


$AST(\lambda y.(\lambda xy.y\ x)\ y)$

Variable capture

A naïve substitution leads to a variable capture whenever we:

1. naïvely contract a redex $(\lambda x.M)\ N$ where (r-edge)
2. where some variable y occurs free in N (a-edge)
3. is moved into M, where some x that is free (b-edge)
4. is in the scope of a λy

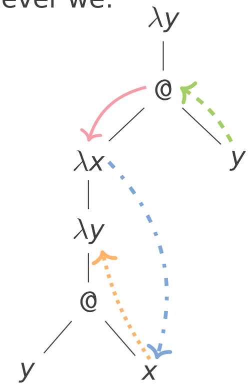


$AST(\lambda y.(\lambda xy.y\,x)\,y)$

Variable capture

A naïve substitution leads to a variable capture whenever we:

1. naïvely contract a redex $(\lambda x.M)\ N$ where (*r*-edge)
2. where some variable y occurs free in N (*a*-edge)
3. is moved into M, where some x that is free (*b*-edge)
4. is in the scope of a λy (*c*-edge)



$AST(\lambda y.(\lambda xy.y\ x)\ y)$

α via paths

***arbc* α-path**

$x \dashrightarrow @ \longrightarrow \lambda y \dashrightarrow y \dashrightarrow \lambda x$

α via paths

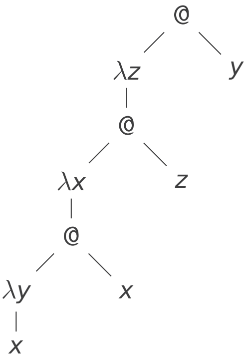
arbc α-path

$x \dashrightarrow @ \longrightarrow \lambda y \dashrightarrow y \dashrightarrow \lambda x$

α via paths

arbc **α-path**

$x \dashrightarrow @ \longrightarrow \lambda y \dashrightarrow y \dashrightarrow \lambda x$

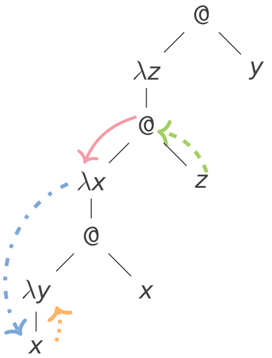


$(\lambda z.(\lambda x.(\lambda y.x)\,x)\,z)\,y$

α via paths

arbc α-path

$x \dashrightarrow @ \rightarrow \lambda y \dashrightarrow y \dashrightarrow \lambda x$

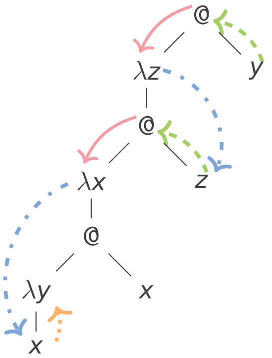


$$(\lambda z.(\lambda x.(\lambda y.x)\,x)\,z)\,y$$

α via paths

arbc α-path

$x \dashrightarrow @ \longrightarrow \lambda y \dashrightarrow y \dashrightarrow \lambda x$

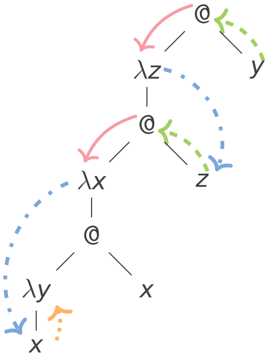


$(\lambda z.(\lambda x.(\lambda y.x)\,x)\,z)\,y$

α via paths

$(arb)^i c$ α-path

$(x \dashrightarrow @ \longrightarrow \lambda y \dashrightarrow y)^i \dashrightarrow \lambda x$

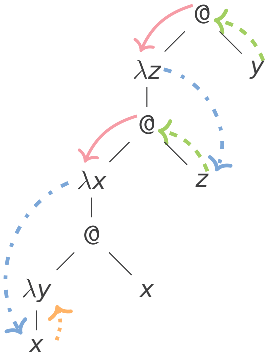


$\underline{(\lambda z.(\lambda x.(\lambda y.x)\,x)\,z)\,y}$

α via paths

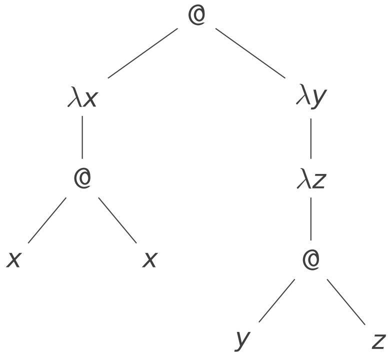
$(arb)^i c$ α-path

$(x \dashrightarrow @ \longrightarrow \lambda y \dashrightarrow y)^i \dashrightarrow \lambda x$



$$\underline{(\lambda z.(\lambda x.(\lambda y.x)\,x)\,z)\,y}$$
$$\to_\beta \underline{(\lambda x.(\lambda y.x)\,x)\,y}$$
$$\to_\beta \underline{(\lambda y'.y)\,x}$$
$$\to_\beta y$$

α via paths

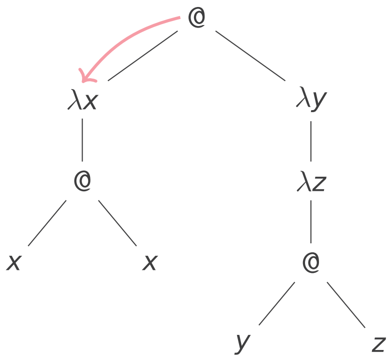


$(\lambda x.x\,x)\,(\lambda yz.y\,z)$

$(arb)^i c$ α-path

$(x \dashrightarrow @ \longrightarrow \lambda y \dashrightarrow y)^+ \dashrightarrow \lambda x$

α via paths

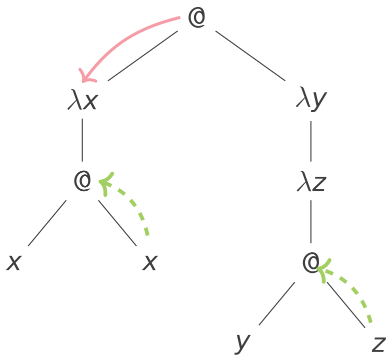


$(\lambda x.x\,x)\,(\lambda yz.y\,z)$

$(arb)^i c$ α-path

$(x \dashrightarrow @ \rightarrow \lambda y \dashrightarrow y)^+ \dashrightarrow \lambda x$

α via paths

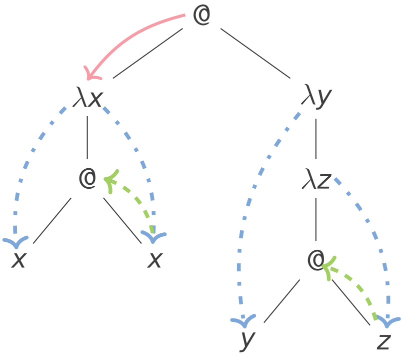


$(\lambda x.x\,x)\,(\lambda yz.y\,z)$

$(arb)^i c$ α-path

$(x \dashrightarrow @ \longrightarrow \lambda y \dashrightarrow y\,)^+ \dashrightarrow \lambda x$

α via paths

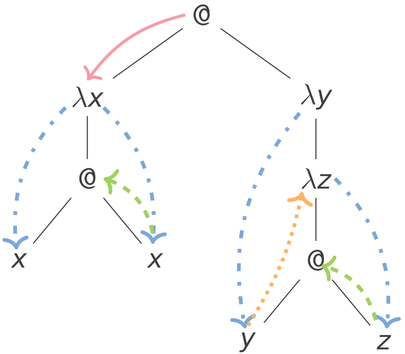


$(\lambda x.x\,x)\,(\lambda yz.y\,z)$

$(arb)^i c$ α-path

$(x \dashrightarrow @ \longrightarrow \lambda y \dashrightarrow y\,)^+ \dashrightarrow \lambda x$

α via paths

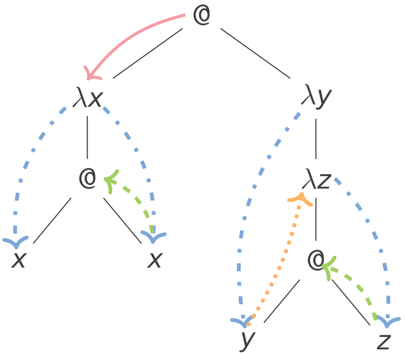


$(\lambda x.x\,x)\,(\lambda yz.y\,z)$

$(arb)^i c$ α-path

$(x \dashrightarrow @ \longrightarrow \lambda y \dashrightarrow y\,)^+ \dashrightarrow \lambda x$

α via paths

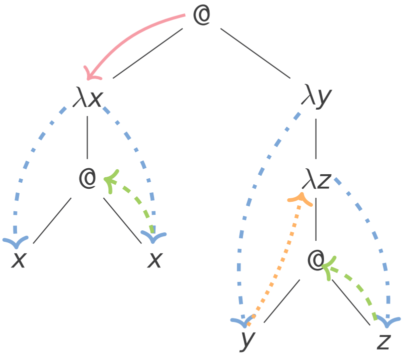


$$\underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)}$$
$$\to_\beta \quad (\lambda yz.y\,z)\,(\lambda yz.y\,z)$$

$(arb)^i c$ α-path

$(x \dashrightarrow @ \longrightarrow \lambda y \dashrightarrow y\,)^+ \dashrightarrow \lambda x$

α via paths



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

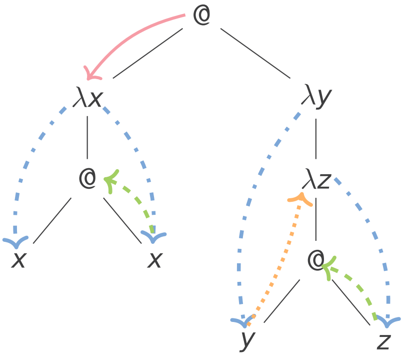
$$\to_\beta\quad (\lambda yz.y\,z)\,(\lambda yz.y\,z)$$

$$\to_\beta\quad \lambda z.(\lambda yz.y\,z)\,z$$

$(arb)^i c$ α-path

$(x \dashrightarrow @ \longrightarrow \lambda y \dashrightarrow y\,)^+ \dashrightarrow \lambda x$

α via paths

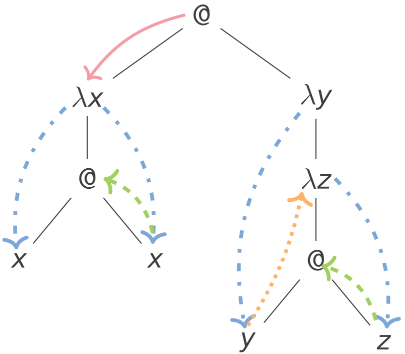


$$
\begin{aligned}
& \underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)} \\
\to_\beta\; & \underline{(\lambda yz.y\,z)\,(\lambda yz.y\,z)} \\
\to_\beta\; & \lambda z.\underline{(\lambda yz.y\,z)\,z} \\
\to_\beta\; & \lambda z.(\lambda z'.z\,z')
\end{aligned}
$$

$(arb)^i c$ α-path

$(x \dashrightarrow @ \rightarrow \lambda y \dashrightarrow y\,)^+ \dashrightarrow \lambda x$

α via paths



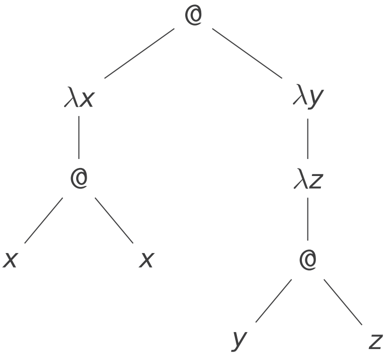
$$\underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)}$$
$$\to_\beta \quad \underline{(\lambda yz.y\,z)\,(\lambda yz.y\,z)}$$
$$\to_\beta \quad \lambda z.\underline{(\lambda yz.y\,z)\,z}$$
$$\to_\beta \quad \lambda z.(\lambda z'.z\,z')$$

need characterisation of created redexes

$(arb)^i c$ α-path

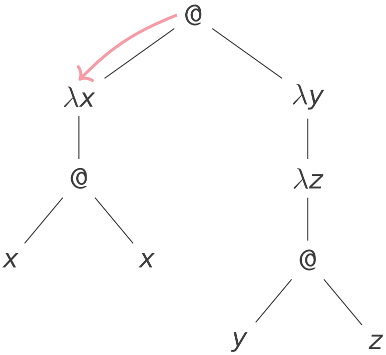
$(x \dashrightarrow @ \longrightarrow \lambda y \dashrightarrow y\,)^+ \dashrightarrow \lambda x$

Created redexes



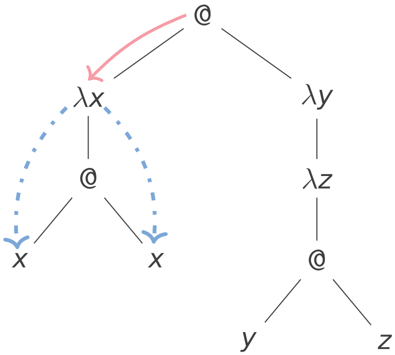
$$(\lambda x.x\,x)\,(\lambda yz.y\,z)$$

Created redexes



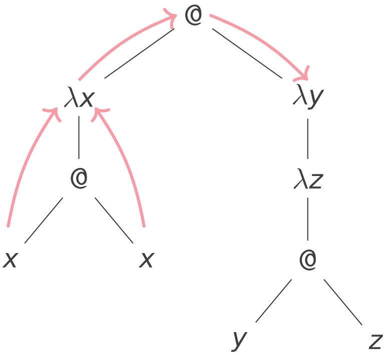
$\underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)}$

Created redexes



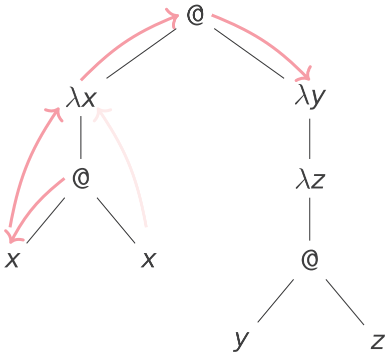
$\underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)}$

Created redexes



$$\underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)}$$
$$\to_\beta \quad (\lambda yz.y\,z)\,(\lambda yz.y\,z)$$

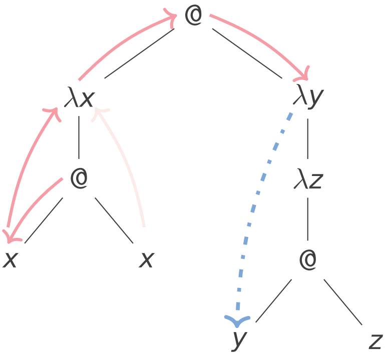
Created redexes



$$\underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)}$$

$$\to_\beta \quad \underline{(\lambda yz.y\,z)\,(\lambda yz.y\,z)}$$

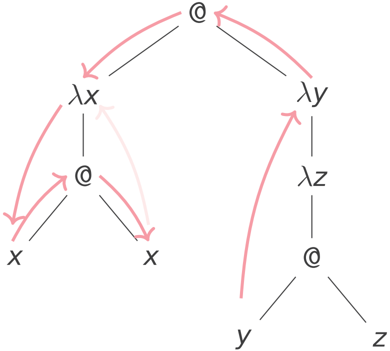
Created redexes



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

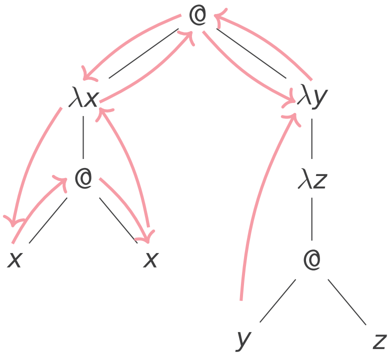
$$\to_\beta \quad (\lambda yz.y\,z)\,(\lambda yz.y\,z)$$

Created redexes



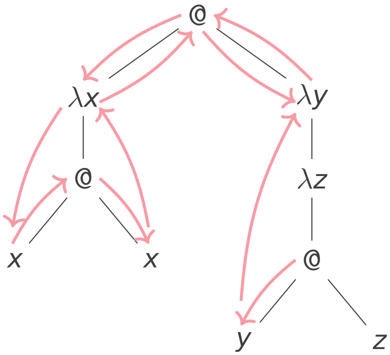
$$(\lambda x.x\,x)\,(\lambda yz.y\,z)$$
$$\to_\beta\ (\lambda yz.y\,z)\,(\lambda yz.y\,z)$$
$$\to_\beta\ \lambda z.(\lambda yz.y\,z)\,z$$

Created redexes



$$\underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)}$$
$$\to_\beta\ \underline{(\lambda yz.y\,z)\,(\lambda yz.y\,z)}$$
$$\to_\beta\ \lambda z.(\lambda yz.y\,z)\,z$$

Created redexes

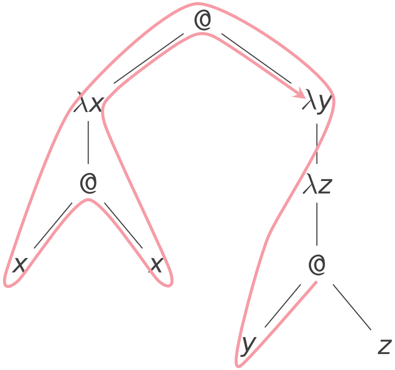


$$\underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)}$$

$$\to_\beta \quad \underline{(\lambda yz.y\,z)\,(\lambda yz.y\,z)}$$

$$\to_\beta \quad \lambda z.\underline{(\lambda yz.y\,z)\,z}$$

Created redexes

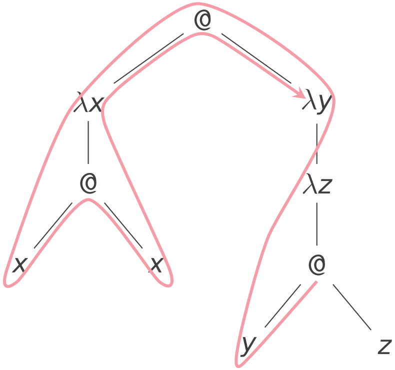


$$\underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)}$$
$$\to_\beta \underline{(\lambda yz.y\,z)\,(\lambda yz.y\,z)}$$
$$\to_\beta \lambda z.\underline{(\lambda yz.y\,z)\,z}$$

Legal paths (Asperti et al. 1994)

Characterise virtual redexes.

Created redexes



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

$$\rightarrow_\beta \quad (\lambda yz.y\,z)\,(\lambda yz.y\,z)$$

$$\rightarrow_\beta \quad \lambda z.(\lambda yz.y\,z)\,z$$

$$\rightarrow_\beta \quad \lambda z.(\lambda z'.z\,z')$$

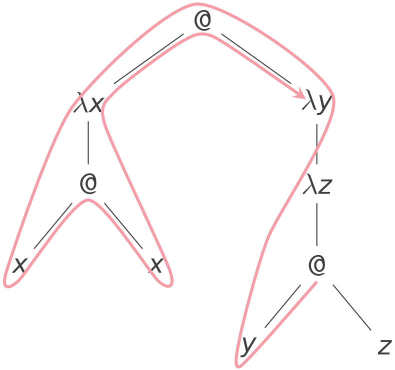
Legal paths (Asperti et al. 1994)

Characterise virtual redexes.

Overview

1. **Motivation**
2. **α-Paths**
3. **α-Avoidance in different calculi**
4. **Soundness and Undecidability**
5. **Conclusion and Future Work**

α-Paths

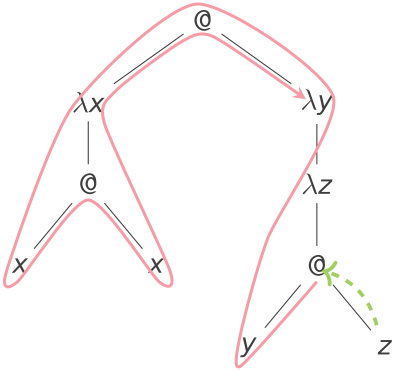


$$\underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)}$$
$$\to_\beta \underline{(\lambda yz.y\,z)\,(\lambda yz.y\,z)}$$
$$\to_\beta \lambda z.\underline{(\lambda yz.y\,z)\,z}$$
$$\to_\beta \lambda z.(\lambda z'.z\,z')$$

Combining a-, b- and c-edges with legal paths $\Longrightarrow$ $(a|b)^i c$ α-path

Allows the prediction of the potential need for α.

α-Paths

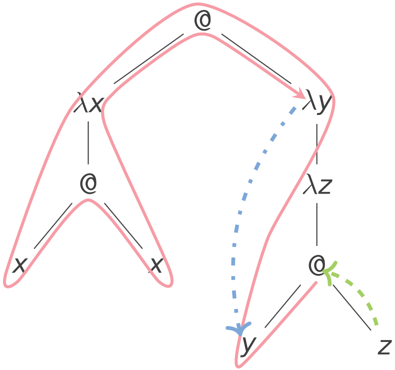


$$\underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)}$$
$$\to_\beta \quad \underline{(\lambda yz.y\,z)\,(\lambda yz.y\,z)}$$
$$\to_\beta \quad \lambda z.\underline{(\lambda yz.y\,z)\,z}$$
$$\to_\beta \quad \lambda z.(\lambda z'.z\,z')$$

Combining *a*-, *b*- and *c*-edges with legal paths $\implies$ $(a|b)^i c$ α-path

Allows the prediction of the potential need for α.

α-Paths

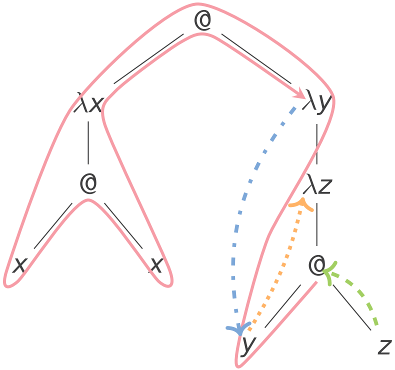


$$\underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)}$$
$$\rightarrow_\beta \quad \underline{(\lambda yz.y\,z)\,(\lambda yz.y\,z)}$$
$$\rightarrow_\beta \quad \lambda z.\underline{(\lambda yz.y\,z)\,z}$$
$$\rightarrow_\beta \quad \lambda z.(\lambda z'.z\,z')$$

Combining a-, b- and c-edges with legal paths $\implies$ $(a|b)^i c$ α-path

Allows the prediction of the potential need for α.

α-Paths

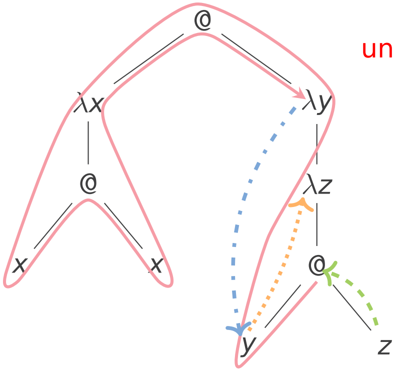


$$\underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)}$$
$$\rightarrow_\beta \quad \underline{(\lambda yz.y\,z)\,(\lambda yz.y\,z)}$$
$$\rightarrow_\beta \quad \lambda z.\underline{(\lambda yz.y\,z)\,z}$$
$$\rightarrow_\beta \quad \lambda z.(\lambda z'.z\,z')$$

Combining *a*-, *b*- and *c*-edges with legal paths $\implies$ $(a|b)^i c$ α-path

Allows the prediction of the potential need for α.

α-Paths



unremovable

$$\underline{(\lambda x.x\,x)\,(\lambda yz.y\,z)}$$
$$\to_\beta \quad \underline{(\lambda yz.y\,z)\,(\lambda yz.y\,z)}$$
$$\to_\beta \quad \lambda z.\underline{(\lambda yz.y\,z)\,z}$$
$$\to_\beta \quad \lambda z.(\lambda z'.z\,z')$$

Combining a-, b- and c-edges with legal paths $\Longrightarrow$ $(a|b)^i c$ α-path

Allows the prediction of the potential need for α.

α-Avoidance

Question

Can α-conversion steps be avoided for a λ-term M, by suitably α-converting it up front, say to a term M' such that no α-conversion step needs to be invoked along any reduction from M'.

$$\begin{array}{ccccccc} M & \to_\beta & N_1 & \to_\beta & \ldots & \to_\beta & N_k \\ \equiv_\alpha & & \equiv_\alpha & & & & \equiv_\alpha \\ M' & \to_{\beta_{naive}} & N'_1 & \to_{\beta_{naive}} & \ldots & \to_{\beta_{naive}} & N'_k \end{array}$$

$\to_\beta$: ordinary β-step where we may (need to) apply α.

$\to_{\beta_{naive}}$: naïve β-step with naïve substitution (no α).

α-Avoidance

Question

Can α-conversion steps be avoided for a λ-term M, by suitably α-converting it up front, say to a term M' such that no α-conversion step needs to be invoked along any reduction from M'.

$$\begin{array}{lccccccc}
\alpha\text{-paths} & M & \to_\beta & N_1 & \to_\beta & \cdots & \to_\beta & N_k \\
 & \equiv_\alpha & & \equiv_\alpha & & & & \equiv_\alpha \\
\text{no } \alpha\text{-paths} & M' & \to_{\beta_{naive}} & N'_1 & \to_{\beta_{naive}} & \cdots & \to_{\beta_{naive}} & N'_k
\end{array}$$

$\to_\beta$: ordinary β-step where we may (need to) apply α.

$\to_{\beta_{naive}}$: naïve β-step with naïve substitution (no α).

Overview

1. Motivation
2. α-Paths
3. **α-Avoidance in different calculi**
4. Soundness and Undecidability
5. Conclusion and Future Work

α-Avoidance in different calculi

λ-calculus

α is unavoidable

$$
\begin{aligned}
& \underline{(\lambda x.x\ x)\ (\lambda y \lambda z.y\ z)} \\
\to_\beta\ & \underline{(\lambda y \lambda z.yz)\ (\lambda y \lambda z.y\ z)} \\
\to_\beta\ & \lambda z.\underline{(\lambda y \lambda z.y\ z)\ z} \\
\to_\alpha\ & \lambda z.\underline{(\lambda y.\lambda z'.y\ z')\ z} \\
\to_\beta\ & \lambda z \lambda z'.z\ z'
\end{aligned}
$$

α-Avoidance in different calculi

λ-calculus

α is unavoidable

$$
\begin{aligned}
& \underline{(\lambda x.x\ x)\ (\lambda y \lambda z.y\ z)} && \text{duplication} \\
\to_\beta\ & \underline{(\lambda y \lambda z.yz)\ (\lambda y \lambda z.y\ z)} \\
\to_\beta\ & \lambda z.\underline{(\lambda y \lambda z.y\ z)\ z} \\
\to_\alpha\ & \lambda z.\underline{(\lambda y.\lambda z'.y\ z')\ z} \\
\to_\beta\ & \lambda z \lambda z'.z\ z'
\end{aligned}
$$

α-Avoidance in different calculi

λ-calculus

α is unavoidable

$$
\begin{aligned}
& \underline{(\lambda x.x\ x)\ (\lambda y \lambda z.y\ z)} && \text{duplication} \\
\to_\beta\ & \underline{(\lambda y \lambda z.yz)\ (\lambda y \lambda z.y\ z)} && \text{redex creation} \\
\to_\beta\ & \lambda z.\underline{(\lambda y \lambda z.y\ z)\ z} \\
\to_\alpha\ & \lambda z.\underline{(\lambda y.\lambda z'.y\ z')\ z} \\
\to_\beta\ & \lambda z \lambda z'.z\ z'
\end{aligned}
$$

α-Avoidance in different calculi

λ-calculus

α is unavoidable

$$
\begin{aligned}
& \underline{(\lambda x.x\ x)\ (\lambda y \lambda z.y\ z)} && \text{duplication} \\
\to_\beta\ & \underline{(\lambda y \lambda z.yz)\ (\lambda y \lambda z.y\ z)} && \text{redex creation} \\
\to_\beta\ & \lambda z.\underline{(\lambda y \lambda z.y\ z)\ z} && \text{open redex contraction} \\
\to_\alpha\ & \lambda z.\underline{(\lambda y.\lambda z'.y\ z')\ z} \\
\to_\beta\ & \lambda z \lambda z'.z\ z'
\end{aligned}
$$

α-Avoidance in different calculi

λ-calculus

α is unavoidable

$$
\begin{array}{ll}
 & \underline{(\lambda x.x\ x)\ (\lambda y \lambda z.y\ z)} \\
\to_\beta & \underline{(\lambda y \lambda z.yz)\ (\lambda y \lambda z.y\ z)} \\
\to_\beta & \lambda z.\underline{(\lambda y \lambda z.y\ z)\ z} \\
\to_\alpha & \lambda z.\underline{(\lambda y.\lambda z'.y\ z')\ z} \\
\to_\beta & \lambda z \lambda z'.z\ z'
\end{array}
$$

- duplication
- redex creation
- open redex contraction

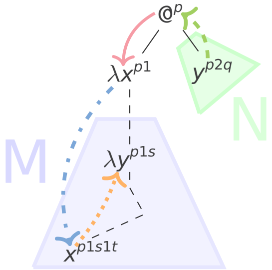
3 phenomena causing α – absence of each allows to avoid α

i.e. we always can α-rename up front such that no α-paths occur

α-Avoidance in different calculi

Developments are α-avoiding (Church and Rosser 1936)

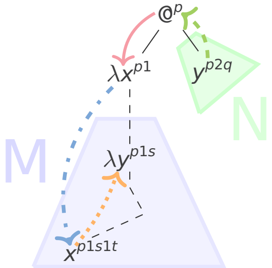
No redex creation (r-edges are enough)



α-Avoidance in different calculi

Developments are α-avoiding (Church and Rosser 1936)

No redex creation (r-edges are enough)



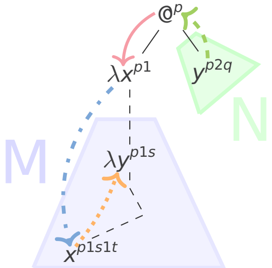
Final λx-node at the left of the starting variable x

$(x^{p2q} \dashrightarrow @^{p} \longrightarrow \lambda y^{p1} \dashrightarrow y^{p1s1t})^{+} \dashrightarrow \lambda x^{p1s}$

α-Avoidance in different calculi

Developments are α-avoiding (Church and Rosser 1936)

No redex creation (r-edges are enough)



$\Longrightarrow$ no unremovable α-paths

Final λx-node at the left of the starting variable x

$(x^{p2q} \dashrightarrow @^{p} \longrightarrow \lambda y^{p1} \dashrightarrow y^{p1s1t})^{+} \dashrightarrow \lambda x^{p1s}$

α-Avoidance in different calculi

The linear (affine) λ-calculus (Hindley 1989)

Forbids duplication by restricting term-formation

α-Avoidance in different calculi

The linear (affine) λ-calculus (Hindley 1989)

Forbids duplication by restricting term-formation

Lemma 19.

Let M be a linear λ-term, $M \to_\beta N$ and $q \prec p$ for some positions p, q in M. If $p \blacktriangleright p'$ and $q \blacktriangleright q'$, then $q' \prec p'$.

α-Avoidance in different calculi

The linear (affine) λ-calculus (Hindley 1989)

Forbids duplication by restricting term-formation

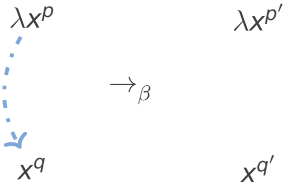
Lemma 19.

Let M be a linear λ-term, $M \to_\beta N$ and $q \prec p$ for some positions p, q in M. If $p \blacktriangleright p'$ and $q \blacktriangleright q'$, then $q' \prec p'$.

α-Avoidance in different calculi

The linear (affine) λ-calculus (Hindley 1989)

Forbids duplication by restricting term-formation



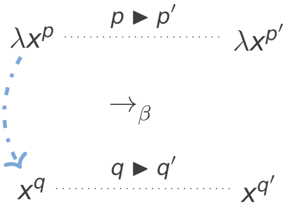
Lemma 19.

Let M be a linear λ-term, $M \to_\beta N$ and $q \prec p$ for some positions p, q in M. If $p \blacktriangleright p'$ and $q \blacktriangleright q'$, then $q' \prec p'$.

α-Avoidance in different calculi

The linear (affine) λ-calculus (Hindley 1989)

Forbids duplication by restricting term-formation



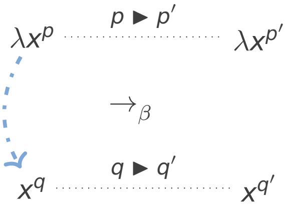
Lemma 19.

Let M be a linear λ-term, $M \to_\beta N$ and $q \prec p$ for some positions p, q in M. If $p \blacktriangleright p'$ and $q \blacktriangleright q'$, then $q' \prec p'$.

α-Avoidance in different calculi

The linear (affine) λ-calculus (Hindley 1989)

Forbids duplication by restricting term-formation



context	$p \blacktriangleright p$	if o is not prefix of p
body	$o11p \blacktriangleright op$	if $p \neq \epsilon$ and $p \neq q$
arg	$o2p \blacktriangleright oqp$	for all positions q, such that $o11q$ is bound by $o1$

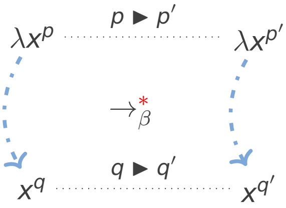
Lemma 19.

Let M be a linear λ-term, $M \to_\beta N$ and $q \prec p$ for some positions p, q in M. If $p \blacktriangleright p'$ and $q \blacktriangleright q'$, then $q' \prec p'$.

α-Avoidance in different calculi

The linear (affine) λ-calculus (Hindley 1989)

Forbids duplication by restricting term-formation



context	$p \blacktriangleright p$	if o is not prefix of p
body	$o11p \blacktriangleright op$	if $p \neq \epsilon$ and $p \neq q$
arg	$o2p \blacktriangleright oqp$	for all positions q, such that $o11q$ is bound by $o1$

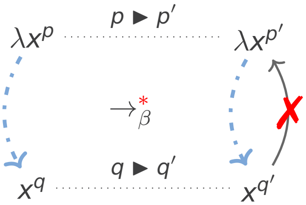
Lemma 19.

Let M be a linear λ-term, $M \to_\beta N$ and $q \prec p$ for some positions p, q in M. If $p \blacktriangleright p'$ and $q \blacktriangleright q'$, then $q' \prec p'$.

α-Avoidance in different calculi

The linear (affine) λ-calculus (Hindley 1989)

Forbids duplication by restricting term-formation



context	$p \blacktriangleright p$	if o is not prefix of p
body	$o11p \blacktriangleright op$	if $p \neq \epsilon$ and $p \neq q$
arg	$o2p \blacktriangleright oqp$	for all positions q, such that $o11q$ is bound by $o1$

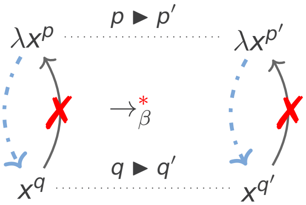
Lemma 19.

Let M be a linear λ-term, $M \to_\beta N$ and $q \prec p$ for some positions p, q in M. If $p \blacktriangleright p'$ and $q \blacktriangleright q'$, then $q' \prec p'$.

α-Avoidance in different calculi

The linear (affine) λ-calculus (Hindley 1989)

Forbids duplication by restricting term-formation



context	$p \blacktriangleright p$	if o is not prefix of p
body	$o11p \blacktriangleright op$	if $p \neq \epsilon$ and $p \neq q$
arg	$o2p \blacktriangleright oqp$	for all positions q, such that $o11q$ is bound by $o1$

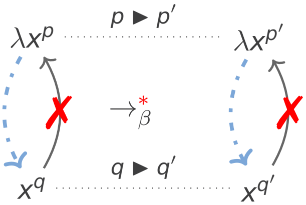
Lemma 19.

Let M be a linear λ-term, $M \to_\beta N$ and $q \prec p$ for some positions p, q in M. If $p \blacktriangleright p'$ and $q \blacktriangleright q'$, then $q' \prec p'$.

α-Avoidance in different calculi

The linear (affine) λ-calculus (Hindley 1989)

Forbids duplication by restricting term-formation



context	$p \blacktriangleright p$	if o is not prefix of p
body	$o11p \blacktriangleright op$	if $p \neq \epsilon$ and $p \neq q$
arg	$o2p \blacktriangleright oqp$	for all positions q, such that $o11q$ is bound by $o1$

$\Longrightarrow$ no unremovable α-paths

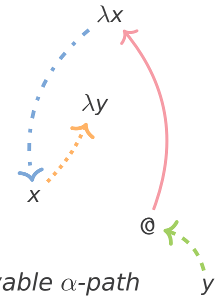
Lemma 19.

Let M be a linear λ-term, $M \to_\beta N$ and $q \prec p$ for some positions p, q in M. If $p \blacktriangleright p'$ and $q \blacktriangleright q'$, then $q' \prec p'$.

α-Avoidance in different calculi

The weak λ-calculus (Çağman and Hindley 1998)

Forbids to contract open redexes

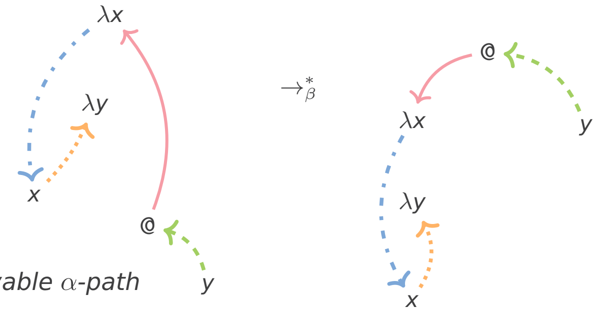


a term with an unremovable α-path

α-Avoidance in different calculi

The weak λ-calculus (Çağman and Hindley 1998)

Forbids to contract open redexes



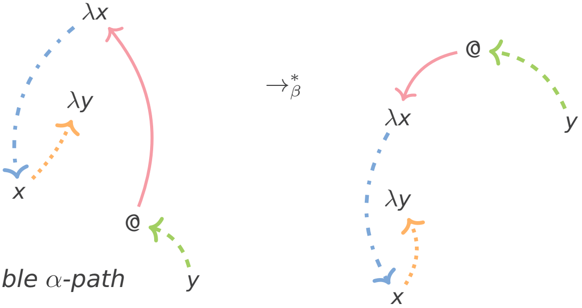
a term with an unremovable α-path

α-Avoidance in different calculi

The weak λ-calculus (Çağman and Hindley 1998)

Forbids to contract open redexes

“bound variables are never relased”



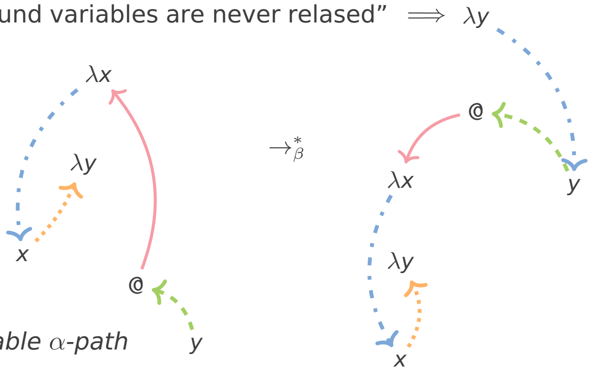
a term with an unremovable α-path

α-Avoidance in different calculi

The weak λ-calculus (Çağman and Hindley 1998)

Forbids to contract open redexes

"bound variables are never relased" $\Longrightarrow$



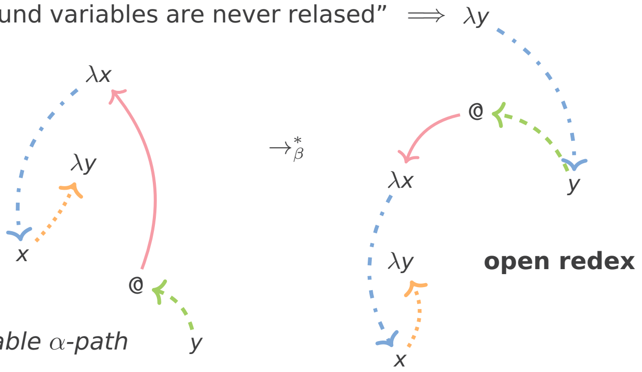
a term with an unremovable α-path

α-Avoidance in different calculi

The weak λ-calculus (Çağman and Hindley 1998)

Forbids to contract open redexes

“bound variables are never relased” $\Longrightarrow$



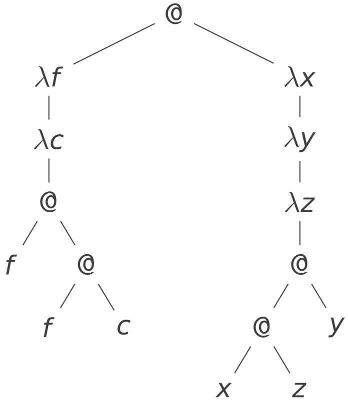
a term with an unremovable α-path

α-Avoidance in different calculi

The simply-typed λ-calculus à la Church

α is unavoidable

$$(\lambda f^{\tau \to \tau} x^{\sigma} . f\,(f\,x))\,(\lambda x^{\tau} y^{\sigma} z^{\sigma} . x\,z\,y)$$

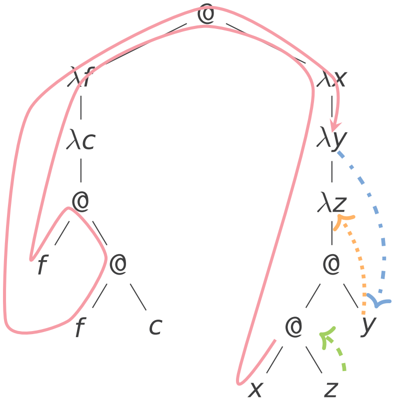


$\sigma = o, \tau = o \to o \to o$

The simply-typed λ-calculus à la Church

α is unavoidable

$$(\lambda f^{\tau \to \tau} x^{\sigma}.f\,(f\,x))\,(\lambda x^{\tau} y^{\sigma} z^{\sigma}.x\,z\,y)$$



$\sigma = o, \tau = o \to o \to o$

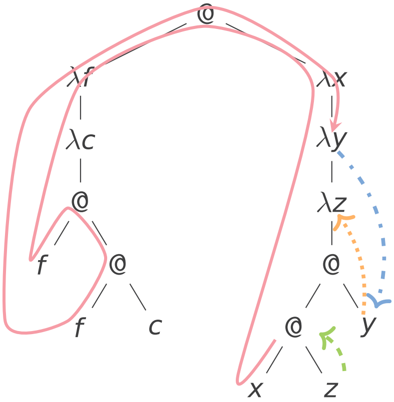
α-Avoidance in different calculi

The simply-typed λ-calculus à la Church

α is unavoidable

$$(\lambda f^{\tau\to\tau} x^{\sigma}.f\,(f\,x))\,(\lambda x^{\tau} y^{\sigma} z^{\sigma}.x\,z\,y)$$
$$\to_\beta \quad \lambda x.(\lambda xyz.x\,z\,y)\,((\lambda xyz.x\,z\,y)\,x)$$
$$\to_\beta \quad \lambda x.(\lambda xyz.x\,z\,y)\,(\lambda yz.x\,z\,y)$$
$$\to_\beta \quad \lambda xyz.(\lambda yz.x\,z\,y)\,z\,y$$
$$\to_\alpha \quad \lambda xyz.(\lambda yz'.x\,z'\,y)\,z\,y$$
$$\to_\beta \quad \lambda xyz.(\lambda z'.x\,z'\,z)\,y$$
$$\to_\beta \quad \lambda xyz.x\,y\,z$$

$\sigma = o, \tau = o \to o \to o$



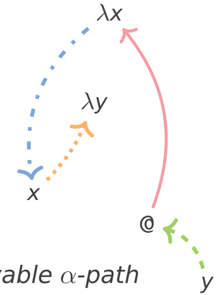
α-Avoidance in different calculi

The safe λ-calculus (Blum and Ong 2007)

safety: if $x \in \mathcal{F}V(M)$, then $ord\, M \leq ord\, x$.

$$ord\, o := 0$$

$$ord\, \sigma \to \tau := max(1 + ord\, \sigma, ord\, \tau)$$



a term with an unremovable α-path

α-Avoidance in different calculi

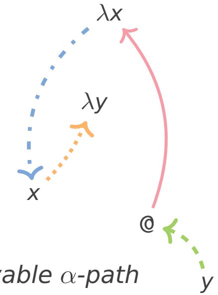
The safe λ-calculus (Blum and Ong 2007)

safety: if $x \in \mathcal{FV}(M)$, then $ord\, M \leq ord\, x$.

$ord\, o := 0$

$ord\, \sigma \to \tau := max(1 + ord\, \sigma, ord\, \tau)$

$ord\, y \geq ord\, x$



a term with an unremovable α-path

α-Avoidance in different calculi

The safe λ-calculus (Blum and Ong 2007)

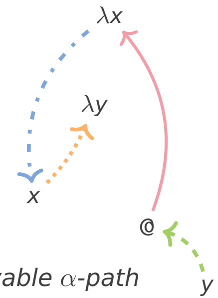
safety: if $x \in \mathcal{FV}(M)$, then $ord\, M \leq ord\, x$.

$ord\, o := 0$

$ord\, \sigma \to \tau := max(1 + ord\, \sigma, ord\, \tau)$

$ord\, y \geq ord\, x$

$ord\, (\lambda y.t) \leq ord\, x$



a term with an unremovable α-path

α-Avoidance in different calculi

The safe λ-calculus (Blum and Ong 2007)

safety: if $x \in \mathcal{FV}(M)$, then $ord\,M \leq ord\,x$.

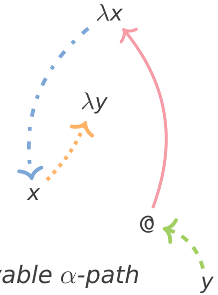
$ord\,o := 0$

$ord\,\sigma \to \tau := max(1 + ord\,\sigma, ord\,\tau)$

$ord\,y \geq ord\,x$

$ord\,(\lambda y.t) \leq ord\,x$

$ord\,y < ord\,x$



a term with an unremovable α-path

α-Avoidance in different calculi

The safe λ-calculus (Blum and Ong 2007)

safety: if $x \in \mathcal{FV}(M)$, then $ord\, M \leq ord\, x$.

$ord\, o := 0$

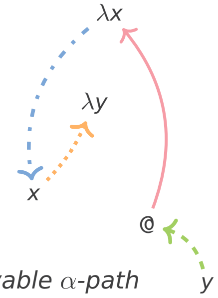
$ord\, \sigma \to \tau := max(1 + ord\, \sigma, ord\, \tau)$

$ord\, y \geq ord\, x$

$ord\, (\lambda y.t) \leq ord\, x$

$ord\, y < ord\, x$

↯ $\implies$ no α-paths



a term with an unremovable α-path

α-Avoidance in different calculi

The safe λ-calculus (Blum and Ong 2007)

safety: if $x \in \mathcal{FV}(M)$, then $ord\, M \leq ord\, x$.

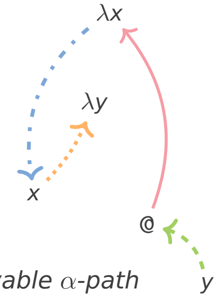
$ord\, o := 0$

$ord\, \sigma \to \tau := max(1 + ord\, \sigma, ord\, \tau)$

$ord\, y \geq ord\, x$

$ord\, (\lambda y.t) \leq ord\, x$

$ord\, y < ord\, x$



⚡ $\implies$ no α-paths

a term with an unremovable α-path

$\to$ analysing the safe λ-calculus as presented in (Blum and Ong 2009) using our tools, we found that the claim that α could be avoided in it, was not entirely correct.

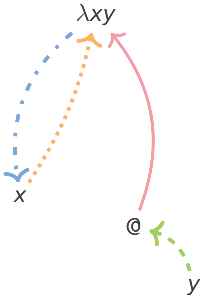
α-Avoidance in different calculi

The safe λ-calculus (Blum and Ong 2007)

safety + combined abstractions and simultaneous substitution.

$$ord\, o := 0$$

$$ord\, \sigma \to \tau := max(1 + ord\, \sigma, ord\, \tau)$$



α-Avoidance in different calculi

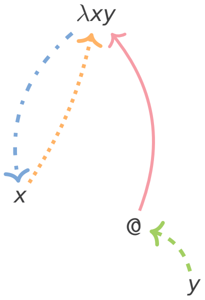
The safe λ-calculus (Blum and Ong 2007)

safety $+$ combined abstractions and simultaneous substitution.

$$ord\, o := 0$$

$$ord\, \sigma \to \tau := max(1 + ord\, \sigma, ord\, \tau)$$

$$ord\, y \geq ord\, x$$



α-Avoidance in different calculi

The safe λ-calculus (Blum and Ong 2007)

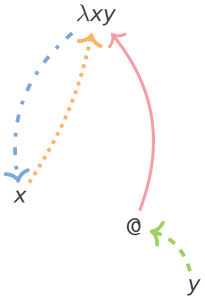
safety $+$ combined abstractions and simultaneous substitution.

$$\mathit{ord}\, o := 0$$

$$\mathit{ord}\, \sigma \to \tau := \mathit{max}(1 + \mathit{ord}\, \sigma, \mathit{ord}\, \tau)$$

$$\mathit{ord}\, y \geq \mathit{ord}\, x$$

$$\mathit{ord}\, y < \mathit{ord}\, x$$



α-Avoidance in different calculi

The safe λ-calculus (Blum and Ong 2007)

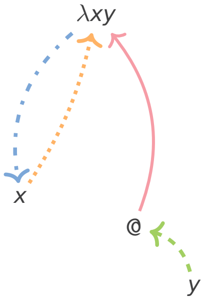
safety + combined abstractions and simultaneous substitution.

$\mathit{ord}\, o := 0$

$\mathit{ord}\, \sigma \to \tau := \mathit{max}(1 + \mathit{ord}\, \sigma, \mathit{ord}\, \tau)$

$\mathit{ord}\, y \geq \mathit{ord}\, x$

$\mathit{ord}\, y \not< \mathit{ord}\, x$ ✗



α-Avoidance in different calculi

The safe λ-calculus (Blum and Ong 2007)

safety + combined abstractions and simultaneous substitution.

$ord\, o := 0$

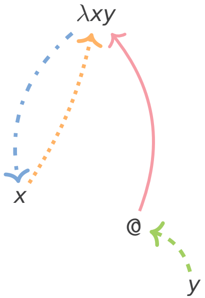
$ord\, \sigma \to \tau := max(1 + ord\, \sigma, ord\, \tau)$

$ord\, y \geq ord\, x$

$ord\, y$ ✗ $ord\, x$

$\Longrightarrow$ cannot exclude variable capture

a more restrictive system needed



Overview

1. **Motivation**
2. **α-Paths**
3. **α-Avoidance in different calculi**
4. **Soundness and Undecidability**
5. **Conclusion and Future Work**

Soundness, but not completeness

variable capture $\implies$ α-path

Proven in our paper □

Soundness, but not completeness

variable capture $\implies$ α-path

Proven in our paper $\square$

α-path $\not\Longrightarrow$ variable capture

$(\lambda x.x\,x)\,(\lambda yx.y\,z)$ is α-free

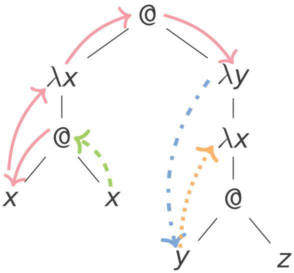
Soundness, but not completeness

variable capture $\implies$ α-path

Proven in our paper □

α-path $\not\Longrightarrow$ variable capture

$(\lambda x.x\,x)\,(\lambda yx.y\,z)$ is α-free



$$
\begin{aligned}
& \underline{(\lambda x.x\,x)\,(\lambda yx.y\,z)} \\
\to_\beta\; & \underline{(\lambda yx.y\,z)\,(\lambda yx.y\,z)} \\
\to_\beta\; & \lambda x.\underline{(\lambda yx.y\,z)\,z} \\
\to_\beta\; & \lambda x.(\lambda x.z\,z)
\end{aligned}
$$

Undecidability

Reduction from Post's correspondence problem (Post 1946)

α-avoidance is undecidable for the leftmost–outermost reduction strategy.

$$(PCP\ PAIRS\ AA\ BB)\ (\lambda xyz.(x\,z)\,y)$$

$AA \ldots$ encoding of string "aa" $\quad BB \ldots$ encoding of string "bb"

Undecidability

Reduction from Post's correspondence problem (Post 1946)

α-avoidance is undecidable for the leftmost–outermost reduction strategy.

$$(PCP\ PAIRS\ AA\ BB)\ (\lambda xyz.(x\,z)\,y)$$

solution: $AA\ (\lambda xyz.(x\,z)\,y)$

no solution: $BB\ (\lambda xyz.(x\,z)\,y)$

AA … encoding of string "aa" BB … encoding of string "bb"

Undecidability

Reduction from Post's correspondence problem (Post 1946)

α-avoidance is undecidable for the leftmost–outermost reduction strategy.

$$(PCP\ PAIRS\ AA\ BB)\ (\lambda xyz.(x\,z)\,y)$$

solution

$$AA\ (\lambda xyz.(x\,z)\,y)$$

$$(\lambda abx.a\,(a\,x))\ (\lambda xyz.(x\,z)\,y)$$

no solution

$$BB\ (\lambda xyz.(x\,z)\,y)$$

AA . . . encoding of string "aa" BB . . . encoding of string "bb"

Undecidability

Reduction from Post's correspondence problem (Post 1946)

α-avoidance is undecidable for the leftmost–outermost reduction strategy.

$(PCP\ PAIRS\ AA\ BB)\ (\lambda xyz.(x\,z)\,y)$

solution

no solution

$AA\ (\lambda xyz.(x\,z)\,y)$

$BB\ (\lambda xyz.(x\,z)\,y)$

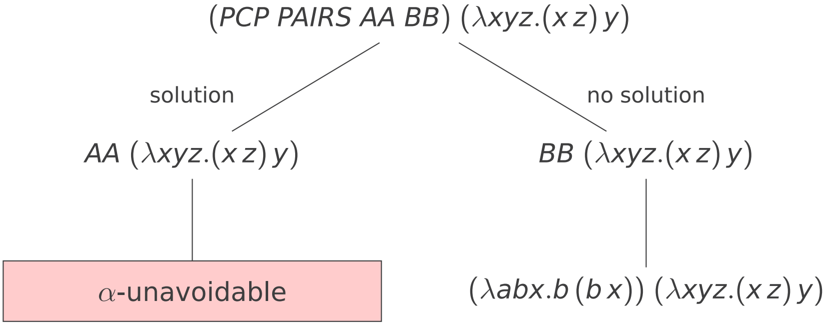
α-unavoidable

AA . . . encoding of string "aa" BB . . . encoding of string "bb"

Undecidability

Reduction from Post's correspondence problem (Post 1946)

α-avoidance is undecidable for the leftmost–outermost reduction strategy.



AA . . . encoding of string "aa" *BB* . . . encoding of string "bb"

Undecidability

Reduction from Post's correspondence problem (Post 1946)

α-avoidance is undecidable for the leftmost–outermost reduction strategy.

$(PCP\ PAIRS\ AA\ BB)\ (\lambda xyz.(x\,z)\,y)$

solution

$AA\ (\lambda xyz.(x\,z)\,y)$

α-unavoidable

no solution

$BB\ (\lambda xyz.(x\,z)\,y)$

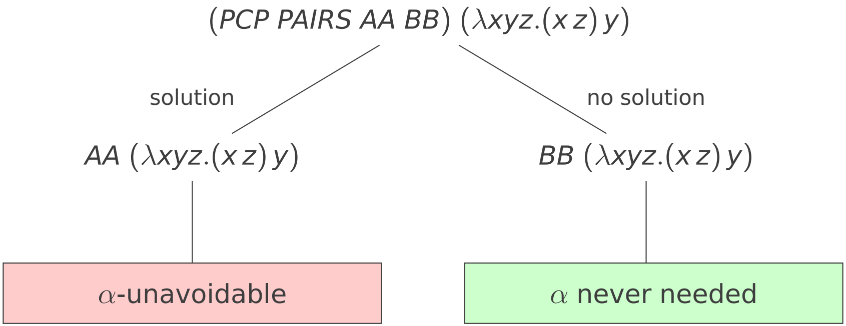
$\lambda bx.b\,(b\,x)$

$AA \ldots$ encoding of string "aa" $BB \ldots$ encoding of string "bb"

Undecidability

Reduction from Post's correspondence problem (Post 1946)

α-avoidance is undecidable for the leftmost–outermost reduction strategy.



AA . . . encoding of string "aa"   *BB* . . . encoding of string "bb"

Overview

1. **Motivation**
2. **α-Paths**
3. **α-Avoidance in different calculi**
4. **Soundness and Undecidability**
5. **Conclusion and Future Work**

α-Avoidance – Tool

Alpha Avoidance

Untyped Term:

```
(/x.x x) (/y z.y z)
```

```
UT = (UT) | /x ... y.UT | UT ... UT | x
```

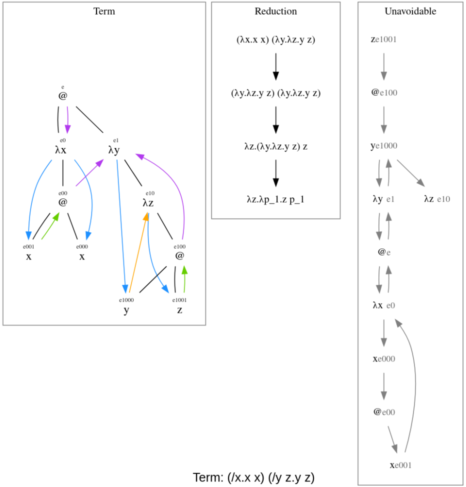
Max depth:

16

Analyze

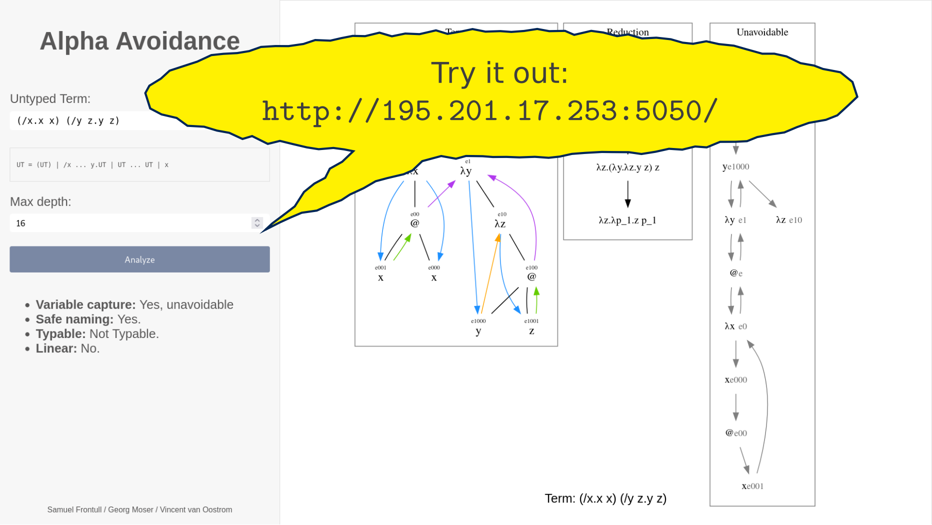
- **Variable capture:** Yes, unavoidable
- **Safe naming:** Yes.
- **Typable:** Not Typable.
- **Linear:** No.

Samuel Frontull / Georg Moser / Vincent van Oostrom



Term: (/x.x x) (/y z.y z)

α-Avoidance – Tool



Conclusion & Future Work

Known results...

from a new perspective/novel approach

Conclusion & Future Work

Known results...

from a new perspective/novel approach

Completeness

Find a complete characterisation for α-avoidance via (α-)paths

Conclusion & Future Work

Known results...

from a new perspective/novel approach

Completeness

Find a complete characterisation for α-avoidance via (α-)paths

Undecidability

Do we have general undecidability of α-avoidance?

Conclusion & Future Work

Known results...

from a new perspective/novel approach

Completeness

Find a complete characterisation for α-avoidance via (α-)paths

Undecidability

Do we have general undecidability of α-avoidance?

Alpha "circumvention"

Given some λ-term M, find a maximal reduction sequence where α is never needed.

Thank you for your attention!

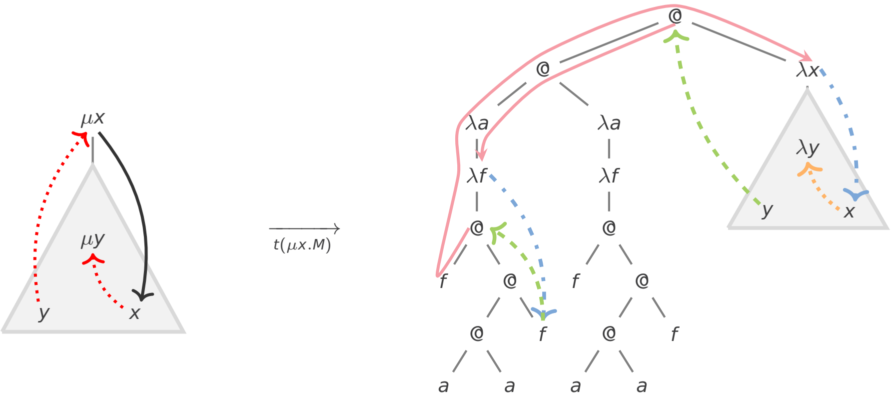
Reference

Samuel Frontull, Georg Moser, and Vincent van Oostrom. “α-Avoidance”. In: 8th International Conference on Formal Structures for Computation and Deduction (FSCD 2023). Ed. by Marco Gaboardi and Femke van Raamsdonk. Vol. 260. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023, 22:1–22:22. ISBN: 978-3-95977-277-8. URL: https://drops.dagstuhl.de/opus/volltexte/2023/18006.

Correspondence to binding–capturing chains in μ

The modal μ-calculus (Kozen 1983)

Unfolding does not create new redexes (Endrullis et al. 2011).



The safe λ-calculus

Claim, assuming the safe variable naming convention

Variable capture is guaranteed not to happen (Blum and Ong 2009).

$$(var)\ \frac{}{x : A \vdash_s x : A} \qquad (const)\ \frac{}{\vdash_s f : A}\ f : A \in \Xi \qquad (wk)\ \frac{\Gamma \vdash_s M : A}{\Delta \vdash_s M : A}\ \Gamma \subset \Delta \qquad (\delta)\ \frac{\Gamma \vdash_s M : A}{\Gamma \vdash_{asa} M : A}$$

$$(app_{asa})\ \frac{\Gamma \vdash_{asa} M : A \to B \quad \Gamma \vdash_s N : A}{\Gamma \vdash_{asa} M\ N : B} \qquad (app)\ \frac{\Gamma \vdash_{asa} M : A \to B \quad \Gamma \vdash_s N : A}{\Gamma \vdash_s M\ N : B}\ ord\ B \leq ord\ \Gamma$$

$$(abs)\ \frac{\Gamma, x_1 : A_1, \ldots, x_n : A_n \vdash_{asa} M : B}{\Gamma \vdash_s \lambda x_1^{A_1} \ldots x_n^{A_n}.M : (A_1, \ldots, A_n, B)}\ ord\ (A_1, \ldots, A_n, B) \leq ord\ \Gamma$$

The safe λ-calculus

Claim, assuming the safe variable naming convention

Variable capture is guaranteed not to happen (Blum and Ong 2009).

$$(var)\ \frac{}{x : A \vdash_s x : A} \qquad (const)\ \frac{}{\vdash_s f : A}\ f : A \in \Xi \qquad (wk)\ \frac{\Gamma \vdash_s M : A}{\Delta \vdash_s M : A}\ \Gamma \subset \Delta \qquad (\delta)\ \frac{\Gamma \vdash_s M : A}{\Gamma \vdash_{asa} M : A}$$

$$(app_{asa})\ \frac{\Gamma \vdash_{asa} M : A \to B \quad \Gamma \vdash_s N : A}{\Gamma \vdash_{asa} M\ N : B} \qquad (app)\ \frac{\Gamma \vdash_{asa} M : A \to B \quad \Gamma \vdash_s N : A}{\Gamma \vdash_s M\ N : B}\ ord\ B \leq ord\ \Gamma$$

$$(abs)\ \frac{\Gamma, x_1 : A_1, \ldots, x_n : A_n \vdash_{asa} M : B}{\Gamma \vdash_s \lambda x_1^{A_1} \ldots x_n^{A_n}.M : (A_1, \ldots, A_n, B)}\ ord\ (A_1, \ldots, A_n, B) \leq ord\ \Gamma$$

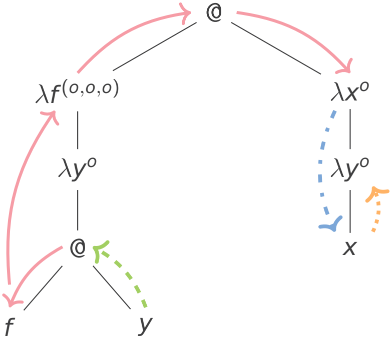
A term where α is needed can be derived: $\vdash_s (\lambda f^{(o,o,o)} y^o.f\ y)\,(\lambda x^o y^o.x)$

$$(\lambda fy.f\ y)\,(\lambda xy.x) \to_{\beta_{sim}} \lambda y.(\lambda xy.x)\,y \to_{\beta_{sim}} \lambda y.\lambda y'.y$$

The safe λ-calculus

Counterexample: $\vdash_s (\lambda f^{(o,o,o)} y^o . f\, y)\,(\lambda x^o y^o . x)$

α is needed although the term is safe and the naming convention is followed.



The safe λ-calculus

Solution

A more restrictive set of rules forbidding ”almost-safe” constructions.

$$(var)\ \frac{}{\{x : A\} \vdash_{s\alpha} x : A} \qquad (const)\ \frac{}{\vdash_{s\alpha} f : A}\ f : A \in \Xi \qquad (wk)\ \frac{\Gamma' \vdash_{s\alpha} M : A}{\Gamma \vdash_{s\alpha} M : A}\ \Gamma' \subset \Gamma$$

$$(app)\ \frac{\Gamma \vdash_{s\alpha} M : (A_1, \ldots, A_n, B) \quad \Gamma_{\geq m} \vdash_{s\alpha} N_1 : A_1 \quad \ldots \quad \Gamma_{\geq m} \vdash_{s\alpha} N_j : B_j}{\Gamma \vdash_{s\alpha} M\, N_1\ \ldots\ N_j : B} \quad m = ord\, B$$

$$(abs)\ \frac{\Gamma_{\geq m} \cup \{x_1 : A_1, \ldots, x_n : A_n\} \vdash_{s\alpha} M : B}{\Gamma \vdash_{s\alpha} \lambda x_1 \ldots x_n.M : (A_1, \ldots, A_n, B)} \quad m = ord\, (A_1, \ldots, A_n, B)$$

Long-safety

These rules correspond to the typing rules for long-safe terms (Blum 2009; Blum and Ong 2009).

Naïve β-step

M	$[\![x := N]\!]$ (capture-avoiding)	$[x := N]$ (capture-permitting)
x	N	N
y	y	y
$e_1\, e_2$	$e_1[\![x := N]\!]\, e_2[\![x := N]\!]$	$e_1[x := N]\, e_2[x := N]$
$\lambda x.e$	$\lambda x.e$	$\lambda x.e$
$\lambda y.e$	$\lambda y.e[\![x := N]\!]$ if $y \notin \mathcal{F}V(N)$	$\lambda y.e[x := N]$
	$\lambda z.e[\![y := z]\!][\![x := N]\!]$ else with z fresh for e and N.	

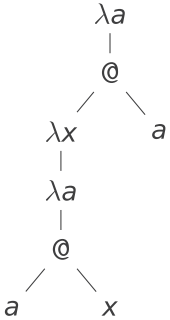
Definition

$$(\lambda x.M)\, N \to_{\beta_{naive}} M[x := N]$$

Variable names are irrelevant

De Bruijn's lambda notation (Bruijn 1972)

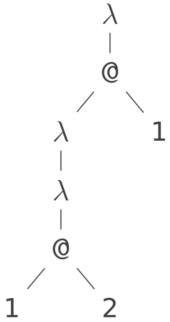
Exclusively work with (representatives of) α-equivalence classes of λ-terms



Variable names are irrelevant

De Bruijn's lambda notation (Bruijn 1972)

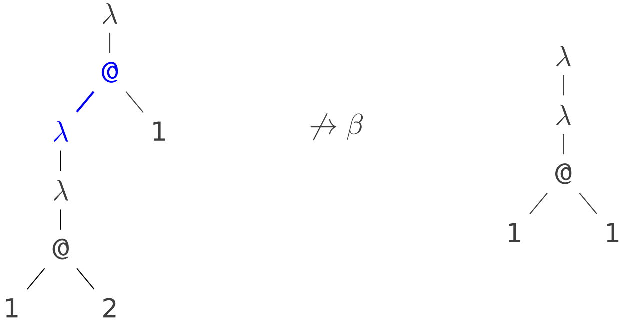
Exclusively work with (representatives of) α-equivalence classes of λ-terms



Variable names are irrelevant

De Bruijn's lambda notation (Bruijn 1972)

Exclusively work with (representatives of) α-equivalence classes of λ-terms



Variable names are irrelevant

De Bruijn's lambda notation (Bruijn 1972)

Exclusively work with (representatives of) α-equivalence classes of λ-terms