Typed λ-Calculus

We can avoid Russell's paradox using simple types.

Typed λ-Calculus

We can avoid Russell's paradox using simple types.
Simple Types:

- o Base type of propositions

Typed λ-Calculus

We can avoid Russell's paradox using simple types.

Simple Types:

- o Base type of propositions
- ι Base type of individuals

Typed λ-Calculus

We can avoid Russell's paradox using simple types.
Simple Types:

- o Base type of propositions
- ι Base type of individuals
- $(\alpha\beta)$ (or $(\beta \to \alpha)$) Type of functions from β to α

Typed λ-Calculus

We can avoid Russell's paradox using simple types.
Simple Types:

- o Base type of propositions
- ι Base type of individuals
- $(\alpha\beta)$ (or $(\beta \to \alpha)$) Type of functions from β to α

One may include arbitrarily many base types $\iota^1, \ldots, \iota^n, \ldots$.

Typed λ-Calculus

We can avoid Russell's paradox using simple types.
Simple Types:

- o Base type of propositions
- ι Base type of individuals
- $(\alpha\beta)$ (or $(\beta \to \alpha)$) Type of functions from β to α

We often omit parenthesis in types. $(\alpha\beta\gamma)$ means $((\alpha\beta)\gamma)$

Typed λ-Calculus

We can avoid Russell's paradox using simple types.
Simple Types:

- o Base type of propositions
- ι Base type of individuals
- $(\alpha\beta)$ (or $(\beta \to \alpha)$) Type of functions from β to α

We often omit parenthesis in types. $(\alpha\beta\gamma)$ means $((\alpha\beta)\gamma)$
Likewise $(\gamma \to \beta \to \alpha)$ means $(\gamma \to (\beta \to \alpha))$

Typed λ-Calculus

We can avoid Russell's paradox using simple types.
Simple Types:

- o Base type of propositions
- ι Base type of individuals
- $(\alpha\beta)$ (or $(\beta \to \alpha)$) Type of functions from β to α

We often omit parenthesis in types. $(\alpha\beta\gamma)$ means $((\alpha\beta)\gamma)$
Likewise $(\gamma \to \beta \to \alpha)$ means $(\gamma \to (\beta \to \alpha))$
Note that the type $(\alpha\beta\gamma)$ (or $(\gamma \to \beta \to \alpha)$) is the type of a (Curried) function of two arguments which returns a value of type α.

Typed λ-Calculus: Typed Terms

- Typed Variables x_α

Typed λ-Calculus: Typed Terms

- Typed Variables x_α
- Typed Constants and Parameters P_α

Typed λ-Calculus: Typed Terms

- Typed Variables x_α
- Typed Constants and Parameters P_α
- Application $[F_{\alpha\beta} B_\beta]_\alpha$ – or $[F_{\beta\rightarrow\alpha} B_\beta]_\alpha$

Typed λ-Calculus: Typed Terms

- Typed Variables x_α
- Typed Constants and Parameters P_α
- Application $[F_{\alpha\beta} B_\beta]_\alpha$ – or $[F_{\beta\rightarrow\alpha} B_\beta]_\alpha$
- λ-abstraction $[\lambda y_\beta . A_\alpha]_{\alpha\beta}$ – or $[\lambda y_\beta . A_\alpha]_{\beta\rightarrow\alpha}$

Typed λ-Calculus: Typed Terms

- Typed Variables x_α
- Typed Constants and Parameters P_α
- Application $[F_{\alpha\beta}B_\beta]_\alpha$ – or $[F_{\beta\rightarrow\alpha}B_\beta]_\alpha$
- λ-abstraction $[\lambda y_\beta . A_\alpha]_{\alpha\beta}$ – or $[\lambda y_\beta . A_\alpha]_{\beta\rightarrow\alpha}$

Examples:

- $[\lambda x_\alpha . x_\alpha]$ term of type $(\alpha\alpha)$ – identity on type α

Typed λ-Calculus: Typed Terms

- Typed Variables x_α
- Typed Constants and Parameters P_α
- Application $[F_{\alpha\beta}B_\beta]_\alpha$ – or $[F_{\beta\rightarrow\alpha}B_\beta]_\alpha$
- λ-abstraction $[\lambda y_\beta.\, A_\alpha]_{\alpha\beta}$ – or $[\lambda y_\beta.\, A_\alpha]_{\beta\rightarrow\alpha}$

Examples:

- $[\lambda x_\alpha.\, x_\alpha]$ term of type $(\alpha\alpha)$ – identity on type α
- $[\lambda y_\beta.\, x_\alpha]$ term of type $(\alpha\beta)$ – constant x-valued function

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.x^2 - 1]$$

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.x^2 - 1]$$

This is shorthand for

$$[\lambda x.\,[\mathsf{MINUS}\,[\mathsf{SQUARE}\,x]\,1]]$$

where MINUS, SQUARE and 1 are constants.

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda \mathsf{x}.\mathsf{x}^2 - 1]$$

This is shorthand for

$$[\lambda \mathsf{x}.\ [\mathsf{MINUS}\ [\mathsf{SQUARE}\ \mathsf{x}]\ 1]]$$

where MINUS, SQUARE and 1 are constants.

Is there a corresponding typed term?

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda \mathsf{x}.\mathsf{x}^2 - 1]$$

This is shorthand for

$$[\lambda \mathsf{x}.\ [\mathsf{MINUS}\ [\mathsf{SQUARE}\ \mathsf{x}]\ 1]]$$

where MINUS, SQUARE and 1 are constants.

Is there a corresponding typed term?

Assume the type of individuals ι corresponds to real numbers.

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.x^2 - 1]$$

This is shorthand for

$$[\lambda x.\,[\mathsf{MINUS}\,[\mathsf{SQUARE}\,x]\,1]]$$

where MINUS, SQUARE and 1 are constants.

Is there a corresponding typed term?

Assume the type of individuals ι corresponds to real numbers.

- x and 1 should be real numbers (type ι)

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.x^2 - 1]$$

This is shorthand for

$$[\lambda x.\, [\mathsf{MINUS}\, [\mathsf{SQUARE}\, x]\, 1]]$$

where MINUS, SQUARE and 1 are constants.

Is there a corresponding typed term?

Assume the type of individuals ι corresponds to real numbers.

- x and 1 should be real numbers (type ι)
- SQUARE should take a real number to a real number (type $(\iota\iota)$)

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda \mathsf{x}.\mathsf{x}^2 - 1]$$

This is shorthand for

$$[\lambda \mathsf{x}.\ [\mathsf{MINUS}\ [\mathsf{SQUARE}\ \mathsf{x}]\ 1]]$$

where MINUS, SQUARE and 1 are constants.

Is there a corresponding typed term?

Assume the type of individuals ι corresponds to real numbers.

- x and 1 should be real numbers (type ι)
- SQUARE should take a real number to a real number (type $(\iota\iota)$)
- MINUS should take two real numbers to a real number (type $(\iota\iota\iota)$)

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.x^2 - 1]$$

This is shorthand for

$$[\lambda x.\, [\mathsf{MINUS}\, [\mathsf{SQUARE}\, x]\, 1]]$$

where MINUS, SQUARE and 1 are constants.

Is there a corresponding typed term?

Assume the type of individuals ι corresponds to real numbers.

Typed Term:

$$[\lambda x_\iota.\, [\mathsf{MINUS}_{\iota\iota\iota}\, [\mathsf{SQUARE}_{\iota\iota}\, x_\iota]\, 1_\iota]]$$

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda \mathsf{x}.\mathsf{x}^2 - 1]$$

This is shorthand for

$$[\lambda \mathsf{x}.\, [\mathsf{MINUS}\, [\mathsf{SQUARE}\, \mathsf{x}]\, 1]]$$

where MINUS, SQUARE and 1 are constants.

Is there a corresponding typed term?

Assume the type of individuals ι corresponds to real numbers.

Typed Term:

$$[\lambda \mathsf{x}_\iota.\, [\mathsf{MINUS}_{\iota\iota\iota}\, [\mathsf{SQUARE}_{\iota\iota}\, \mathsf{x}_\iota]\, 1_\iota]]$$

This term has type $(\iota\iota)$.

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\![\lambda x.\, [\![x^2 - 1 = 0]\!]]\!]$$

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.\,[x^2 - 1 = 0]]$$

This is shorthand for

$$[\lambda x\,.[=\ [\mathsf{MINUS}\,[\mathsf{SQUARE}\,x]\,1]\,0]]$$

where $=$, MINUS, SQUARE, 0 and 1 are constants.

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.\, [x^2 - 1 = 0]]$$

This is shorthand for

$$[\lambda x\,.[=\ [\mathsf{MINUS}\,[\mathsf{SQUARE}\,x]\,1]\,0]]$$

where $=$, MINUS, SQUARE, 0 and 1 are constants.

- Already know types of MINUS, SQUARE and 1.

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.\, [x^2 - 1 = 0]]$$

This is shorthand for

$$[\lambda x\, .[= \ [\mathsf{MINUS}\, [\mathsf{SQUARE}\, x]\, 1]\, 0]]$$

where $=$, MINUS, SQUARE, 0 and 1 are constants.

- Already know types of MINUS, SQUARE and 1.
- 0 should be a real number (type ι)

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.\, [x^2 - 1 = 0]]$$

This is shorthand for

$$[\lambda x\,.[=\ [\mathsf{MINUS}\,[\mathsf{SQUARE}\,x]\,1]\,0]]$$

where $=$, MINUS, SQUARE, 0 and 1 are constants.

- Already know types of MINUS, SQUARE and 1.
- 0 should be a real number (type ι)
- $=$ takes two real numbers and returns a truth value (type $(o\iota\iota)$)

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.\, [x^2 - 1 = 0]]$$

This is shorthand for

$$[\lambda x\,.[=\ [\mathsf{MINUS}\,[\mathsf{SQUARE}\,x]\,1]\,0]]$$

where $=$, MINUS, SQUARE, 0 and 1 are constants.
Typed Term:

$$[\lambda x_\iota.\,[=_{o\iota\iota}\ [\mathsf{MINUS}_{\iota\iota\iota}\,[\mathsf{SQUARE}_{\iota\iota}\,x_\iota]\,1_\iota]\,0_\iota]$$

Typed λ-Calculus: Typed Terms

Consider the untyped term

$$[\lambda x.\,[x^2 - 1 = 0]]$$

This is shorthand for

$$[\lambda x\,.[=\ [\mathsf{MINUS}\,[\mathsf{SQUARE}\,x]\,1]\,0]]$$

where $=$, MINUS, SQUARE, 0 and 1 are constants.
Typed Term:

$$[\lambda x_\iota.\,[=_{o\iota\iota}\ [\mathsf{MINUS}_{\iota\iota\iota}\,[\mathsf{SQUARE}_{\iota\iota}\,x_\iota]\,1_\iota]\,0_\iota]$$

This term has type $(o\iota)$.

Typed λ-Calculus: Assigning Types

General algorithm for assigning types to terms (when this is possible) – see Hindley97.

Typed λ-Calculus: Assigning Types

The basis for such an algorithm is the following deduction system:

Typed λ-Calculus: Assigning Types

The basis for such an algorithm is the following deduction system:

$$\frac{\mathsf{C} : \alpha \in \Gamma \quad \mathsf{C} \text{ variable, parameter or constant}}{\Gamma \vdash_{\mathsf{TA}} \mathsf{C} : \alpha}\,\mathsf{Hyp}$$

Typed λ-Calculus: Assigning Types

The basis for such an algorithm is the following deduction system:

$$\frac{\mathsf{C} : \alpha \in \Gamma \quad \text{C variable, parameter or constant}}{\Gamma \vdash_{\mathsf{TA}} \mathsf{C} : \alpha}\,\mathsf{Hyp}$$

$$\frac{\Gamma, \mathsf{y} : \beta \vdash_{\mathsf{TA}} \mathsf{A} : \alpha}{\Gamma \vdash_{\mathsf{TA}} [\lambda \mathsf{y}.\, \mathsf{A}] : \alpha\beta}\,\mathsf{Lam}$$

Typed λ-Calculus: Assigning Types

The basis for such an algorithm is the following deduction system:

$$\frac{\mathsf{C} : \alpha \in \Gamma \quad \mathsf{C}\text{ variable, parameter or constant}}{\Gamma \vdash_{\mathsf{TA}} \mathsf{C} : \alpha}\ \mathsf{Hyp}$$

$$\frac{\Gamma, \mathsf{y} : \beta \vdash_{\mathsf{TA}} \mathsf{A} : \alpha}{\Gamma \vdash_{\mathsf{TA}} [\lambda \mathsf{y}.\, \mathsf{A}] : \alpha\beta}\ \mathsf{Lam} \qquad \frac{\Gamma \vdash_{\mathsf{TA}} \mathsf{F} : \alpha\beta \quad \Gamma \vdash_{\mathsf{TA}} \mathsf{B} : \beta}{\Gamma \vdash_{\mathsf{TA}} [\mathsf{F}\,\mathsf{B}] : \alpha}\ \mathsf{App}$$

Typed λ-Calculus: Assigning Types

The basis for such an algorithm is the following deduction system:

$$\frac{\mathsf{C} : \alpha \in \Gamma \quad \mathsf{C}\text{ variable, parameter or constant}}{\Gamma \vdash_{\mathsf{TA}} \mathsf{C} : \alpha}\,\mathsf{Hyp}$$

$$\frac{\Gamma, \mathsf{y} : \beta \vdash_{\mathsf{TA}} \mathsf{A} : \alpha}{\Gamma \vdash_{\mathsf{TA}} [\lambda \mathsf{y}.\, \mathsf{A}] : \alpha\beta}\,\mathsf{Lam} \qquad \frac{\Gamma \vdash_{\mathsf{TA}} \mathsf{F} : \alpha\beta \quad \Gamma \vdash_{\mathsf{TA}} \mathsf{B} : \beta}{\Gamma \vdash_{\mathsf{TA}} [\mathsf{F}\,\mathsf{B}] : \alpha}\,\mathsf{App}$$

We can assign the type α to a term A in context Γ whenever we can derive

$$\Gamma \vdash_{\mathsf{TA}} \mathsf{A} : \alpha$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}.\, [\mathsf{SQUARE}\,\mathsf{x}]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\, [\mathsf{SQUARE}\,\mathsf{x}]] : \alpha$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}.\, [\mathsf{SQUARE}\, \mathsf{x}]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\, [\mathsf{SQUARE}\, \mathsf{x}]] : \alpha$

$$\begin{array}{c} \vdots \\ \mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\, [\mathsf{SQUARE}\, \mathsf{x}]] : \alpha \end{array}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}. [\mathsf{SQUARE}\,\mathsf{x}]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}. [\mathsf{SQUARE}\,\mathsf{x}]] : \alpha$

α is $(\gamma\beta)$

$$\dfrac{\begin{array}{c}\vdots \\ \mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \beta \vdash_{\mathsf{TA}} [\mathsf{SQUARE}\,\mathsf{x}] : \gamma\end{array}}{\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}. [\mathsf{SQUARE}\,\mathsf{x}]] : \gamma\beta}\ \mathsf{Lam}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}.\, [\mathsf{SQUARE}\, \mathsf{x}]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\, [\mathsf{SQUARE}\, \mathsf{x}]] : \alpha$

$$\dfrac{\dfrac{\dfrac{\vdots}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \beta \vdash_{\mathsf{TA}} \mathsf{SQUARE} : (\gamma\delta)} \qquad \dfrac{\vdots}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \beta \vdash_{\mathsf{TA}} \mathsf{x} : \delta}}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \beta \vdash_{\mathsf{TA}} [\mathsf{SQUARE}\, \mathsf{x}] : \gamma}\,\mathsf{App}}{\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\, [\mathsf{SQUARE}\, \mathsf{x}]] : \gamma\beta}\,\mathsf{Lam}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}. [\mathsf{SQUARE}\,\mathsf{x}]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}. [\mathsf{SQUARE}\,\mathsf{x}]] : \alpha$

γ and δ are both ι

$$\dfrac{\dfrac{\dfrac{}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \beta \vdash_{\mathsf{TA}} \mathsf{SQUARE} : (\iota\iota)}\,\mathsf{Hyp} \qquad \dfrac{\vdots}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \beta \vdash_{\mathsf{TA}} \mathsf{x} : \iota}}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \beta \vdash_{\mathsf{TA}} [\mathsf{SQUARE}\,\mathsf{x}] : \iota}\,\mathsf{App}}{\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}. [\mathsf{SQUARE}\,\mathsf{x}]] : \iota\beta}\,\mathsf{Lam}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}.\,[\mathsf{SQUARE}\,\mathsf{x}]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\,[\mathsf{SQUARE}\,\mathsf{x}]] : \alpha$

β is ι

$$\dfrac{\dfrac{\dfrac{}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \iota \vdash_{\mathsf{TA}} \mathsf{SQUARE} : (\iota\iota)}\,\mathsf{Hyp} \qquad \dfrac{}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \iota \vdash_{\mathsf{TA}} \mathsf{x} : \iota}\,\mathsf{Hyp}}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \iota \vdash_{\mathsf{TA}} [\mathsf{SQUARE}\,\mathsf{x}] : \iota}\,\mathsf{App}}{\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\,[\mathsf{SQUARE}\,\mathsf{x}]] : \iota\iota}\,\mathsf{Lam}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}.\,[\mathsf{SQUARE}\,\mathsf{x}]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\,[\mathsf{SQUARE}\,\mathsf{x}]] : \alpha$

β is ι

$$\dfrac{\dfrac{\dfrac{}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \iota \vdash_{\mathsf{TA}} \mathsf{SQUARE} : (\iota\iota)}\,\mathsf{Hyp} \qquad \dfrac{}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \iota \vdash_{\mathsf{TA}} \mathsf{x} : \iota}\,\mathsf{Hyp}}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \iota \vdash_{\mathsf{TA}} [\mathsf{SQUARE}\,\mathsf{x}] : \iota}\,\mathsf{App}}{\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\,[\mathsf{SQUARE}\,\mathsf{x}]] : \iota\iota}\,\mathsf{Lam}$$

So $[\lambda \mathsf{x}.\,[\mathsf{SQUARE}\,\mathsf{x}]]$ can be assigned the type $(\iota\iota)$ in context $\mathsf{SQUARE} : (\iota\iota)$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}.\,[\mathsf{SQUARE}\,\mathsf{x}]]$

Goal: Find a type α such that

$\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\,[\mathsf{SQUARE}\,\mathsf{x}]] : \alpha$

β is ι

$$\dfrac{\dfrac{}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \iota \vdash_{\mathsf{TA}} \mathsf{SQUARE} : (\iota\iota)}\,\mathsf{Hyp} \qquad \dfrac{}{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \iota \vdash_{\mathsf{TA}} \mathsf{x} : \iota}\,\mathsf{Hyp}}{\dfrac{\mathsf{SQUARE} : (\iota\iota), \mathsf{x} : \iota \vdash_{\mathsf{TA}} [\mathsf{SQUARE}\,\mathsf{x}] : \iota}{\mathsf{SQUARE} : (\iota\iota) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\,[\mathsf{SQUARE}\,\mathsf{x}]] : \iota\iota}\,\mathsf{Lam}}\,\mathsf{App}$$

So $[\lambda \mathsf{x}.\,[\mathsf{SQUARE}\,\mathsf{x}]]$ can be assigned the type $(\iota\iota)$ in context $\mathsf{SQUARE} : (\iota\iota)$

Corresponding Typed Term: $[\lambda \mathsf{x}_\iota.\,[\mathsf{SQUARE}_{\iota\iota}\,\mathsf{x}_\iota]]$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}\,.\,\neg\,[\mathsf{x}\,\mathsf{x}]]$

Goal: Find a type α such that $\neg : (\mathsf{oo}) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}\,.\,\neg\,[\mathsf{x}\,\mathsf{x}]] : \alpha$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x} . \neg [\mathsf{x}\,\mathsf{x}]]$

Goal: Find a type α such that $\neg : (\mathsf{oo}) \vdash_{\mathsf{TA}} [\lambda \mathsf{x} . \neg [\mathsf{x}\,\mathsf{x}]] : \alpha$

$$\begin{array}{c} \vdots \\ \neg : (\mathsf{oo}) \vdash_{\mathsf{TA}} [\lambda \mathsf{x} . \neg [\mathsf{x}\,\mathsf{x}]] : \alpha \end{array}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x} . \neg [\mathsf{x}\,\mathsf{x}]]$

Goal: Find a type α such that $\neg : (\mathsf{oo}) \vdash_{\mathsf{TA}} [\lambda \mathsf{x} . \neg [\mathsf{x}\,\mathsf{x}]] : \alpha$

α is $(\gamma\beta)$

$$\cfrac{\begin{array}{c}\vdots\\ \neg : (\mathsf{oo}), \mathsf{x} : \beta \vdash_{\mathsf{TA}} [\neg [\mathsf{x}\,\mathsf{x}]] : \gamma\end{array}}{\neg : (\mathsf{oo}) \vdash_{\mathsf{TA}} [\lambda \mathsf{x} . \neg [\mathsf{x}\,\mathsf{x}]] : \gamma\beta}\ \mathsf{Lam}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}\,.\,\neg\,[\mathsf{x}\,\mathsf{x}]]$

Goal: Find a type α such that $\neg : (oo) \vdash_{TA} [\lambda \mathsf{x}\,.\,\neg\,[\mathsf{x}\,\mathsf{x}]] : \alpha$

$$\dfrac{\dfrac{\dfrac{\vdots}{\neg : (oo), \mathsf{x} : \beta \vdash_{TA} \neg : (\gamma\delta)} \qquad \dfrac{\vdots}{\neg : (oo), \mathsf{x} : \beta \vdash_{TA} [\mathsf{x}\,\mathsf{x}] : \delta}}{\neg : (oo), \mathsf{x} : \beta \vdash_{TA} [\neg\,[\mathsf{x}\,\mathsf{x}]] : \gamma}\,\mathrm{App}}{\neg : (oo) \vdash_{TA} [\lambda \mathsf{x}.\,[\neg\,[\mathsf{x}\,\mathsf{x}]] : \gamma\beta}\,\mathrm{Lam}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}\,.\,\neg\,[\mathsf{x}\,\mathsf{x}]]$

Goal: Find a type α such that $\neg : (\mathsf{oo}) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}\,.\,\neg\,[\mathsf{x}\,\mathsf{x}]] : \alpha$

γ and δ are both o

$$\dfrac{\dfrac{}{\neg : (\mathsf{oo}), \mathsf{x} : \beta \vdash_{\mathsf{TA}} \neg : (\mathsf{oo})}\,\mathsf{Hyp} \qquad \dfrac{\vdots}{\neg : (\mathsf{oo}), \mathsf{x} : \beta \vdash_{\mathsf{TA}} [\mathsf{x}\,\mathsf{x}] : \mathsf{o}}}{\dfrac{\neg : (\mathsf{oo}), \mathsf{x} : \beta \vdash_{\mathsf{TA}} [\neg\,[\mathsf{x}\,\mathsf{x}]] : \mathsf{o}}{\neg : (\mathsf{oo}) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}.\,[\neg\,[\mathsf{x}\,\mathsf{x}]] : \mathsf{o}\beta}\,\mathsf{Lam}}\,\mathsf{App}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}\,.\,\neg\,[\mathsf{x}\,\mathsf{x}]]$

Goal: Find a type α such that $\neg : (\mathsf{oo}) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}\,.\,\neg\,[\mathsf{x}\,\mathsf{x}]] : \alpha$

$$\begin{array}{c} \vdots \\ \neg : (\mathsf{oo}), \mathsf{x} : \beta \vdash_{\mathsf{TA}} [\mathsf{x}\,\mathsf{x}] : \mathsf{o} \end{array}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x} \,.\, \neg\, [\mathsf{x}\,\mathsf{x}]]$

Goal: Find a type α such that $\neg : (\mathsf{oo}) \vdash_{\mathsf{TA}} [\lambda \mathsf{x} \,.\, \neg\, [\mathsf{x}\,\mathsf{x}]] : \alpha$

$$\dfrac{\dfrac{\vdots}{\neg : (\mathsf{oo}), \mathsf{x} : \beta \vdash_{\mathsf{TA}} \mathsf{x} : (\mathsf{o}\epsilon)} \qquad \dfrac{\vdots}{\neg : (\mathsf{oo}), \mathsf{x} : \beta \vdash_{\mathsf{TA}} \mathsf{x} : \epsilon}}{\neg : (\mathsf{oo}), \mathsf{x} : \beta \vdash_{\mathsf{TA}} [\mathsf{x}\,\mathsf{x}] : \mathsf{o}}\ \mathsf{App}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda x . \neg [x\, x]]$

Goal: Find a type α such that $\neg : (oo) \vdash_{TA} [\lambda x . \neg [x\, x]] : \alpha$

β is $(o\epsilon)$

$$\dfrac{\dfrac{}{\neg : (oo), x : (o\epsilon) \vdash_{TA} x : (o\epsilon)}\text{Hyp} \qquad \begin{array}{c}\vdots \\ \neg : (oo), x : (o\epsilon) \vdash_{TA} x : \epsilon\end{array}}{\neg : (oo), x : (o\epsilon) \vdash_{TA} [x\, x] : o}\text{App}$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}\,.\,\neg\,[\mathsf{x}\,\mathsf{x}]]$

Goal: Find a type α such that $\neg : (\mathsf{oo}) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}\,.\,\neg\,[\mathsf{x}\,\mathsf{x}]] : \alpha$

Only remaining subgoal:

$$\neg : (\mathsf{oo}), \mathsf{x} : (\mathsf{o}\epsilon) \vdash_{\mathsf{TA}} \mathsf{x} : \epsilon$$

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x} \,.\, \neg\, [\mathsf{x}\,\mathsf{x}]]$
Goal: Find a type α such that $\neg : (\mathsf{oo}) \vdash_{\mathsf{TA}} [\lambda \mathsf{x} \,.\, \neg\, [\mathsf{x}\,\mathsf{x}]] : \alpha$
Only remaining subgoal:

$$\neg : (\mathsf{oo}), \mathsf{x} : (\mathsf{o}\epsilon) \vdash_{\mathsf{TA}} \mathsf{x} : \epsilon$$

This goal cannot be solved since $(\mathsf{o}\epsilon)$ cannot equal ϵ.

Typed λ-Calculus: Assigning Types

Untyped Term: $[\lambda \mathsf{x}\,.\,\neg\,[\mathsf{x}\,\mathsf{x}]]$
Goal: Find a type α such that $\neg : (\mathsf{oo}) \vdash_{\mathsf{TA}} [\lambda \mathsf{x}\,.\,\neg\,[\mathsf{x}\,\mathsf{x}]] : \alpha$
Only remaining subgoal:

$$\neg : (\mathsf{oo}), \mathsf{x} : (\mathsf{o}\epsilon) \vdash_{\mathsf{TA}} \mathsf{x} : \epsilon$$

This goal cannot be solved since $(\mathsf{o}\epsilon)$ cannot equal ϵ.

Hence $[\lambda \mathsf{x}.\,[\neg\,[\mathsf{x}\,\mathsf{x}]]]$ cannot be typed – avoiding Russell's Paradox.

Typed λ-Calculus: $\beta\eta$

β-reduction:

$$[[\lambda \mathsf{y}_\beta\,.\,\mathsf{A}_\alpha]\ \mathsf{B}_\beta] \longrightarrow_\beta \mathsf{A}_\alpha[\mathsf{y}_\beta/\mathsf{B}_\beta]$$

Typed λ-Calculus: $\beta\eta$

β-reduction:

$$[[\lambda y_\beta \,.\, A_\alpha]\ B_\beta] \longrightarrow_\beta A_\alpha[y_\beta/B_\beta]$$

η-reduction:

$$[\lambda y_\beta . F_{\alpha\beta}\, y_\beta] \longrightarrow_\eta F_{\alpha\beta}$$

Typed λ-Calculus: $\beta\eta$

β-reduction:

$$[[\lambda \mathsf{y}_\beta \,.\, \mathsf{A}_\alpha]\ \mathsf{B}_\beta] \longrightarrow_\beta \mathsf{A}_\alpha[\mathsf{y}_\beta/\mathsf{B}_\beta]$$

η-reduction:

$$[\lambda \mathsf{y}_\beta.\mathsf{F}_{\alpha\beta}\,\mathsf{y}_\beta] \longrightarrow_\eta \mathsf{F}_{\alpha\beta}$$

Facts:

- $\beta\eta$-normalization terminates for typed terms.

Typed λ-Calculus: $\beta\eta$

β-reduction:

$$[[\lambda \mathsf{y}_\beta \,.\, \mathsf{A}_\alpha]\ \mathsf{B}_\beta] \longrightarrow_\beta \mathsf{A}_\alpha[\mathsf{y}_\beta/\mathsf{B}_\beta]$$

η-reduction:

$$[\lambda \mathsf{y}_\beta.\mathsf{F}_{\alpha\beta}\, \mathsf{y}_\beta] \longrightarrow_\eta \mathsf{F}_{\alpha\beta}$$

Facts:

- $\beta\eta$-normalization terminates for typed terms.
- Every typed term has a unique $\beta\eta$-normal form.