Simply Typed Lambda Calculus

Stephen A. Edwards

Columbia University

Spring 2023

Lambda Calculus

$$v ::= x \mid \lambda x \,.\, t$$

$$t ::= v \mid t\ t$$

Lambda Calculus

$$v ::= x \mid \lambda x \,.\, t$$

$$t ::= v \mid t\ t$$

Call-by-value Operational Semantics: Reduce to a value

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\,\text{func} \quad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\,\text{arg} \quad \frac{t[x := v] = t'}{(\lambda x\,.\,t)\ v \longrightarrow t'}\,\text{beta}$$

Lambda Calculus + Booleans

$$v ::= x \mid \lambda x\ .\ t \mid \mathbf{true} \mid \mathbf{false}$$

$$t ::= v \mid t\ t \mid \mathbf{if}\ t\ \mathbf{then}\ t\ \mathbf{else}\ t$$

Call-by-value Operational Semantics: Reduce to a value

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\ \text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x\ .\ t)\ v \longrightarrow t'}\ \text{beta}$$

Lambda Calculus + Booleans

$$v \ ::= \ x \mid \lambda x\ .\ t \mid \mathtt{true} \mid \mathtt{false}$$

$$t \ ::= \ v \mid t\ t \mid \mathtt{if}\ t\ \mathtt{then}\ t\ \mathtt{else}\ t$$

Call-by-value Operational Semantics: Reduce to a value

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\,\text{func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\,\text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x\ .\ t)\ v \longrightarrow t'}\,\text{beta} \qquad \frac{}{\mathtt{if}\ \mathtt{true}\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow t_2}\,\text{true}$$

$$\frac{t_1 \longrightarrow t_1'}{\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow \mathtt{if}\ t_1'\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3}\,\text{pred} \qquad \frac{}{\mathtt{if}\ \mathtt{false}\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow t_3}\,\text{false}$$

Lambda Calculus + Booleans

$v ::= x \mid \lambda x\,.\,t \mid \mathtt{true} \mid \mathtt{false}$

$t ::= v \mid t\ t \mid \mathtt{if}\ t\ \mathtt{then}\ t\ \mathtt{else}\ t$

Call-by-value Operational Semantics: Reduce to a value

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\ \text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x\,.\,t)\ v \longrightarrow t'}\ \text{beta} \qquad \frac{}{\mathtt{if}\ \mathtt{true}\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow t_2}\ \text{true}$$

$$\frac{t_1 \longrightarrow t_1'}{\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow \mathtt{if}\ t_1'\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3}\ \text{pred} \qquad \frac{}{\mathtt{if}\ \mathtt{false}\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow t_3}\ \text{false}$$

$$(\lambda x\,.\,\lambda y\,.\,\mathtt{if}\ x\ \mathtt{then}\ y\ \mathtt{else}\ \mathtt{false})\ \mathtt{false}\ \mathtt{true}$$

Lambda Calculus + Booleans

$$v ::= x \mid \lambda x \,.\, t \mid \mathbf{true} \mid \mathbf{false}$$

$$t ::= v \mid t\ t \mid \mathbf{if}\ t\ \mathbf{then}\ t\ \mathbf{else}\ t$$

Call-by-value Operational Semantics: Reduce to a value

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\ \text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x \,.\, t)\ v \longrightarrow t'}\ \text{beta} \qquad \frac{}{\mathbf{if\ true\ then}\ t_2\ \mathbf{else}\ t_3 \longrightarrow t_2}\ \text{true}$$

$$\frac{t_1 \longrightarrow t_1'}{\mathbf{if}\ t_1\ \mathbf{then}\ t_2\ \mathbf{else}\ t_3 \longrightarrow \mathbf{if}\ t_1'\ \mathbf{then}\ t_2\ \mathbf{else}\ t_3}\ \text{pred} \qquad \frac{}{\mathbf{if\ false\ then}\ t_2\ \mathbf{else}\ t_3 \longrightarrow t_3}\ \text{false}$$

$$(\lambda x \,.\, \lambda y \,.\, \mathbf{if}\ x\ \mathbf{then}\ y\ \mathbf{else\ false})\ \mathbf{false\ true}$$

$$\longrightarrow (\lambda y \,.\, \mathbf{if\ false\ then}\ y\ \mathbf{else\ false})\ \mathbf{true}$$

Lambda Calculus + Booleans

$$v ::= x \mid \lambda x\,.\,t \mid \texttt{true} \mid \texttt{false}$$

$$t ::= v \mid t\ t \mid \texttt{if}\ t\ \texttt{then}\ t\ \texttt{else}\ t$$

Call-by-value Operational Semantics: Reduce to a value

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\,\text{func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\,\text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x\,.\,t)\ v \longrightarrow t'}\,\text{beta} \qquad \frac{}{\texttt{if true then}\ t_2\ \texttt{else}\ t_3 \longrightarrow t_2}\,\text{true}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 \longrightarrow \texttt{if}\ t_1'\ \texttt{then}\ t_2\ \texttt{else}\ t_3}\,\text{pred} \qquad \frac{}{\texttt{if false then}\ t_2\ \texttt{else}\ t_3 \longrightarrow t_3}\,\text{false}$$

$$\begin{aligned}
&(\lambda x\,.\,\lambda y\,.\,\texttt{if}\ x\ \texttt{then}\ y\ \texttt{else false})\ \texttt{false true}\\
&\longrightarrow (\lambda y\,.\,\texttt{if false then}\ y\ \texttt{else false})\ \texttt{true}\\
&\longrightarrow \texttt{if false then true else false}
\end{aligned}$$

Lambda Calculus + Booleans

$$v \; ::= \; x \mid \lambda x \,.\, t \mid \texttt{true} \mid \texttt{false}$$

$$t \; ::= \; v \mid t\ t \mid \texttt{if}\ t\ \texttt{then}\ t\ \texttt{else}\ t$$

Call-by-value Operational Semantics: Reduce to a value

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\ \text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x \,.\, t)\ v \longrightarrow t'}\ \text{beta} \qquad \frac{}{\texttt{if true then}\ t_2\ \texttt{else}\ t_3 \longrightarrow t_2}\ \text{true}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 \longrightarrow \texttt{if}\ t_1'\ \texttt{then}\ t_2\ \texttt{else}\ t_3}\ \text{pred} \qquad \frac{}{\texttt{if false then}\ t_2\ \texttt{else}\ t_3 \longrightarrow t_3}\ \text{false}$$

$$\begin{aligned}
&(\lambda x \,.\, \lambda y \,.\, \texttt{if}\ x\ \texttt{then}\ y\ \texttt{else false})\ \texttt{false true} \\
&\longrightarrow (\lambda y \,.\, \texttt{if false then}\ y\ \texttt{else false})\ \texttt{true} \\
&\longrightarrow \texttt{if false then true else false} \\
&\longrightarrow \texttt{false}
\end{aligned}$$

Lambda Calculus + Booleans

$$v ::= x \mid \lambda x\,.\,t \mid \texttt{true} \mid \texttt{false}$$

$$t ::= v \mid t\ t \mid \texttt{if}\ t\ \texttt{then}\ t\ \texttt{else}\ t$$

Call-by-value Operational Semantics: Reduce to a value

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\text{func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x\,.\,t)\ v \longrightarrow t'}\text{beta} \qquad \frac{}{\texttt{if true then}\ t_2\ \texttt{else}\ t_3 \longrightarrow t_2}\text{true}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 \longrightarrow \texttt{if}\ t_1'\ \texttt{then}\ t_2\ \texttt{else}\ t_3}\text{pred} \qquad \frac{}{\texttt{if false then}\ t_2\ \texttt{else}\ t_3 \longrightarrow t_3}\text{false}$$

$$(\lambda x\,.\,\lambda y\,.\,\texttt{if}\ x\ \texttt{then}\ y\ \texttt{else false})\ (\lambda x\,.\,x)\ \texttt{true}$$

Lambda Calculus + Booleans

$$v ::= x \mid \lambda x\,.\,t \mid \texttt{true} \mid \texttt{false}$$

$$t ::= v \mid t\ t \mid \texttt{if}\ t\ \texttt{then}\ t\ \texttt{else}\ t$$

Call-by-value Operational Semantics: Reduce to a value

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\ \text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x\,.\,t)\ v \longrightarrow t'}\ \text{beta} \qquad \frac{}{\texttt{if}\ \texttt{true}\ \texttt{then}\ t_2\ \texttt{else}\ t_3 \longrightarrow t_2}\ \text{true}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 \longrightarrow \texttt{if}\ t_1'\ \texttt{then}\ t_2\ \texttt{else}\ t_3}\ \text{pred} \qquad \frac{}{\texttt{if}\ \texttt{false}\ \texttt{then}\ t_2\ \texttt{else}\ t_3 \longrightarrow t_3}\ \text{false}$$

$$\begin{aligned} &(\lambda x\,.\,\lambda y\,.\,\texttt{if}\ x\ \texttt{then}\ y\ \texttt{else}\ \texttt{false})\ (\lambda x\,.\,x)\ \texttt{true} \\ &\longrightarrow (\lambda y\,.\,\texttt{if}\ (\lambda x\,.\,x)\ \texttt{then}\ y\ \texttt{else}\ \texttt{false})\ \texttt{true} \end{aligned}$$

Lambda Calculus + Booleans

$$v ::= x \mid \lambda x\,.\,t \mid \mathtt{true} \mid \mathtt{false}$$

$$t ::= v \mid t\ t \mid \mathtt{if}\ t\ \mathtt{then}\ t\ \mathtt{else}\ t$$

Call-by-value Operational Semantics: Reduce to a value

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\ \text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x\,.\,t)\ v \longrightarrow t'}\ \text{beta} \qquad \frac{}{\mathtt{if}\ \mathtt{true}\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow t_2}\ \text{true}$$

$$\frac{t_1 \longrightarrow t_1'}{\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow \mathtt{if}\ t_1'\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3}\ \text{pred} \qquad \frac{}{\mathtt{if}\ \mathtt{false}\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow t_3}\ \text{false}$$

$$\begin{aligned}
&(\lambda x\,.\,\lambda y\,.\,\mathtt{if}\ x\ \mathtt{then}\ y\ \mathtt{else}\ \mathtt{false})\ (\lambda x\,.\,x)\ \mathtt{true}\\
&\quad\longrightarrow (\lambda y\,.\,\mathtt{if}\ (\lambda x\,.\,x)\ \mathtt{then}\ y\ \mathtt{else}\ \mathtt{false})\ \mathtt{true}\\
&\quad\longrightarrow \mathtt{if}\ (\lambda x\,.\,x)\ \mathtt{then}\ \mathtt{true}\ \mathtt{else}\ \mathtt{false}
\end{aligned}$$

Lambda Calculus + Booleans

$$v ::= x \mid \lambda x \,.\, t \mid \mathbf{true} \mid \mathbf{false}$$

$$t ::= v \mid t\ t \mid \mathbf{if}\ t\ \mathbf{then}\ t\ \mathbf{else}\ t$$

Call-by-value Operational Semantics: Reduce to a value

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\ \text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x \,.\, t)\ v \longrightarrow t'}\ \text{beta} \qquad \frac{}{\mathbf{if}\ \mathbf{true}\ \mathbf{then}\ t_2\ \mathbf{else}\ t_3 \longrightarrow t_2}\ \text{true}$$

$$\frac{t_1 \longrightarrow t_1'}{\mathbf{if}\ t_1\ \mathbf{then}\ t_2\ \mathbf{else}\ t_3 \longrightarrow \mathbf{if}\ t_1'\ \mathbf{then}\ t_2\ \mathbf{else}\ t_3}\ \text{pred} \qquad \frac{}{\mathbf{if}\ \mathbf{false}\ \mathbf{then}\ t_2\ \mathbf{else}\ t_3 \longrightarrow t_3}\ \text{false}$$

$$
\begin{aligned}
&(\lambda x \,.\, \lambda y \,.\, \mathbf{if}\ x\ \mathbf{then}\ y\ \mathbf{else}\ \mathbf{false})\ (\lambda x \,.\, x)\ \mathbf{true}\\
&\qquad \longrightarrow (\lambda y \,.\, \mathbf{if}\ (\lambda x \,.\, x)\ \mathbf{then}\ y\ \mathbf{else}\ \mathbf{false})\ \mathbf{true}\\
&\qquad \longrightarrow \mathbf{if}\ (\lambda x \,.\, x)\ \mathbf{then}\ \mathbf{true}\ \mathbf{else}\ \mathbf{false}\\
&\qquad \mathit{stuck}
\end{aligned}
$$

Q: Can we statically characterize terms that will never get stuck?

Q: Can we statically characterize terms that will never get stuck?

Lambda Calculus + Booleans

$$v ::= x \mid \lambda x\,.\,t \mid \texttt{true} \mid \texttt{false} \qquad t ::= v \mid t\ t \mid \texttt{if}\ t\ \texttt{then}\ t\ \texttt{else}\ t$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\ \text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x\,.\,t)\ v \longrightarrow t'}\ \text{beta} \qquad \frac{}{\texttt{if true then}\ t_2\ \texttt{else}\ t_3 \longrightarrow t_2}\ \text{true}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 \longrightarrow \texttt{if}\ t_1'\ \texttt{then}\ t_2\ \texttt{else}\ t_3}\ \text{pred} \qquad \frac{}{\texttt{if false then}\ t_2\ \texttt{else}\ t_3 \longrightarrow t_3}\ \text{false}$$

Q: Can we statically characterize terms that will never get stuck?

Lambda Calculus + Booleans

$$v ::= x \mid \lambda x\,.\,t \mid \mathtt{true} \mid \mathtt{false} \qquad t ::= v \mid t\ t \mid \mathtt{if}\ t\ \mathtt{then}\ t\ \mathtt{else}\ t$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\ \text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x\,.\,t)\ v \longrightarrow t'}\ \text{beta} \qquad \frac{}{\mathtt{if\ true\ then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow t_2}\ \text{true}$$

$$\frac{t_1 \longrightarrow t_1'}{\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow \mathtt{if}\ t_1'\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3}\ \text{pred} \qquad \frac{}{\mathtt{if\ false\ then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow t_3}\ \text{false}$$

`bool`: May become either **`true`** or **`false`** `func`: May become a λ

$$v ::= x \mid \lambda x\,.\,t \mid \mathtt{true} \mid \mathtt{false} \qquad t ::= v \mid t\ t \mid \mathtt{if}\ t\ \mathtt{then}\ t\ \mathtt{else}\ t$$

Q: Can we statically characterize terms that will never get stuck?

Lambda Calculus + Booleans

$$v ::= x \mid \lambda x\,.\,t \mid \texttt{true} \mid \texttt{false} \qquad t ::= v \mid t\;t \mid \texttt{if}\;t\;\texttt{then}\;t\;\texttt{else}\;t$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\;t_2 \longrightarrow t_1'\;t_2}\text{func} \qquad \frac{t \longrightarrow t'}{v\;t \longrightarrow v\;t'}\text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x\,.\,t)\;v \longrightarrow t'}\text{beta} \qquad \frac{}{\texttt{if true then}\;t_2\;\texttt{else}\;t_3 \longrightarrow t_2}\text{true}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if}\;t_1\;\texttt{then}\;t_2\;\texttt{else}\;t_3 \longrightarrow \texttt{if}\;t_1'\;\texttt{then}\;t_2\;\texttt{else}\;t_3}\text{pred} \qquad \frac{}{\texttt{if false then}\;t_2\;\texttt{else}\;t_3 \longrightarrow t_3}\text{false}$$

bool: May become either **true** or **false** **func**: May become a λ

$$v ::= x \mid \lambda x\,.\,t \mid \texttt{true} \mid \texttt{false} \qquad t ::= v \mid t\;t \mid \texttt{if}\;t\;\texttt{then}\;t\;\texttt{else}\;t$$

Q: Can we statically characterize terms that will never get stuck?

Lambda Calculus + Booleans

$$v ::= x \mid \lambda x\ .\ t \mid \mathtt{true} \mid \mathtt{false} \qquad t ::= v \mid t\ t \mid \mathtt{if}\ t\ \mathtt{then}\ t\ \mathtt{else}\ t$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\text{func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x\ .\ t)\ v \longrightarrow t'}\text{beta} \qquad \frac{}{\mathtt{if}\ \mathtt{true}\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow t_2}\text{true}$$

$$\frac{t_1 \longrightarrow t_1'}{\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow \mathtt{if}\ t_1'\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3}\text{pred} \qquad \frac{}{\mathtt{if}\ \mathtt{false}\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow t_3}\text{false}$$

bool: May become either **true** or **false** **func**: May become a λ

$$v ::= x \mid \lambda x\ .\ t \mid \mathtt{true} \mid \mathtt{false} \qquad t ::= v \mid t\ t \mid \mathtt{if}\ t\ \mathtt{then}\ t\ \mathtt{else}\ t$$

What about the type of an argument? The *then* and *else* branches? Not good enough

Q: Can we statically characterize terms that will never get stuck?

Lambda Calculus + Booleans

$$v ::= x \mid \lambda x\ .\ t \mid \mathtt{true} \mid \mathtt{false} \qquad t ::= v \mid t\ t \mid \mathtt{if}\ t\ \mathtt{then}\ t\ \mathtt{else}\ t$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{func} \qquad \frac{t \longrightarrow t'}{v\ t \longrightarrow v\ t'}\ \text{arg} \qquad \frac{t[x := v] = t'}{(\lambda x\ .\ t)\ v \longrightarrow t'}\ \text{beta} \qquad \frac{}{\mathtt{if}\ \mathtt{true}\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow t_2}\ \text{true}$$

$$\frac{t_1 \longrightarrow t_1'}{\mathtt{if}\ t_1\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow \mathtt{if}\ t_1'\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3}\ \text{pred} \qquad \frac{}{\mathtt{if}\ \mathtt{false}\ \mathtt{then}\ t_2\ \mathtt{else}\ t_3 \longrightarrow t_3}\ \text{false}$$

Type Safety

Progress: If e is *well-typed*, then either e is a value or there exists e' such that $e \longrightarrow e'$.

Preservation: If e is well-typed and $e \longrightarrow e'$, then e' is well-typed.

Simply-Typed Lambda Calculus

Types $\tau ::= \texttt{bool} \mid \tau \longrightarrow \tau$

Types are either **bool** or functions from type to type, e.g., $\texttt{bool} \longrightarrow \texttt{bool}$

The $\longrightarrow$ operator associates right-to-left: curried functions are unparenthesized

$\texttt{bool} \longrightarrow \texttt{bool} \longrightarrow \texttt{bool}$ means $\texttt{bool} \longrightarrow (\texttt{bool} \longrightarrow \texttt{bool})$

This is opposite from application to make unparenthesized sequences “match up”

$f\ a\ b$ means $(f\ a)\ b$

Simply-Typed Lambda Calculus

Types $\tau ::= \texttt{bool} \mid \tau \longrightarrow \tau$

Values $v ::= \texttt{true} \mid \texttt{false} \mid x \mid \lambda x : \tau \,.\, t$

Types of function arguments are annotated

$\lambda x : \tau \,.\, t$ means x is an argument of type τ in the body t

Simply-Typed Lambda Calculus

Types	$\tau ::= \texttt{bool} \mid \tau \longrightarrow \tau$
Values	$v ::= \texttt{true} \mid \texttt{false} \mid x \mid \lambda x : \tau \,.\, t$
Terms	$t ::= v \mid t\; t \mid \texttt{if}\; t\; \texttt{then}\; t\; \texttt{else}\; t$

$t : \tau$ indicates term (or value) t is a well-typed with type τ

Simply-Typed Lambda Calculus

Types	$\tau ::= \texttt{bool} \mid \tau \longrightarrow \tau$
Values	$v ::= \texttt{true} \mid \texttt{false} \mid x \mid \lambda x : \tau \,.\, t$
Terms	$t ::= v \mid t\ t \mid \texttt{if}\ t\ \texttt{then}\ t\ \texttt{else}\ t$
Context	$\Gamma ::= \varnothing \mid \Gamma, x : \tau$

A context Γ is a partial map from variables to types.

$\varnothing$ is the empty map

$\Gamma, x : \tau$ adds the mapping from variable x to type τ, replacing any existing mapping of x

Simply-Typed Lambda Calculus

Types	$\tau ::= \mathtt{bool} \mid \tau \longrightarrow \tau$
Values	$v ::= \mathtt{true} \mid \mathtt{false} \mid x \mid \lambda x : \tau \,.\, t$
Terms	$t ::= v \mid t\ t \mid \mathtt{if}\ t\ \mathtt{then}\ t\ \mathtt{else}\ t$
Context	$\Gamma ::= \emptyset \mid \Gamma, x : \tau$

$$\frac{}{\Gamma \vdash \mathtt{true} : \mathtt{bool}}\ \text{t-true}$$

$$\frac{}{\Gamma \vdash \mathtt{false} : \mathtt{bool}}\ \text{t-false}$$

Type judgments: $\Gamma \vdash t : \tau$ means term t has type τ in context Γ

Base cases: **`true`** and **`false`** have type **`bool`** in any context

Simply-Typed Lambda Calculus

Types	$\tau ::= \texttt{bool} \mid \tau \longrightarrow \tau$
Values	$v ::= \texttt{true} \mid \texttt{false} \mid x \mid \lambda x : \tau \,.\, t$
Terms	$t ::= v \mid t\ t \mid \texttt{if}\ t\ \texttt{then}\ t\ \texttt{else}\ t$
Context	$\Gamma ::= \varnothing \mid \Gamma, x : \tau$

$$\frac{}{\Gamma \vdash \texttt{true} : \texttt{bool}}\ \text{t-true}$$

$$\frac{}{\Gamma \vdash \texttt{false} : \texttt{bool}}\ \text{t-false} \qquad \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ \text{t-var}$$

$x : \tau \in \Gamma$ holds true when variable x maps to type τ in Γ

This is the only purpose (or use) of a context: to tell us the type of an argument

Simply-Typed Lambda Calculus

Types	$\tau ::= \texttt{bool} \mid \tau \longrightarrow \tau$
Values	$v ::= \texttt{true} \mid \texttt{false} \mid x \mid \lambda x : \tau \,.\, t$
Terms	$t ::= v \mid t\ t \mid \texttt{if}\ t\ \texttt{then}\ t\ \texttt{else}\ t$
Context	$\Gamma ::= \varnothing \mid \Gamma, x : \tau$

$$\frac{}{\Gamma \vdash \texttt{true} : \texttt{bool}}\ \text{t-true}$$

$$\frac{}{\Gamma \vdash \texttt{false} : \texttt{bool}}\ \text{t-false} \qquad \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ \text{t-var}$$

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1 \,.\, t) : \tau_1 \longrightarrow \tau_2}\ \text{t-abs}$$

The type of $\lambda x \,.\, \tau_1 : t$

is a function from τ_1 to τ_2

provided its body t has type τ_2

when you assume the argument x has type τ_1.

This is the only time the context is extended

Simply-Typed Lambda Calculus

Types	$\tau ::= \mathbf{bool} \mid \tau \longrightarrow \tau$
Values	$v ::= \mathbf{true} \mid \mathbf{false} \mid x \mid \lambda x : \tau \,.\, t$
Terms	$t ::= v \mid t\ t \mid \mathbf{if}\ t\ \mathbf{then}\ t\ \mathbf{else}\ t$
Context	$\Gamma ::= \varnothing \mid \Gamma, x : \tau$

$$\frac{}{\Gamma \vdash \mathbf{true} : \mathbf{bool}}\ \text{t-true}$$

$$\frac{}{\Gamma \vdash \mathbf{false} : \mathbf{bool}}\ \text{t-false} \qquad \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ \text{t-var}$$

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1 \,.\, t) : \tau_1 \longrightarrow \tau_2}\ \text{t-abs}$$

$$\frac{\Gamma \vdash t_1 : \tau_1 \longrightarrow \tau_2 \qquad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\ t_2 : \tau_2}\ \text{t-app}$$

Applying t_2 to t_1

returns a type τ_2

provided t_1 is a function of type $\tau_1 \longrightarrow \tau_2$

and t_2 is of type τ_1

Simply-Typed Lambda Calculus

Types	$\tau ::= \texttt{bool} \mid \tau \longrightarrow \tau$
Values	$v ::= \texttt{true} \mid \texttt{false} \mid x \mid \lambda x : \tau \,.\, t$
Terms	$t ::= v \mid t\ t \mid \texttt{if}\ t\ \texttt{then}\ t\ \texttt{else}\ t$
Context	$\Gamma ::= \varnothing \mid \Gamma, x : \tau$

$$\frac{}{\Gamma \vdash \texttt{true} : \texttt{bool}}\ \text{t-true}$$

$$\frac{}{\Gamma \vdash \texttt{false} : \texttt{bool}}\ \text{t-false} \qquad \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ \text{t-var}$$

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1 \,.\, t) : \tau_1 \longrightarrow \tau_2}\ \text{t-abs}$$

$$\frac{\Gamma \vdash t_1 : \tau_1 \longrightarrow \tau_2 \qquad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\ t_2 : \tau_2}\ \text{t-app} \qquad \frac{\Gamma \vdash t_1 : \texttt{bool} \qquad \Gamma \vdash t_2 : \tau \qquad \Gamma \vdash t_3 : \tau}{\Gamma \vdash \texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 : \tau}\ \text{t-cond}$$

The type of a *if-then-else* term is a type τ

provided the predicate is a **bool**

and the *then* and *else* branches are of type τ

Simply-Typed Lambda Calculus

Types	$\tau ::= \texttt{bool} \mid \tau \longrightarrow \tau$
Values	$v ::= \texttt{true} \mid \texttt{false} \mid x \mid \lambda x : \tau \,.\, t$
Terms	$t ::= v \mid t\ t \mid \texttt{if}\ t\ \texttt{then}\ t\ \texttt{else}\ t$
Context	$\Gamma ::= \emptyset \mid \Gamma, x : \tau$

$$\frac{}{\Gamma \vdash \texttt{true} : \texttt{bool}}\ \text{t-true}$$

$$\frac{}{\Gamma \vdash \texttt{false} : \texttt{bool}}\ \text{t-false} \qquad \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ \text{t-var}$$

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1 \,.\, t) : \tau_1 \longrightarrow \tau_2}\ \text{t-abs}$$

$$\frac{\Gamma \vdash t_1 : \tau_1 \longrightarrow \tau_2 \quad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\ t_2 : \tau_2}\ \text{t-app} \qquad \frac{\Gamma \vdash t_1 : \texttt{bool} \quad \Gamma \vdash t_2 : \tau \quad \Gamma \vdash t_3 : \tau}{\Gamma \vdash \texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 : \tau}\ \text{t-cond}$$

This type system is safe but rejects programs that work under call-by-value:

$$(\lambda f : ?\,.\, f\ f\ \texttt{true})\,(\lambda x : ?\,.\, x)$$

Note that the call-by-value semantics ignores types (statically typed)

What is the type of $(\lambda x \,.\, x\, x)\,(\lambda y \,.\, y\, y)$?

$$\Gamma \vdash (\lambda x : \tau_1 \,.\, x\, x)\,(\lambda y : \tau_2 \,.\, y\, y) : ?$$

Type Judgments

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ \text{t-var} \qquad \frac{\Gamma \vdash t_1 : \tau_1 \longrightarrow \tau_2 \quad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\, t_2 : \tau_2}\ \text{t-app} \qquad \frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1 \,.\, t) : \tau_1 \longrightarrow \tau_2}\ \text{t-abs}$$

What is the type of $(\lambda x\,.\,x\,x)\,(\lambda y\,.\,y\,y)$?

$$\dfrac{\Gamma \vdash (\lambda x : \tau_1\,.\,x\,x) : ?\ \longrightarrow ? \qquad\qquad \Gamma \vdash (\lambda y : \tau_2\,.\,y\,y) : ?}{\Gamma \vdash (\lambda x : \tau_1\,.\,x\,x)\,(\lambda y : \tau_2\,.\,y\,y) : ?}\ \text{t-app}$$

Type Judgments

$$\dfrac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ \text{t-var} \qquad \dfrac{\Gamma \vdash t_1 : \tau_1 \longrightarrow \tau_2 \quad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\,t_2 : \tau_2}\ \text{t-app} \qquad \dfrac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1\,.\,t) : \tau_1 \longrightarrow \tau_2}\ \text{t-abs}$$

Apply t-app

What is the type of $(\lambda x \,.\, x\, x)\,(\lambda y \,.\, y\, y)$?

$$\dfrac{\dfrac{\Gamma, x : \tau_1 \vdash x\, x : ?}{\Gamma \vdash (\lambda x : \tau_1 \,.\, x\, x) : \tau_1 \longrightarrow ?}\ \text{t-abs} \qquad \Gamma \vdash (\lambda y : \tau_2 \,.\, y\, y) : \tau_1}{\Gamma \vdash (\lambda x : \tau_1 \,.\, x\, x)\,(\lambda y : \tau_2 \,.\, y\, y) : ?}\ \text{t-app}$$

Type Judgments

$$\dfrac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ \text{t-var} \qquad \dfrac{\Gamma \vdash t_1 : \tau_1 \longrightarrow \tau_2 \qquad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\, t_2 : \tau_2}\ \text{t-app} \qquad \dfrac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1 \,.\, t) : \tau_1 \longrightarrow \tau_2}\ \text{t-abs}$$

The left term $\lambda x : \tau_1 \,.\, x\, x$ is an abstraction: apply t-abs

Add the type of the (annotated) argument to the context

Note that the type of the argument fixes part of the type of the abstraction and its argument

What is the type of $(\lambda x\ .\ x\, x)\, (\lambda y\ .\ y\, y)$?

$$\dfrac{\dfrac{\dfrac{\Gamma, x : \tau_1 \vdash x : ?\ \longrightarrow ? \qquad \Gamma, x : \tau_1 \vdash x : ?}{\Gamma, x : \tau_1 \vdash x\, x : ?}\,\text{t-app}}{\Gamma \vdash (\lambda x : \tau_1\ .\ x\, x) : \tau_1 \longrightarrow ?}\,\text{t-abs} \qquad \Gamma \vdash (\lambda y : \tau_2\ .\ y\, y) : \tau_1}{\Gamma \vdash (\lambda x : \tau_1\ .\ x\, x)\, (\lambda y : \tau_2\ .\ y\, y) : ?}\,\text{t-app}$$

Type Judgments

$$\dfrac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\,\text{t-var} \qquad \dfrac{\Gamma \vdash t_1 : \tau_1 \longrightarrow \tau_2 \quad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\, t_2 : \tau_2}\,\text{t-app} \qquad \dfrac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1\ .\ t) : \tau_1 \longrightarrow \tau_2}\,\text{t-abs}$$

Apply t-app to $x\, x$

What is the type of $(\lambda x \,.\, x\,x)\,(\lambda y \,.\, y\,y)$?

$$\dfrac{\dfrac{\dfrac{\Gamma, x : \tau_1 \vdash x : ? \;\longrightarrow ? \qquad \dfrac{x : \tau_1 \in \Gamma, x : \tau_1}{\Gamma, x : \tau_1 \vdash x : ?}\,\text{t-var}}{\Gamma, x : \tau_1 \vdash x\,x : ?}\,\text{t-app}}{\Gamma \vdash (\lambda x : \tau_1 \,.\, x\,x) : \tau_1 \longrightarrow ?}\,\text{t-abs} \qquad \Gamma \vdash (\lambda y : \tau_2 \,.\, y\,y) : \tau_1}{\Gamma \vdash (\lambda x : \tau_1 \,.\, x\,x)\,(\lambda y : \tau_2 \,.\, y\,y) : ?}\,\text{t-app}$$

Type Judgments

$$\dfrac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\,\text{t-var} \qquad \dfrac{\Gamma \vdash t_1 : \tau_1 \longrightarrow \tau_2 \quad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\,t_2 : \tau_2}\,\text{t-app} \qquad \dfrac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1 \,.\, t) : \tau_1 \longrightarrow \tau_2}\,\text{t-abs}$$

Consider the right term x first

Apply t-var to the variable x

The extended context $\Gamma, x : \tau_1$ tells us $x : \tau_1$

What is the type of $(\lambda x\,.\,x\,x)\,(\lambda y\,.\,y\,y)$?

$$\dfrac{\dfrac{\Gamma, x : \tau_1 \vdash x : ? \;\to ? \qquad \dfrac{x : \tau_1 \in \Gamma, x : \tau_1}{\Gamma, x : \tau_1 \vdash x : \tau_1}\,\text{t-var}}{\dfrac{\Gamma, x : \tau_1 \vdash x\,x : ?}{\Gamma \vdash (\lambda x : \tau_1\,.\,x\,x) : \tau_1 \to ?}\,\text{t-abs}}\,\text{t-app} \qquad \Gamma \vdash (\lambda y : \tau_2\,.\,y\,y) : \tau_1}{\Gamma \vdash (\lambda x : \tau_1\,.\,x\,x)\,(\lambda y : \tau_2\,.\,y\,y) : ?}\,\text{t-app}$$

Type Judgments

$$\dfrac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\,\text{t-var} \qquad \dfrac{\Gamma \vdash t_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\,t_2 : \tau_2}\,\text{t-app} \qquad \dfrac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1\,.\,t) : \tau_1 \to \tau_2}\,\text{t-abs}$$

t-var tells us x is well-typed with the type τ_1

What is the type of $(\lambda x \,.\, x\, x)\,(\lambda y \,.\, y\, y)$?

$$
\dfrac{\dfrac{\Gamma, x : \tau_1 \vdash x : \tau_1 \longrightarrow ? \qquad \dfrac{x : \tau_1 \in \Gamma, x : \tau_1}{\Gamma, x : \tau_1 \vdash x : \tau_1}\,\text{t-var}}{\dfrac{\Gamma, x : \tau_1 \vdash x\, x : ?}{\Gamma \vdash (\lambda x : \tau_1 \,.\, x\, x) : \tau_1 \longrightarrow ?}\,\text{t-abs}}\,\text{t-app} \qquad \Gamma \vdash (\lambda y : \tau_2 \,.\, y\, y) : \tau_1}{\Gamma \vdash (\lambda x : \tau_1 \,.\, x\, x)\,(\lambda y : \tau_2 \,.\, y\, y) : ?}\,\text{t-app}
$$

Type Judgments

$$
\dfrac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\,\text{t-var} \qquad \dfrac{\Gamma \vdash t_1 : \tau_1 \longrightarrow \tau_2 \quad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\, t_2 : \tau_2}\,\text{t-app} \qquad \dfrac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1 \,.\, t) : \tau_1 \longrightarrow \tau_2}\,\text{t-abs}
$$

t-app tells us the type of the body must be a function from the type of the argument

What is the type of $(\lambda x\ .\ x\, x)\,(\lambda y\ .\ y\, y)$?

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \tau_1 \in \Gamma, x : \tau_1}{\Gamma, x : \tau_1 \vdash x : \tau_1 \longrightarrow ?}\text{t-var} \quad \dfrac{x : \tau_1 \in \Gamma, x : \tau_1}{\Gamma, x : \tau_1 \vdash x : \tau_1}\text{t-var}}{\Gamma, x : \tau_1 \vdash x\, x : ?}\text{t-app}}{\Gamma \vdash (\lambda x : \tau_1\ .\ x\, x) : \tau_1 \longrightarrow ?}\text{t-abs} \qquad \Gamma \vdash (\lambda y : \tau_2\ .\ y\, y) : \tau_1}{\Gamma \vdash (\lambda x : \tau_1\ .\ x\, x)\,(\lambda y : \tau_2\ .\ y\, y) : ?}\text{t-app}$$

Type Judgments

$$\dfrac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\text{t-var} \qquad \dfrac{\Gamma \vdash t_1 : \tau_1 \longrightarrow \tau_2 \quad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\, t_2 : \tau_2}\text{t-app} \qquad \dfrac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1\ .\ t) : \tau_1 \longrightarrow \tau_2}\text{t-abs}$$

The left term is x, so t-var applies

The context tells us x is well-typed with type τ_1

What is the type of $(\lambda x \,.\, x\,x)\,(\lambda y \,.\, y\,y)$?

$$\dfrac{\dfrac{\dfrac{\dfrac{x : \tau_1 \in \Gamma, x : \tau_1}{\Gamma, x : \tau_1 \vdash x : \tau_1 \longrightarrow ?}\,\text{t-var} \qquad \dfrac{x : \tau_1 \in \Gamma, x : \tau_1}{\Gamma, x : \tau_1 \vdash x : \tau_1}\,\text{t-var}}{\Gamma, x : \tau_1 \vdash x\,x : ?}\,\text{t-app}}{\Gamma \vdash (\lambda x : \tau_1 \,.\, x\,x) : \tau_1 \longrightarrow ?}\,\text{t-abs} \qquad \Gamma \vdash (\lambda y : \tau_2 \,.\, y\,y) : \tau_1}{\Gamma \vdash (\lambda x : \tau_1 \,.\, x\,x)\,(\lambda y : \tau_2 \,.\, y\,y) : ?}\,\text{t-app}$$

Type Judgments

$$\dfrac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\,\text{t-var} \qquad \dfrac{\Gamma \vdash t_1 : \tau_1 \longrightarrow \tau_2 \qquad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\,t_2 : \tau_2}\,\text{t-app} \qquad \dfrac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1 \,.\, t) : \tau_1 \longrightarrow \tau_2}\,\text{t-abs}$$

But because our types are finite, there is no type such that $\tau_1 = \tau_1 \longrightarrow \tau_2$

It follows $(\lambda x \,.\, x\,x)\,(\lambda x \,.\, x\,x)$ is not well-typed

What is the type of $(\lambda x\ .\ x\ x)\,(\lambda y\ .\ y\ y)$?

$$\cfrac{\cfrac{\cfrac{\cfrac{x : \tau_1 \in \Gamma, x : \tau_1}{\Gamma, x : \tau_1 \vdash x : \tau_1 \longrightarrow ?}\,\text{t-var} \qquad \cfrac{x : \tau_1 \in \Gamma, x : \tau_1}{\Gamma, x : \tau_1 \vdash x : \tau_1}\,\text{t-var}}{\Gamma, x : \tau_1 \vdash x\,x : ?}\,\text{t-app}}{\Gamma \vdash (\lambda x : \tau_1\ .\ x\,x) : \tau_1 \longrightarrow ?}\,\text{t-abs} \qquad \Gamma \vdash (\lambda y : \tau_2\ .\ y\,y) : \tau_1}{\Gamma \vdash (\lambda x : \tau_1\ .\ x\,x)\,(\lambda y : \tau_2\ .\ y\,y) : ?}\,\text{t-app}$$

Type Judgments

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\,\text{t-var} \qquad \frac{\Gamma \vdash t_1 : \tau_1 \longrightarrow \tau_2 \quad \Gamma \vdash t_2 : \tau_1}{\Gamma \vdash t_1\,t_2 : \tau_2}\,\text{t-app} \qquad \frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1\ .\ t) : \tau_1 \longrightarrow \tau_2}\,\text{t-abs}$$

More powerfully, this simply typed lambda calculus is *strongly normalizing*:

If $\Gamma \vdash t : \tau$ then there is a value v such that $t \longrightarrow^* v$.

Well-typed terms always terminate; *this simply typed lambda calculus is not Turing-complete*