The λ-calculus

Lecturer: John Wickerson

Imperial College London

A tiny bit of Java

$$\begin{aligned} \mathit{expr} ::= &\ \mathit{expr}\ \texttt{+}\ \mathit{expr} \\ \mid &\ \mathit{expr}\ \texttt{<}\ \mathit{expr} \\ \mid &\ \mathrm{x} \\ \mid &\ \mathrm{n} \end{aligned}$$

$$\begin{aligned} \mathit{block} ::= &\ \mathit{cmd} \\ \mid &\ \texttt{\{}\ \mathit{cmd} \ldots \mathit{cmd}\ \texttt{\}} \end{aligned}$$

$$\begin{aligned} \mathit{cmd} ::= &\ \mathit{expr}\texttt{;} \\ \mid &\ \texttt{if(}\mathit{cmd}\texttt{)}\ \mathit{block}\ \texttt{else}\ \mathit{block}\texttt{;} \\ \mid &\ \texttt{if(}\mathit{cmd}\texttt{)}\ \mathit{block}\texttt{;} \\ \mid &\ \texttt{try\{}\mathit{cmd}\texttt{\}}\ \texttt{finally\{}\mathit{cmd} \ldots \mathit{cmd}\texttt{\};} \\ \mid &\ \texttt{while(}\mathit{cmd}\texttt{)}\ \mathit{block}\texttt{;} \\ \mid &\ \ldots \end{aligned}$$

A tiny bit of Haskell

$$
\begin{aligned}
\mathit{expr} ::= \;& \mathit{expr}\ \texttt{+}\ \mathit{expr} \\
\mid\;& \mathit{expr}\ \texttt{<}\ \mathit{expr} \\
\mid\;& \mathrm{x} \\
\mid\;& \mathrm{n} \\
\mid\;& \texttt{let}\ \mathrm{x}\ \texttt{=}\ \mathit{expr}\ \texttt{in}\ \mathit{expr} \\
\mid\;& \texttt{if}\ \mathit{expr}\ \texttt{then}\ \mathit{expr}\ \texttt{else}\ \mathit{expr} \\
\mid\;& \mathit{expr}\ \mathit{expr} \\
\mid\;& \texttt{\\}\mathrm{x}\ \texttt{.}\ \mathit{expr} \\
\mid\;& \mathit{expr}\ \texttt{:}\ \mathit{expr} \\
\mid\;& \texttt{[]} \\
\mid\;& \texttt{true} \\
\mid\;& \texttt{false} \\
\mid\;& \texttt{(}\mathit{expr}\texttt{,}\ \mathit{expr}\texttt{)}
\end{aligned}
$$

The whole λ-calculus

$$
\begin{aligned}
M ::= {} & \lambda x.\ M \\
\mid {} & M\ M \\
\mid {} & x
\end{aligned}
$$

The λ-calculus is …

- The simplest programming language in the world

- A training ground for studying other programming languages

Outline

1. Syntax (free variables, α-equivalence, substitution)
2. Semantics (β-reduction, confluence, reduction strategies)
3. Usage (encoding arithmetic, recursion)

Examples

$$M ::= \lambda x. M \mid M\ M \mid x$$

- $\lambda x.\ x$
- $\lambda x.\ \mathbf{y}$
- $(\lambda x.\ x)(\lambda y.\ y)$
- $\lambda x.\ \lambda y.\ \lambda z.\ x$
- $(\lambda x.\ x)(\lambda y.\ \lambda z.\ \mathbf{x}\ (y\ z))(\lambda x.\ \mathbf{y}\ x\ x)$

$$\lambda x.\ \lambda y.\ \lambda z.\ M = \lambda xyz.\ M$$

$$M_1\ M_2\ M_3 = (M_1\ M_2)\ M_3$$

KEY CONCEPT

$$\lambda x.\ x\ y$$

binder

bound variable

free variable

Free variables

- Let $\mathrm{FV}(M)$ denote the set of free variables in the λ-term M.
- For instance: $\mathrm{FV}(\lambda x.\ y) = \{y\}$.
- If $\mathrm{FV}(M) = \varnothing$ then we say M is "closed".

α-equivalence

- For example: $(\lambda x.\ x) =_\alpha (\lambda y.\ y)$
- **`int i; for(i=0; i<5; i++) x+=i;`**
- $(\lambda x.\ x\ (\lambda y.\ y)\ \mathbf{y}) =_\alpha?\ (\lambda y.\ y\ (\lambda x.\ x)\ \mathbf{x})$ 😡
- $(\lambda x.\ x\ (\lambda y.\ y)\ \mathbf{y}) =_\alpha?\ (\lambda y.\ y\ (\lambda x.\ x)\ y)$ 😡
- $(\lambda x.\ x\ (\lambda y.\ y)\ \mathbf{y}) =_\alpha?\ (\lambda w.\ w\ (\lambda w.\ w)\ \mathbf{y})$ 😌
  $(\lambda z.\ z\ (\lambda y.\ y)\ \mathbf{y}) =_\alpha?\ (\lambda z.\ z\ (\lambda w.\ w)\ \mathbf{y})$
  $(\lambda z.\ z\ (\lambda z.\ z)\ \mathbf{y}) =_\alpha?\ (\lambda z.\ z\ (\lambda z.\ z)\ \mathbf{y})$

α-equivalence

$$M ::= \lambda x.M \mid M\ M \mid x$$

$$\frac{}{x =_\alpha x} \qquad \frac{M =_\alpha M' \qquad N =_\alpha N'}{M\ N =_\alpha M'\ N'}$$

$$\frac{z \notin FV(M) \cup FV(N) \qquad M[z/x] =_\alpha N[z/y]}{(\lambda x.\ M) =_\alpha (\lambda y.\ N)}$$

Substitution

- Replace `e` with `2.718` in

```
try {
  f.writeFloat(e ^ 2);
} catch (IOException e) {
  e.printStackTrace();
}
```

Substitution

```
let x=4*y in let y=w+5 in x*y
```

Substitution

```
let y=w+5 in 4*y*y
```
😡

Substitution

```
let x=4*y in let z=w+5 in x*z
```

Substitution

```
let z=w+5 in 4*y*z
```
😌

Substitution

$$M ::= \lambda x.M \mid M\ M \mid x$$

- $y[M/x] = \begin{cases} M & \text{if } y{=}x \\ y & \text{if } y{\neq}x \end{cases}$

- $(\lambda y.\ N)[M/x] = \begin{cases} \lambda y.\ N & \text{if } y{=}x \\ \lambda z.\ N[z/y][M/x] & \text{if } y{\neq}x \end{cases}$

  where $z \notin FV(M) \cup (FV(N) - \{y\}) \cup \{x\}$

- $(N_1\ N_2)[M/y] = (N_1[M/y])(N_2[M/y])$

KEY CONCEPTS

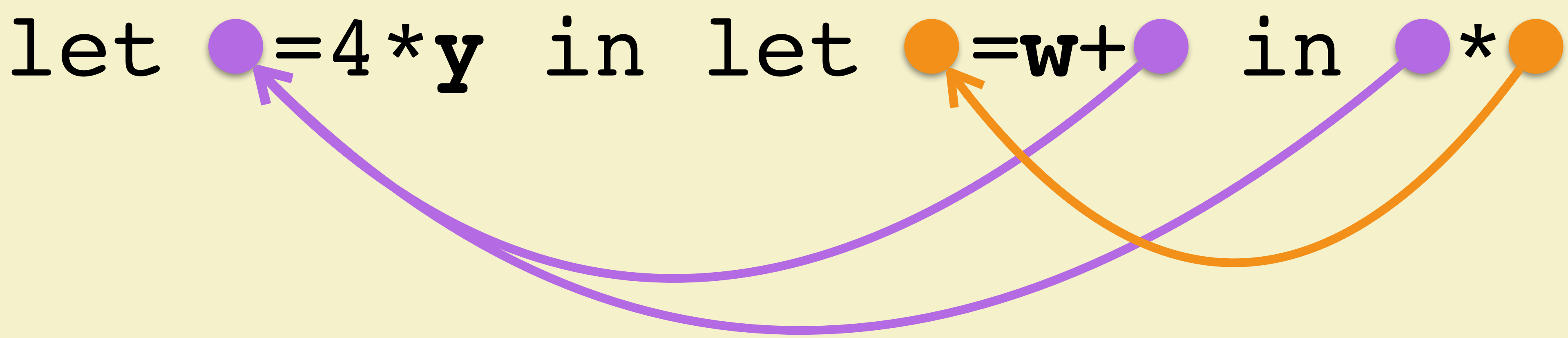
α-equivalence

capture-avoiding substitution

BONUS

DeBruijn indices

```
let x=4*y in let z=w+x in x*z
```



BONUS

DeBruijn indices

- $\lambda x.\ x \implies \lambda.\ 0$
- $\lambda x.\ \mathbf{y} \implies \lambda.\ \mathbf{y}$
- $(\lambda x.\ x)(\lambda y.\ y) \implies (\lambda.\ 0)(\lambda.\ 0)$
- $\lambda x.\ \lambda y.\ \lambda z.\ x \implies \lambda.\ \lambda.\ \lambda.\ 2$
- $(\lambda x.\ x)(\lambda y.\ \lambda z.\ \mathbf{x}\ (y\ z))(\lambda x.\ \mathbf{y}\ x\ x)$
  $\implies (\lambda.\ 0)(\lambda.\ \lambda.\ \mathbf{x}\ (1\ 0))(\lambda.\ \mathbf{y}\ 0\ 0)$

Outline

✓ 1. Syntax (free variables, α-equivalence, substitution)

2. Semantics (β-reduction, confluence, reduction strategies)

3. Usage (encoding arithmetic, recursion)

Java semantics

$$\frac{(\mathrm{C}, \sigma) \longrightarrow (\mathrm{C}', \sigma')}{(\texttt{if(}\mathrm{C}\texttt{)}\mathrm{S}, \sigma) \longrightarrow (\texttt{if(}\mathrm{C}'\texttt{)}\mathrm{S}, \sigma')}$$

$$\frac{}{(\texttt{if(true)}\mathrm{S}, \sigma) \longrightarrow (\mathrm{S}, \sigma)}$$

$$\frac{}{(\texttt{if(false)}\mathrm{S}, \sigma) \longrightarrow (\texttt{skip}, \sigma)}$$

λ-calculus semantics

$$\frac{}{\underbrace{(\lambda x.\ M)\ N}_{\text{"redex"}} \longrightarrow_\beta M[N/x]}$$

$$\frac{M \longrightarrow_\beta M'}{\lambda x.\ M \longrightarrow_\beta \lambda x.\ M'} \qquad \frac{M \longrightarrow_\beta M'}{M\ N \longrightarrow_\beta M'\ N} \qquad \frac{N \longrightarrow_\beta N'}{M\ N \longrightarrow_\beta M\ N'}$$

$$\frac{M =_\alpha M' \qquad M' \longrightarrow_\beta N' \qquad N' =_\alpha N}{M \longrightarrow_\beta N}$$

β-reduction examples

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

β-reduction examples

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

$$\downarrow \beta$$

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

β-reduction examples

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

$$\xrightarrow{\beta}\ ((\lambda x.\ y)\ z)\ ((\lambda x.\ y)\ z)$$

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

β-reduction examples

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

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

$$\xrightarrow{\beta}\ y\ ((\lambda x.\ y)\ z)$$

$$\xrightarrow{\beta}\ y\ y$$

β-reduction examples

$$
\begin{tikzcd}
 & (\lambda x.\ x\ x)\ ((\lambda x.\ y)\ z) \arrow[d, "\beta"] & \\
 & ((\lambda x.\ y)\ z)\ ((\lambda x.\ y)\ z) \arrow[dl, "\beta"] \arrow[dr, "\beta"'] & \\
((\lambda x.\ y)\ z)\ y & & y\ ((\lambda x.\ y)\ z) \arrow[d, "\beta"] \\
 & & y\ y
\end{tikzcd}
$$

β-reduction examples

$$
\begin{tikzcd}
 & (\lambda x.\ x\ x)\ ((\lambda x.\ y)\ z) \arrow[d, "\beta"] & \\
 & ((\lambda x.\ y)\ z)\ ((\lambda x.\ y)\ z) \arrow[dl, "\beta"] \arrow[dr, "\beta"'] & \\
((\lambda x.\ y)\ z)\ y \arrow[dr, "\beta"'] & & y\ ((\lambda x.\ y)\ z) \arrow[dl, "\beta"] \\
 & y\ y &
\end{tikzcd}
$$

β-reduction examples

$$\begin{tikzcd}
 & & (\lambda x.\ x\ x)\ ((\lambda x.\ y)\ z) \arrow[dl, "\beta"'] \arrow[dr, "\beta"] & \\
 & ((\lambda x.\ y)\ z)\ ((\lambda x.\ y)\ z) \arrow[dl, "\beta"] \arrow[dr, "\beta"] & & (\lambda x.\ x\ x)\ \underline{y} \\
((\lambda x.\ y)\ z)\ y \arrow[drr, "\beta"] & & y\ ((\lambda x.\ y)\ z) \arrow[d, "\beta"] & \\
 & & y\ y &
\end{tikzcd}$$

β-reduction examples

$$
\begin{tikzcd}
 & & (\lambda x.\ x\ x)\ ((\lambda x.\ y)\ z) \arrow[dl, "\beta"] \arrow[dr, "\beta"] & \\
 & ((\lambda x.\ y)\ z)\ ((\lambda x.\ y)\ z) \arrow[dl, "\beta"] \arrow[dr, "\beta"] & & (\lambda x.\ x\ x)\ y \arrow[ddl, "\beta"] \\
((\lambda x.\ y)\ z)\ y \arrow[drr, "\beta"] & & y\ ((\lambda x.\ y)\ z) \arrow[d, "\beta"] & \\
 & & y\ y &
\end{tikzcd}
$$

KEY CONCEPTS

β-reduction:

$$(\lambda x.\ M)\ N \longrightarrow_{\beta} M[N/x]$$

Many steps of β-reduction

- Define $\longrightarrow_\beta{}^*$ as follows:

$$\frac{M =_\alpha M'}{M \longrightarrow_\beta{}^* M'} \qquad \frac{M \longrightarrow_\beta M'' \qquad M'' \longrightarrow_\beta{}^* M'}{M \longrightarrow_\beta{}^* M'}$$

Confluence

- **Theorem** (Church–Rosser). If $M \longrightarrow_\beta^* M_1$ and $M \longrightarrow_\beta^* M_2$ then there exists M' such that $M_1 \longrightarrow_\beta^* M'$ and $M_2 \longrightarrow_\beta^* M'$.

$$\begin{tikzcd}
& M \arrow[dl, "*"', "\beta"] \arrow[dr, "\beta"', "*"] & \\
M_1 \arrow[dr, dashed, "\beta"', "*"] & & M_2 \arrow[dl, dashed, "*"', "\beta"] \\
& M' &
\end{tikzcd}$$

β-reduction examples

$$
\begin{tikzcd}
 & & (\lambda x.\ x\ x)\ ((\lambda x.\ y)\ z) \arrow[dl, "\beta"'] \arrow[dr, "\beta"] & \\
 & ((\lambda x.\ y)\ z)\ ((\lambda x.\ y)\ z) \arrow[dl, "\beta"'] \arrow[dr, "\beta"] & & (\lambda x.\ x\ x)\ y \arrow[ddll, "\beta"] \\
((\lambda x.\ y)\ z)\ y \arrow[dr, "\beta"'] & & y\ ((\lambda x.\ y)\ z) \arrow[dl, "\beta"] & \\
 & y\ y & &
\end{tikzcd}
$$

β-normal form

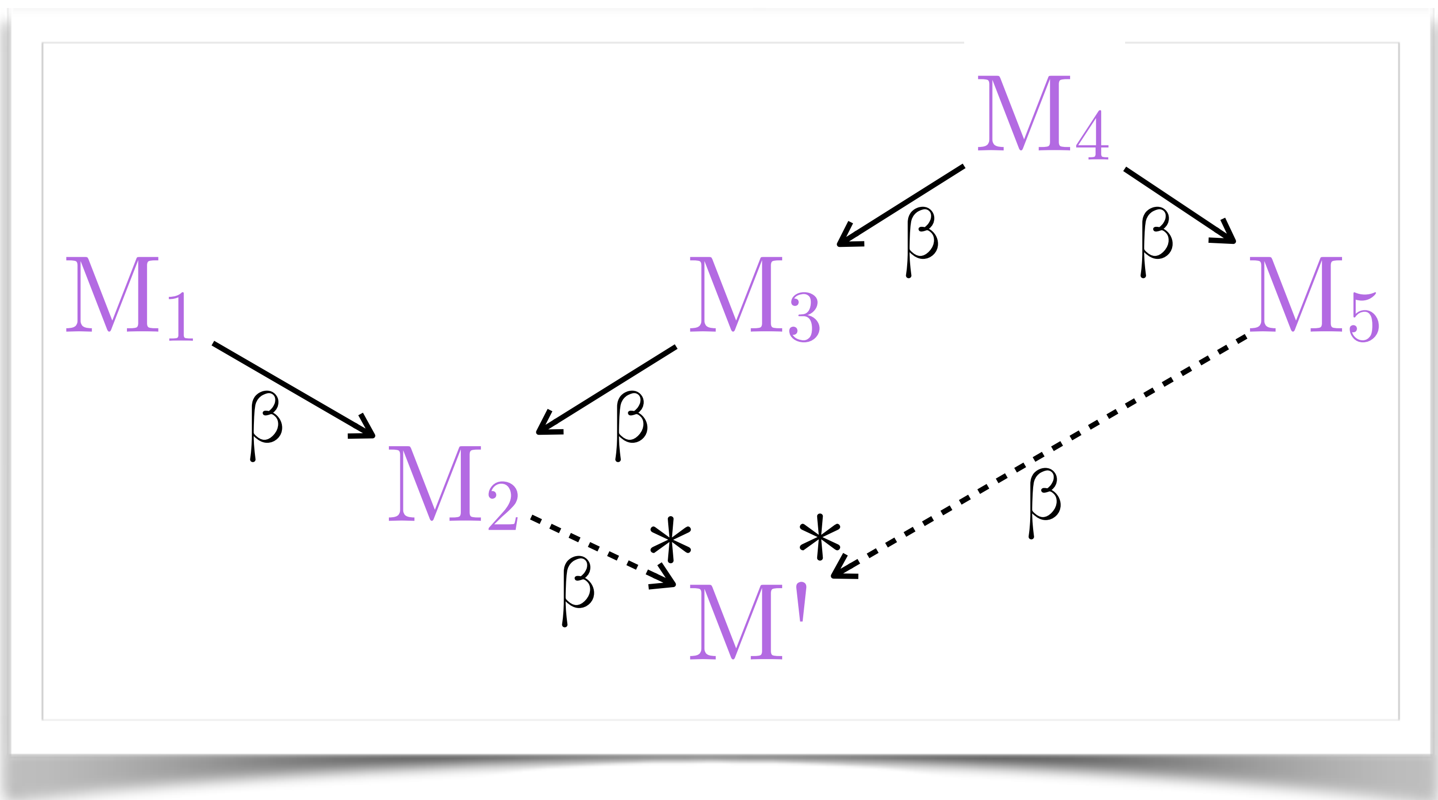
- "in β-normal form" = "contains no redexes"
- M has a β-normal form if $M \longrightarrow_\beta^* N$ for some N in β-normal form.
- **Theorem** (Uniqueness of β-normal forms). If $M \longrightarrow_\beta^* N_1$, $M \longrightarrow_\beta^* N_2$, and N_1 and N_2 are in β-normal form, then $N_1 =_\alpha N_2$.

β-normal form

- **Theorem** (Uniqueness of β-normal forms)**.** If $M \longrightarrow_\beta{}^* N_1$, $M \longrightarrow_\beta{}^* N_2$, and N_1 and N_2 are in β-normal form, then $N_1 =_\alpha N_2$.
- *Proof.* By Church–Rosser, obtain N such that $N_1 \longrightarrow_\beta{}^* N$ and $N_2 \longrightarrow_\beta{}^* N$. But N_1 and N_2 are in β-normal form, so $N_1 =_\alpha N =_\alpha N_2$.

β-equivalence

- $=_\beta$ is the smallest equivalence relation containing $\longrightarrow_\beta$.



- *Simpler version.* $M_1 =_\beta M_2$ iff there exists M' such that $M_1 \longrightarrow_\beta{}^* M'$ and $M_2 \longrightarrow_\beta{}^* M'$.

KEY CONCEPTS

β-reduction

confluence (Church–Rosser)

β-normal form

β-equivalence

Non-termination

- Some terms do not have a β-normal form.
- For instance: $(\lambda x.\ x\ x)(\lambda x.\ x\ x)$

$$(\lambda x.\ x\ x)(\lambda x.\ x\ x) \xrightarrow{\beta} (\lambda x.\ x\ x)(\lambda x.\ x\ x) \xrightarrow{\beta} \ldots$$

Possible non-termination

- Some terms might not terminate.
- For instance:

$$
\begin{tikzcd}
 & (\lambda x.\ y)((\lambda x.\ x\ x)(\lambda x.\ x\ x)) \arrow[dl, "\beta"'] \arrow[d, "\beta"] & \\
y & (\lambda x.\ y)((\lambda x.\ x\ x)(\lambda x.\ x\ x)) \arrow[dl, "\beta"'] \arrow[d, "\beta"] & \\
y & \cdots &
\end{tikzcd}
$$

Reduction strategies

$$(\lambda x.\ M)\ N$$

- Substitute N for x? $\Longrightarrow$ "call by name"
- Reduce N? $\Longrightarrow$ "call by value"
- Reduce M? $\Longrightarrow$ not usually done

Full β-reduction	Call by name	Call by value
$$\frac{M \longrightarrow_{\beta} M'}{\lambda x.\ M \longrightarrow_{\beta} \lambda x.\ M'}$$		
$$\frac{M \longrightarrow_{\beta} M'}{M\ N \longrightarrow_{\beta} M'\ N}$$	$$\frac{M \longrightarrow_{\beta N} M'}{M\ N \longrightarrow_{\beta N} M'\ N}$$	$$\frac{M \longrightarrow_{\beta V} M'}{M\ N \longrightarrow_{\beta V} M'\ N}$$
$$\frac{N \longrightarrow_{\beta} N'}{M\ N \longrightarrow_{\beta} M\ N'}$$		$$\frac{M \not\longrightarrow_{\beta V} \quad N \longrightarrow_{\beta V} N'}{M\ N \longrightarrow_{\beta V} M\ N'}$$
$$\frac{}{(\lambda x.\ M)\ N \longrightarrow_{\beta} M[N/x]}$$	$$\frac{}{(\lambda x.\ M)\ N \longrightarrow_{\beta N} M[N/x]}$$	$$\frac{N \not\longrightarrow_{\beta V}}{(\lambda x.\ M)\ N \longrightarrow_{\beta V} M[N/x]}$$

Reduction strategies

$$
\begin{tikzcd}
 & & (\lambda x.\ x\ x)\ ((\lambda x.\ y)\ z) \arrow[dl, "\beta N"'] \arrow[dr, "\beta V"] & \\
 & ((\lambda x.\ y)\ z)\ ((\lambda x.\ y)\ z) \arrow[dl] \arrow[dr, "\beta N"'] & & (\lambda x.\ x\ x)\ y \arrow[ddl, "\beta V"] \\
((\lambda x.\ y)\ z)\ y \arrow[drr] & & y\ ((\lambda x.\ y)\ z) \arrow[d, "\beta N"] & \\
 & & y\ y &
\end{tikzcd}
$$

Reduction strategies

$$\begin{tikzcd}
 & (\lambda x.\ y)((\lambda x.\ x\ x)(\lambda x.\ x\ x)) \arrow[dl, "\beta N"'] \arrow[dr, "\beta V"] & & \\
y & & (\lambda x.\ y)((\lambda x.\ x\ x)(\lambda x.\ x\ x)) \arrow[dl, "\beta N"'] \arrow[dr, "\beta V"] & \\
 & y & & \cdots
\end{tikzcd}$$

KEY CONCEPTS

call-by-name reduction
(wins if argument is unused)

call-by-value reduction
(wins if argument is used more than once)

BONUS

Extensionality

- Is β-equivalence the best notion of "equality" between λ-terms?

- We don't have $(\lambda x.\ \sin x) =_\beta \sin$.

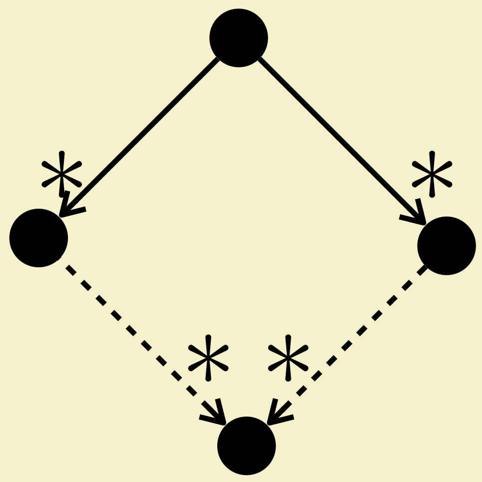
- But we do have $(\lambda x.\ \sin x)\ M =_\beta \sin\ M$, for any M.

- Add η-equivalence: $$\frac{x \notin FV(M)}{(\lambda x.\ M\ x) =_\eta M}$$

- βη-equivalence captures "equality" nicely.

BONUS

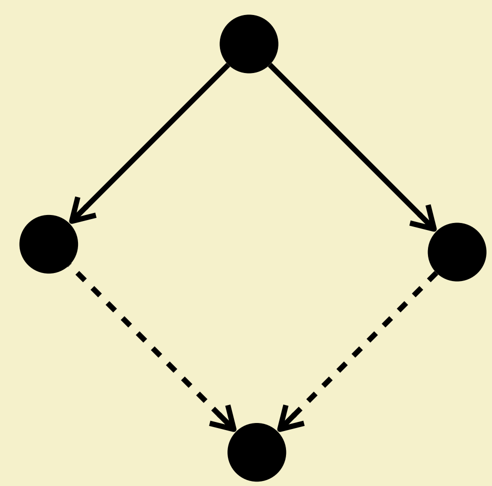
Proving Church–Rosser

β-reduction $\overset{?}{\Rightarrow}$ 

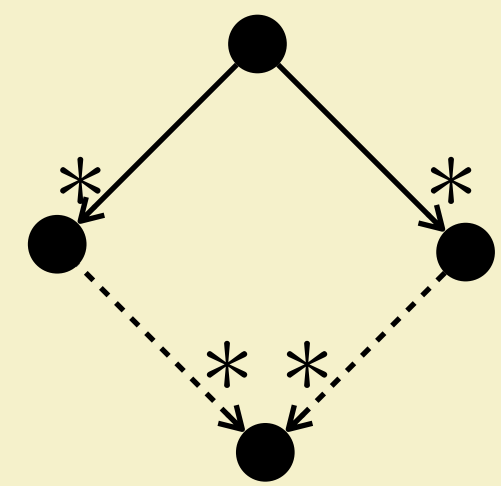
BONUS

Proving Church–Rosser

β-reduction $\overset{?}{\Rightarrow}$

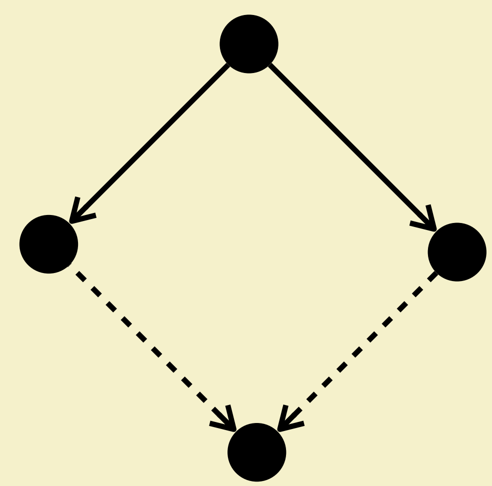
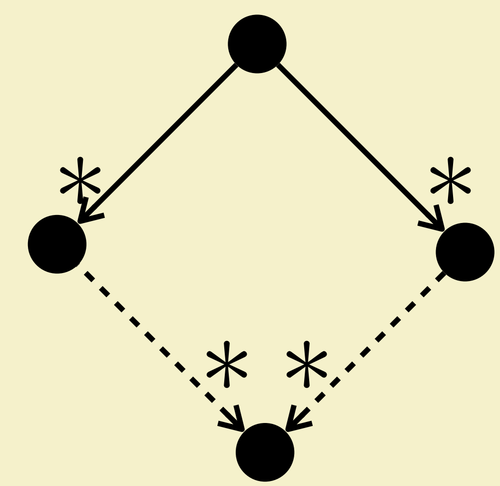


$\overset{?}{\Rightarrow}$



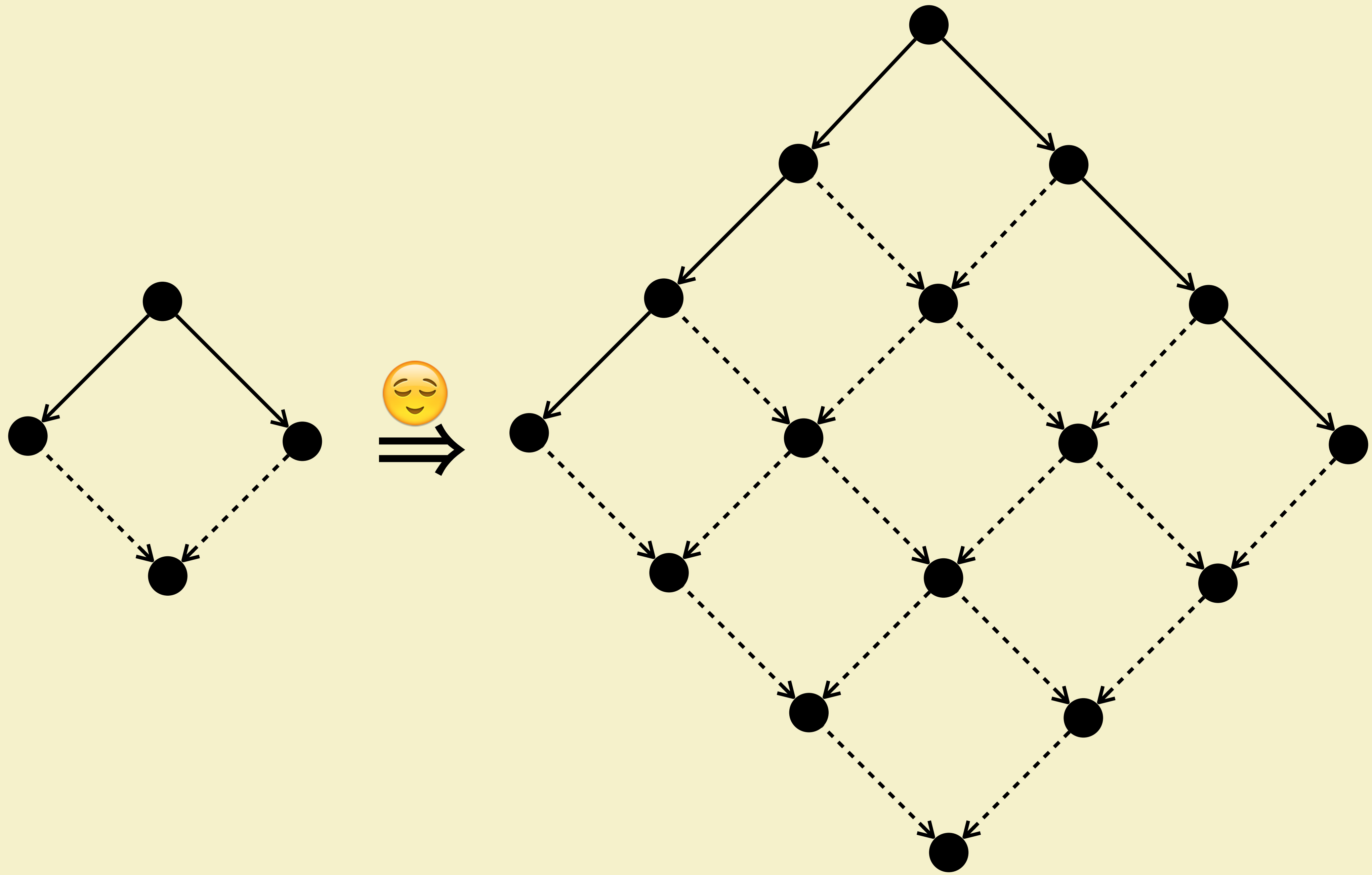
BONUS

Proving Church–Rosser

β-reduction $\overset{?}{\Longrightarrow}$  😌 $\Longrightarrow$ 

BONUS

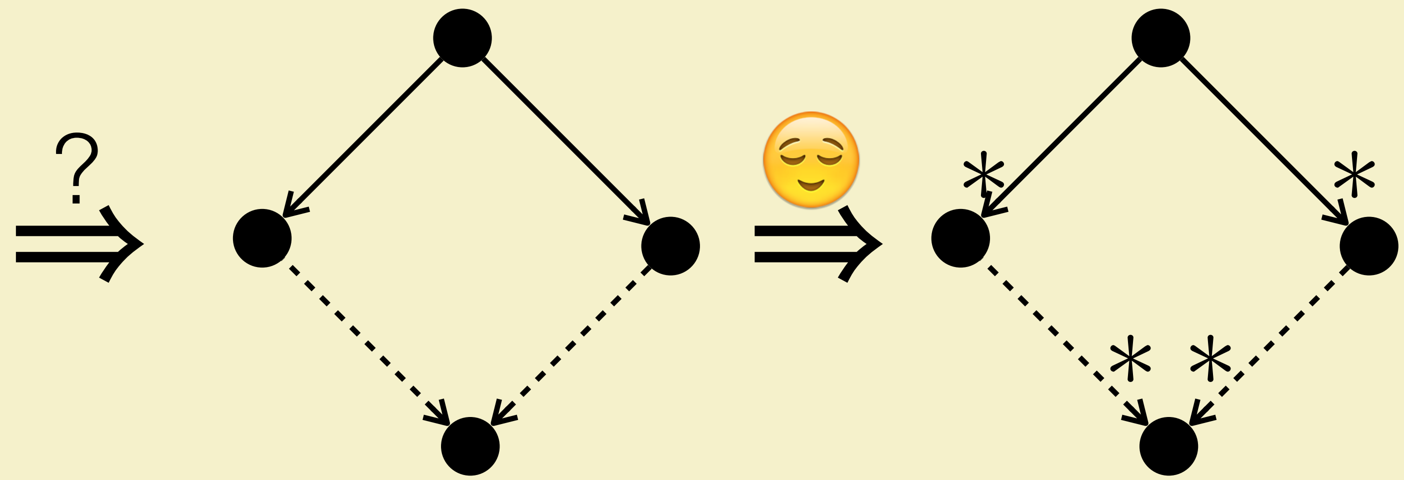
Proving Church–Rosser



BONUS

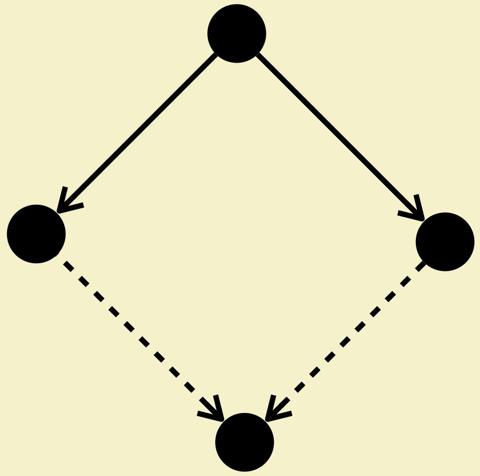
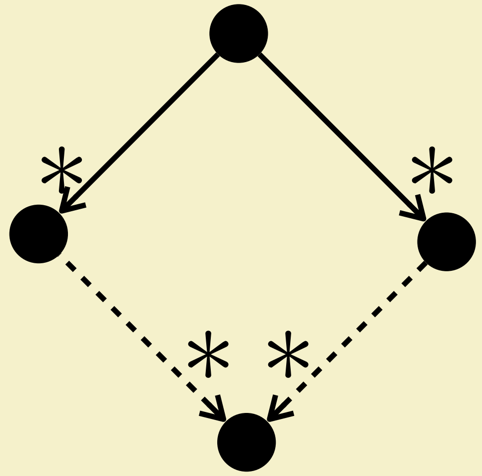
Proving Church–Rosser

β-reduction



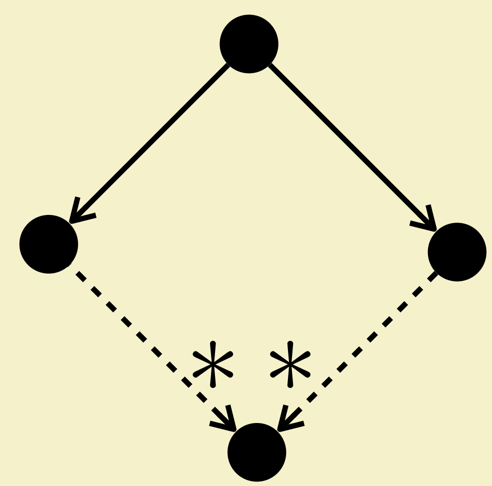
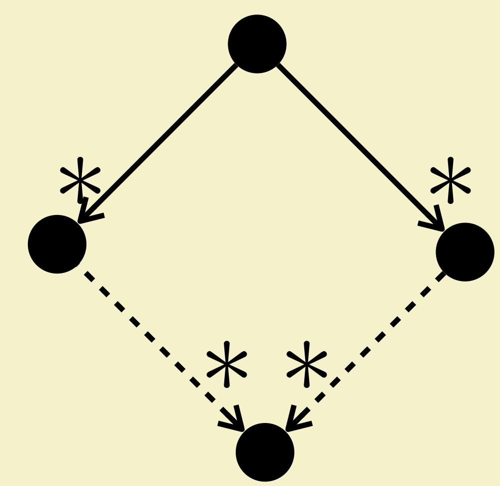
BONUS

Proving Church–Rosser

β-reduction $\overset{😡}{\Rightarrow}$  $\overset{😌}{\Rightarrow}$ 

BONUS

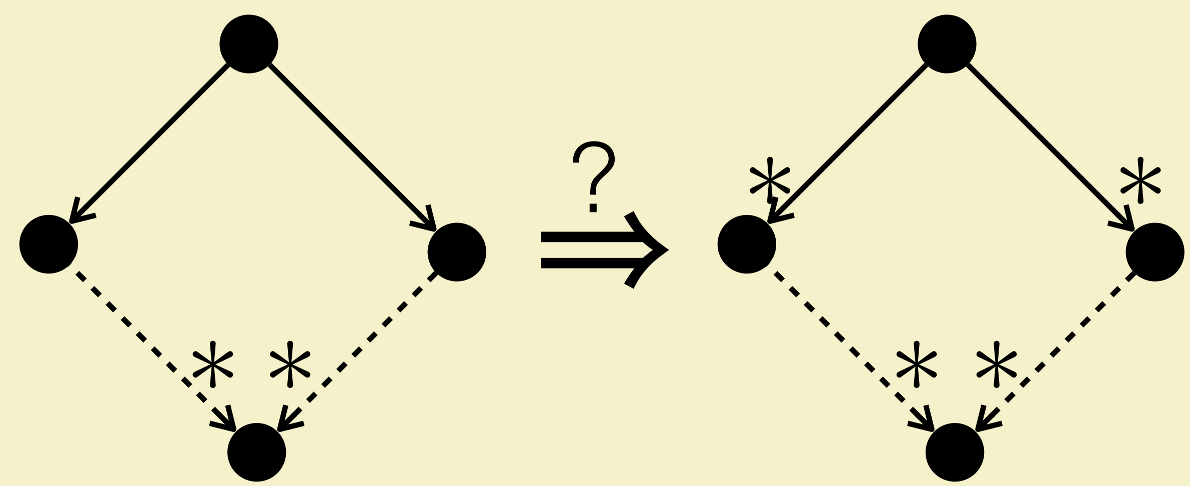
Proving Church–Rosser

β-reduction $\overset{?}{\Rightarrow}$  $\overset{?}{\Rightarrow}$ 

BONUS

Proving Church–Rosser

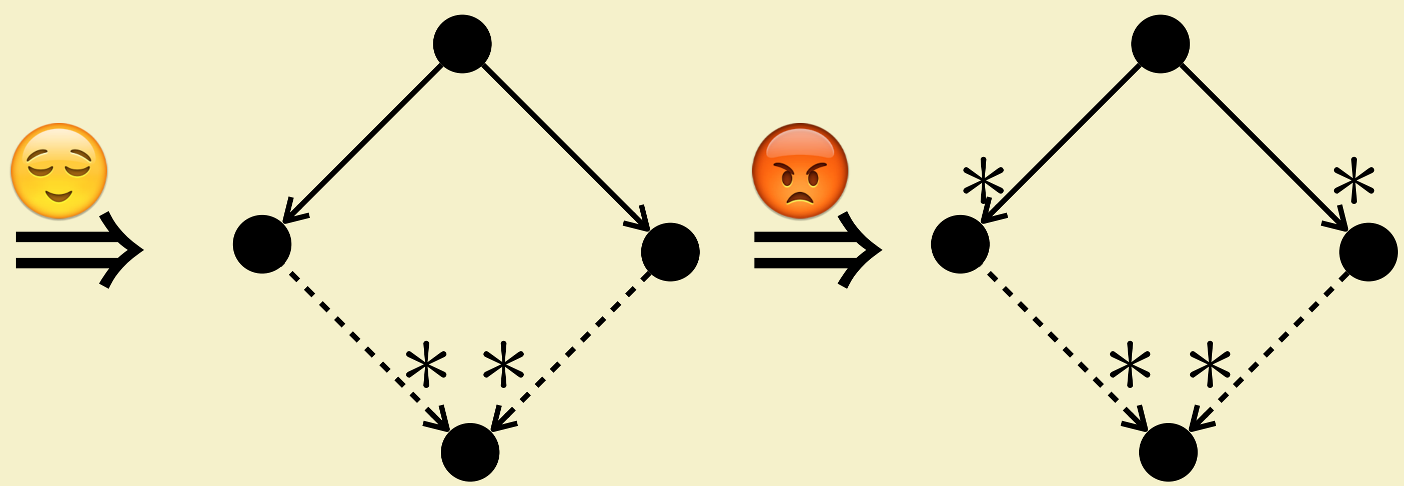
β-reduction $\Rightarrow$



BONUS

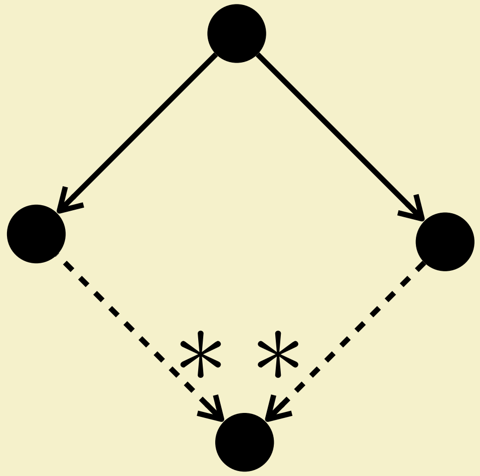
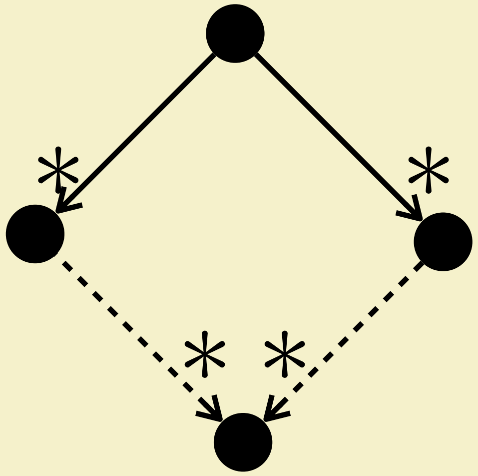
Proving Church–Rosser

β-reduction

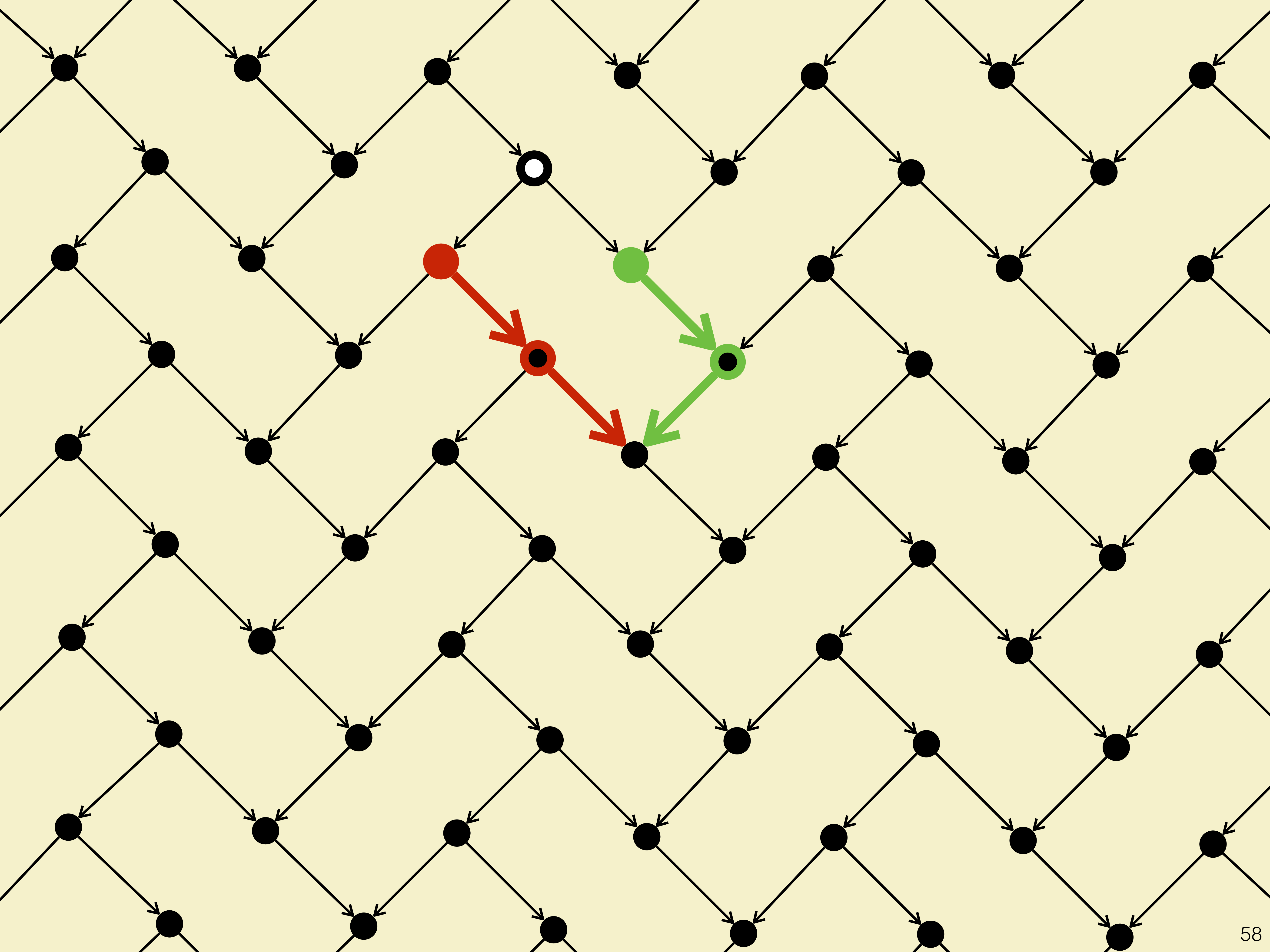


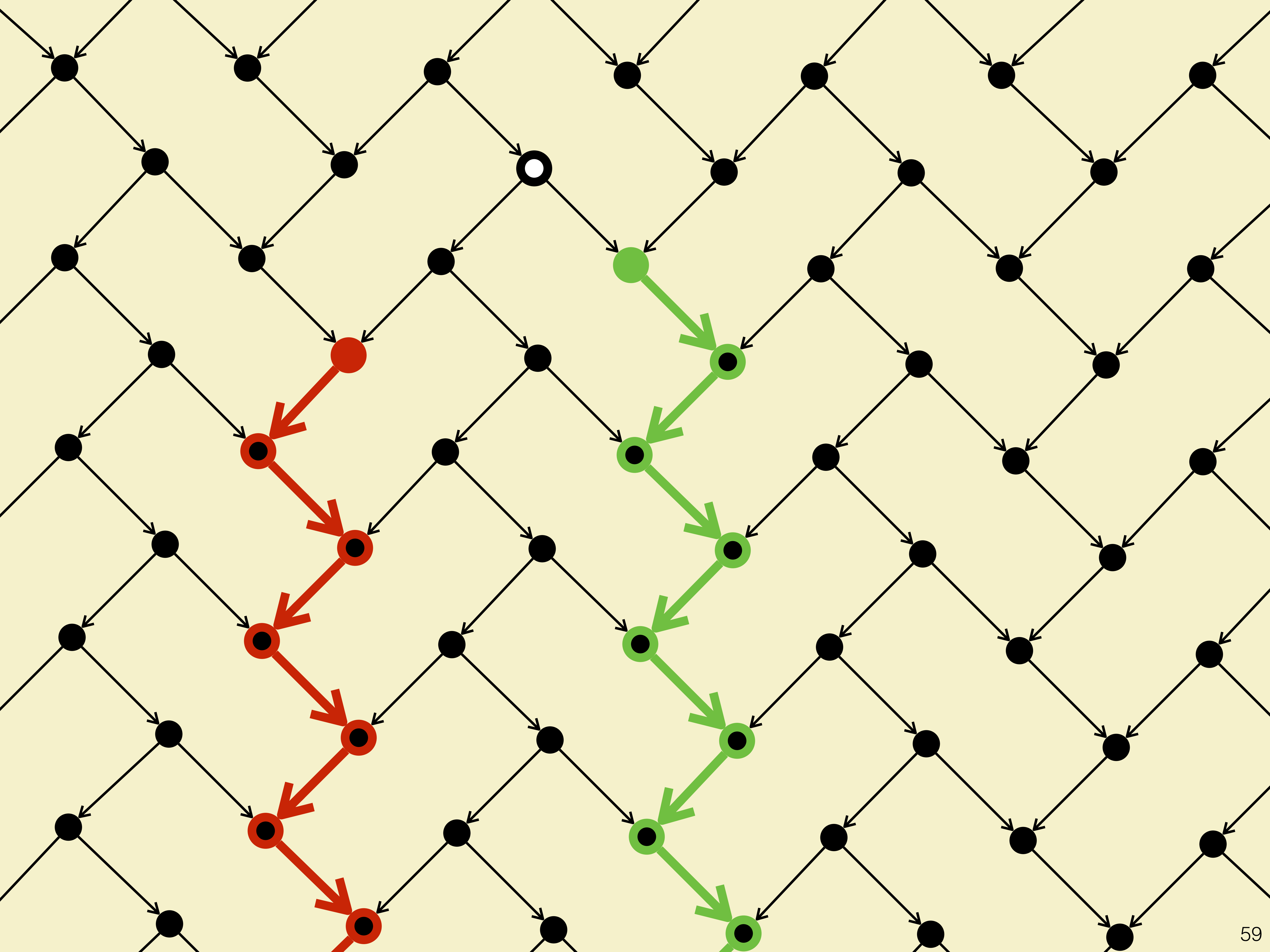
BONUS

Proving Church–Rosser

β-reduction  $\Rightarrow$   $\Rightarrow$ 

("Property A") ("Property B")

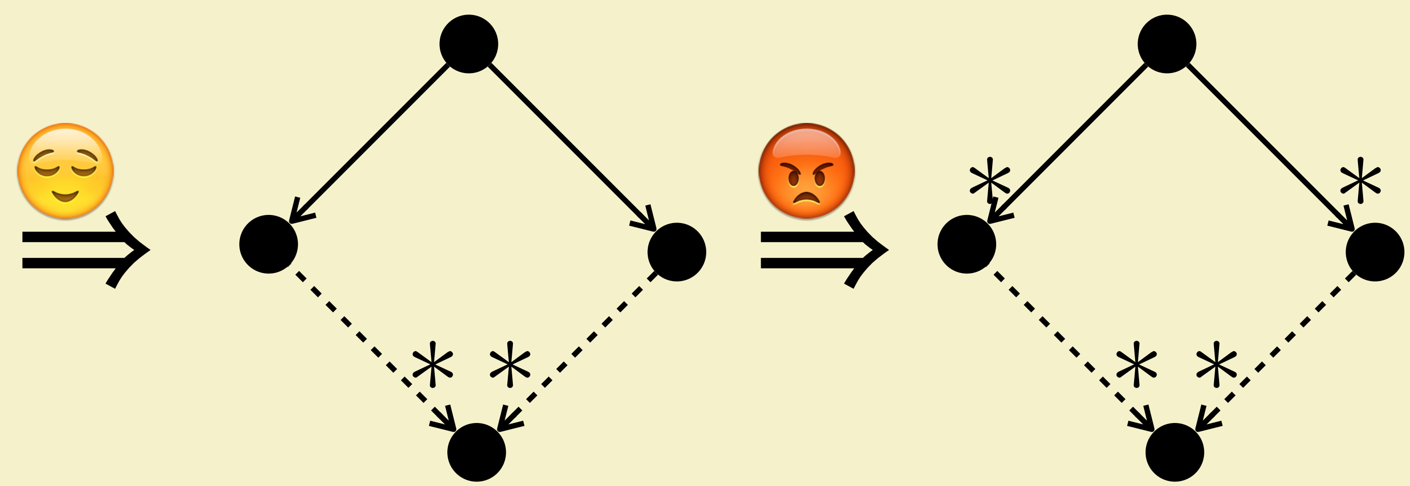




BONUS

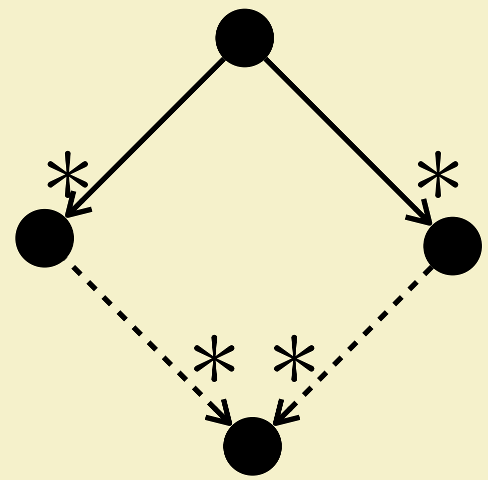
Proving Church–Rosser

β-reduction



BONUS

Proving Church–Rosser

β-reduction $\stackrel{?}{\Rightarrow}$ 

Outline

✓ 1. Syntax (free variables, α-equivalence, substitution)

✓ 2. Semantics (β-reduction, confluence, reduction strategies)

➡ 3. Usage (encoding arithmetic, recursion)

Encoding numbers

- $\underline{0} \;\equiv\; \lambda s.\ \lambda z.\ z$
- $\underline{1} \;\equiv\; \lambda s.\ \lambda z.\ s\ z$
- $\underline{2} \;\equiv\; \lambda s.\ \lambda z.\ s\ (s\ z)$
- $\underline{3} \;\equiv\; \lambda s.\ \lambda z.\ s\ (s\ (s\ z))$

$$\boxed{\underline{n} \;\equiv\; \lambda s.\ \lambda z.\ \underbrace{s\ (\ldots\ s}_{n}(z)\ldots)}$$

Encoding arithmetic

- $plus \equiv \lambda m.\ \lambda n.\ \lambda s.\ \lambda z.\ m\ s\ (n\ s\ z)$
- $mult \equiv \lambda m.\ \lambda n.\ \lambda s.\ \lambda z.\ m\ (n\ s)\ z$

$$\underline{n} \equiv \lambda s.\ \lambda z.\ \underbrace{s\ (\ldots\ s}_{n}(z)\ldots)$$

Encoding arithmetic

- $\mathit{plus} \equiv \lambda m.\ \lambda n.\ \lambda s.\ \lambda z.\ m\ s\ (n\ s\ z)$
- $\mathit{mult} \equiv \lambda m.\ \lambda n.\ \lambda s.\ \lambda z.\ m\ (n\ s)\ z$
- Exercise. Evaluate "$\mathit{plus}\ \underline{2}\ \underline{3}$" and "$\mathit{mult}\ \underline{2}\ \underline{3}$".
- $\mathit{ifz} \equiv \lambda n.\ \lambda x_1.\ \lambda x_2.\ n\ (\lambda z.\ x_2)\ x_1$
- Exercise. Find *pred* such that "$\mathit{pred}\ \underline{0} =_\beta \underline{0}$" and "$\mathit{pred}\ \underline{n+1} =_\beta \underline{n}$".

λ-definability

- A partial function $f : \mathbb{N}^k \rightharpoonup \mathbb{N}$ is λ-definable if there exists a closed λ-term F such that $f(x_1,\ldots,x_k) = y$ iff $F(\underline{x_1},\ldots,\underline{x_k}) =_\beta \underline{y}$, and $f(x_1,\ldots,x_k)\uparrow$ iff $F(\underline{x_1},\ldots,\underline{x_k})$ has no normal form.

- Church-Turing thesis:

f is λ-definable

f is "computable"

f is computable via register machine

f is computable via Turing machine

Example: factorial

- `fac n = if n=0 then 1 else n*fac(n-1)`
- $\mathit{fac} \;=_\beta\; \lambda \mathrm{n}.\ \mathit{ifz}\, \mathrm{n}\, \underline{1}\, (\mathit{mult}\, \mathrm{n}\, (\mathit{fac}\, (\mathit{pred}\, \mathrm{n})))$

Encoding recursion

- Let $\mathit{fix} \equiv (\lambda x.\ \lambda y.\ y\ (x\ x\ y))(\lambda x.\ \lambda y.\ y\ (x\ x\ y))$.
- Observe: $\mathit{fix}\ M \longrightarrow_{\beta}{}^{*}\ M\ (\mathit{fix}\ M)$.
- [Recall: x is a fixpoint of f whenever $x = f(x)$.]
- This means: $\mathit{fix}\ M$ is a fixpoint of M.

Example: factorial

- `fac n = if n=0 then 1 else n*fac(n-1)`

- $\mathit{fac} =_{\beta} \lambda \mathrm{n}.\ \mathit{ifz}\ \mathrm{n}\ \underline{1}\ (\mathit{mult}\ \mathrm{n}\ (\mathit{fac}\ (\mathit{pred}\ \mathrm{n})))$

- $\ldots =_{\beta} (\lambda \mathrm{f}.\ \lambda \mathrm{n}.\ \mathit{ifz}\ \mathrm{n}\ \underline{1}\ (\mathit{mult}\ \mathrm{n}\ (\mathrm{f}\ (\mathit{pred}\ \mathrm{n}))))\ \mathit{fac}$

- $\mathit{fac} \equiv \mathit{fix}\ (\lambda \mathrm{f}.\ \lambda \mathrm{n}.\ \mathit{ifz}\ \mathrm{n}\ \underline{1}\ (\mathit{mult}\ \mathrm{n}\ (\mathrm{f}\ (\mathit{pred}\ \mathrm{n}))))$

- Exercise. Evaluate "$\mathit{fac}\ \underline{2}$".

KEY CONCEPTS

we can encode...

numbers and arithmetic

if-statements

recursion

BONUS

Combinators

- Lambda calculus: $M ::= \lambda x.\ M \mid M\ M \mid x$

- Can we restrict even further? Yes, we can even get rid of variables!

- Let $S \equiv \lambda x.\ \lambda y.\ \lambda z.\ (x\ z)\ (y\ z)$
and $K \equiv \lambda x.\ \lambda y.\ x$.

- $M ::= M\ M \mid S \mid K$

BONUS

Is λ-calculus broken?

- Define $silly \equiv (\lambda x.\ \neg\ (x\ x))(\lambda x.\ \neg\ (x\ x))$.
- Then $silly =_{\beta} \neg\ silly$.

BONUS

Adding types

- Untyped: $M ::= \lambda x.\ M \mid M\ M \mid x$
- Typed: $M ::= \lambda x{:}\tau.\ M \mid M\ M \mid x$
  where $\tau ::= \bullet \mid \tau \to \tau$
- no more *fix* function … not Turing-complete
- add *fix* to language explicitly

Outline

✓ 1. Syntax (free variables, α-equivalence, substitution)

✓ 2. Semantics (β-reduction, confluence, reduction strategies)

✓ 3. Usage (encoding arithmetic, recursion)