Shrinking the Language

• We’ve seen that \textit{with} is not really necessary when we have \textit{fun}...

• ... and \textit{rec} is not really necessary when we have \textit{fun}...

• ... and neither, it turns out, are fancy things like numbers, $\textit{+, –}$ or \textit{if0}
LC Grammar

\[
\begin{align*}
<\text{LC}> & ::= <\text{id}> \\
& \mid \{<\text{LC}> <\text{LC}>\} \\
& \mid \{\text{fun} \{<\text{id}>\} <\text{LC}>\}
\end{align*}
\]
Implementing Programs with LC

Can you write a program that produces the identity function?

```
{fun \{x\} x
```
Implementing Programs with LC

Can you write a program that produces zero?

What’s zero? I only know how to write functions!

Turing Machine programmer: What’s a function? I only know how to write 0 or 1!

We need to encode zero — instead of agreeing to write zero as 0, let’s agree to write it as

\[
\{ \text{fun} \{ f \} \{ \text{fun} \{ x \} x \} \}
\]

This encoding is the start of Church numerals...
Implementing Numbers with LC

Can you write a program that produces zero?

```
  {fun {f} {fun {x} x}}
```

... which is also the function that takes \( f \) and \( x \) and applies \( f \) to \( x \) zero times

From now on, we’ll write \texttt{zero} as shorthand for the above expression:

```
zero = {fun {f} {fun {x} x}}
```
Implementing Numbers with LC

Can you write a program that produces one?

\[
\text{one} \overset{\text{def}}{=} \{ \text{fun} \ {f} \ {\text{fun} \ {x} \ {f \ x}} \}\}
\]

... which is also the function that takes \(f\) and \(x\) and applies \(f\) to \(x\) one time
Implementing Numbers with LC

Can you write a program that produces two?

\[
\text{two} \overset{\text{def}}{=} \{ \text{fun} \ \{ f \} \ \{ \text{fun} \ \{ x \} \ \{ f \ \{ f \ \{ x \} \} \} \} \} \\
\]

... which is also the function that takes \( f \) and \( x \) and applies \( f \) to \( x \) two times
Implementing Booleans with LC

Can you write a program that produces true?

\[
\text{true} = \{ \text{fun } \{x\} \{ \text{fun } \{y\} x \} \}
\]

... which is also the function that takes two arguments and returns the first one
Implementing Booleans with LC

Can you write a program that produces false?

\[
\text{false} \overset{\text{def}}{=} \{\text{fun} \ x \ \{\text{fun} \ y \ y\}\}\}
\]

... which is also the function that takes two arguments and returns the second one
Implementing Branches with LC

\[
\begin{align*}
\text{true} & \triangleq \{\text{fun } \{x\} \ (\text{fun } \{y\} \ x)\} \\
\text{false} & \triangleq \{\text{fun } \{x\} \ (\text{fun } \{y\} \ y)\} \\
\text{zero} & \triangleq \{\text{fun } \{f\} \ (\text{fun } \{x\} \ x)\} \\
\text{one} & \triangleq \{\text{fun } \{f\} \ (\text{fun } \{x\} \ (f \ x))\} \\
\text{two} & \triangleq \{\text{fun } \{f\} \ (\text{fun } \{x\} \ (f \ (f \ x)))\}
\end{align*}
\]

Can you write a program that produces zero when given true, one when given false?

\[
\{\text{fun } \{b\} \ (\{b \ \text{zero}\} \ \text{one})\}
\]

... because \text{true} returns its first argument and \text{false} returns its second argument

\[
\begin{align*}
\{\{\text{fun } \{b\} \ (\{b \ \text{zero}\} \ \text{one})\} \ \text{true}\} & \Rightarrow \\{\{\text{true \ zero}\} \ \text{one}\} \\
& \Rightarrow \ \text{zero} \\
\{\{\text{fun } \{b\} \ (\{b \ \text{zero}\} \ \text{one})\} \ \text{false}\} & \Rightarrow \\{\{\text{false \ zero}\} \ \text{one}\} \\
& \Rightarrow \ \text{one}
\end{align*}
\]
Implementing Pairs

Can you write a program that takes two arguments and produces a pair?

\[
\text{cons} \overset{\text{def}}{=} \{ \text{fun} \ x \} \ \{ \text{fun} \ y \} \\
\quad \quad \{ \text{fun} \ b \} \ \{ \{ b \ x \} \ y \} \}\}
\]

Examples:

\[
\{ \{ \text{cons} \ \text{zero} \} \ \text{one} \} \Rightarrow \{ \text{fun} \ b \} \ \{ \{ b \ \text{zero} \} \ \text{one} \} \}\}
\]

\[
\{ \{ \text{cons} \ \text{two} \} \ \text{zero} \} \Rightarrow \{ \text{fun} \ b \} \ \{ \{ b \ \text{two} \} \ \text{zero} \} \}\}
\]
Implementing Pairs

\[
\text{cons} \triangleq \{\text{fun} \ {x} \ \{\text{fun} \ {y} \\
\quad \quad \quad \quad \ \{\text{fun} \ {b} \ \{\{b \ x \} \ y}\}\}\}\}\n\]

Can you write a program that takes a pair and returns the first part?

Can you write a program that takes a pair and returns the rest?

\[
\text{first} \ \triangleq \ \{\text{fun} \ {p} \ \{p \ \text{true}\}\}\n\]
\[
\text{rest} \ \triangleq \ \{\text{fun} \ {p} \ \{p \ \text{false}\}\}\n\]

Example:

\[
\{\text{first} \ \{\{\text{cons} \ \text{zero}\} \ \text{one}\}\} \Rightarrow \{\text{first} \ \{\text{fun} \ {b} \ \{\{b \ \text{zero}\} \ \text{one}\}\}\}\n\]
\[
\Rightarrow \{\{\text{fun} \ {b} \ \{\{b \ \text{zero}\} \ \text{one}\}\} \ \text{true}\}\n\]
\[
\Rightarrow \{\{\text{true} \ \text{zero}\} \ \text{one}\}\n\]
\[
\Rightarrow \text{zero}
\]
Implementing Arithmetic

\[
\begin{align*}
\text{zero} & \quad \overset{\text{def}}{=} \quad \{ \text{fun } f \} \{ \text{fun } x \} \ x \\
\text{one} & \quad \overset{\text{def}}{=} \quad \{ \text{fun } f \} \{ \text{fun } x \} \{ f \ x \} \\
\text{two} & \quad \overset{\text{def}}{=} \quad \{ \text{fun } f \} \{ \text{fun } x \} \{ f \ \{ f \ x \} \} \\
\end{align*}
\]

Can you write a program that takes a number and adds one?

\[
\begin{align*}
\text{add1} & \quad \overset{\text{def}}{=} \quad \{ \text{fun } n \} \\
& \quad \{ \text{fun } g \} \{ \text{fun } y \} \\
& \quad \{ g \ \{ \{ n \ g \} \ y \} \} \} \} \\
\end{align*}
\]

Example:

\[
\begin{align*}
\{ \text{add1 } \text{zero} \} \Rightarrow & \quad \{ \text{fun } g \} \{ \text{fun } y \} \\
& \quad \{ g \ \{ \{ \text{zero } g \} \ y \} \} \} \} \\
= & \quad \{ \text{fun } g \} \{ \text{fun } y \} \\
& \quad \{ g \ \{ \{ \{ \text{fun } f \} \{ \text{fun } x \} \ x \} \ g \} \ y \} \} \} \\
\cong & \quad \{ \text{fun } g \} \{ \text{fun } y \} \\
& \quad \{ g \ y \} \} \\
= & \quad \text{one}
\end{align*}
\]
Implementing Arithmetic

Can you write a program that takes a number and adds two?

\[
\text{add2} = \text{fun } \{n\} \text{ add1 } \text{add1 n}\]

Implementing Arithmetic

Can you write a program that takes a number and adds three?

\[
\text{add3} \quad \text{def} \quad \{ \text{fun } \{n\} \quad \{\text{add1 } \{\text{add1 } \{\text{add1 } n\}\}\}\}
\]
Implementing Arithmetic

\[
\begin{align*}
\text{zero} & \equiv \{\text{fun } f\} \{\text{fun } x\} x \\
\text{one} & \equiv \{\text{fun } f\} \{\text{fun } x\} f x \\
\text{two} & \equiv \{\text{fun } f\} \{\text{fun } x\} f (f x)
\end{align*}
\]

Can you write a program that takes two numbers and adds them?

\[
\text{add} \equiv \{\text{fun } n\} \{\text{fun } m\} \{\{n \text{ add1}\} m\}
\]

... because a number \( n \) applies some function \( n \) times to an argument
Implementing Arithmetic

\[
\begin{align*}
\text{zero} & \overset{\text{def}}{=} \{ \text{fun} \ \{f\} \ \{ \text{fun} \ \{x\} \ \{x\} \} \} \\
\text{one} & \overset{\text{def}}{=} \{ \text{fun} \ \{f\} \ \{ \text{fun} \ \{x\} \ \{f \ \{x\}\} \} \} \\
\text{two} & \overset{\text{def}}{=} \{ \text{fun} \ \{f\} \ \{ \text{fun} \ \{x\} \ \{f \ \{f \ \{x\}\}\} \} \}
\end{align*}
\]

Can you write a program that takes two numbers and multiplies them?

\[
\text{mult} \overset{\text{def}}{=} \{ \text{fun} \ \{n\} \ \{ \text{fun} \ \{m\} \ \{ \{n \ \{\text{add} \ \{m\}\} \ \{\text{zero}\}\} \} \}
\]

... because adding number \( m \) to zero \( n \) times produces \( n \times m \)
Implementing Arithmetic

Can you write a program that tests for zero?

```plaintext
iszero = \{ fun \{ n \} { \{ n \{ fun \{ x \} false \} true \} } \}
```  

because applying `\{ fun \{ x \} false \}` zero times to `true` produces `true`, and applying it any other number of times produces `false`
Implementing Arithmetic

Can you write a program that takes a number and produces one less?

```
shift  def = {fun {p}
       {{cons {rest p}}} {add1 {rest p}}} }

sub1   def = {fun {n}
       {first
       {{n shift} {{cons zero} zero}}} }
```

And then subtraction is obvious...
Implementing Factorial

\[ \text{mk-rec} \overset{\text{def}}{=} \{ \text{fun} \ \{ \text{body} \} \}
\begin{align*}
&\{ \{ \text{fun} \ \{ \text{fx} \} \ \{ \text{fx} \ \text{fx} \} \} \\
&\{ \text{fun} \ \{ \text{fx} \} \\
&\quad \{ \{ \text{fun} \ \{ f \} \ \{ \text{body} \ f \} \} \\
&\quad \{ \text{fun} \ \{ x \} \ \{ \{ \text{fx} \ \text{fx} \} \ x \} \} \} \} \}
\end{align*} \]

Can you write a program that computes factorial?

\[ \{ \text{mk-rec} \}
\begin{align*}
&\{ \text{fun} \ \{ \text{fac} \} \}
&\quad \{ \text{fun} \ \{ n \} \\
&\qquad \{ \{ \{ \text{iszero} \ n \} \}
&\qquad \quad \text{one} \}
&\qquad \quad \{ \{ \text{mult} \ n \} \ \{ \text{fac} \ \{ \text{sub1} \ n \} \} \} \} \}
\end{align*} \]

... and when you can write factorial, you can probably write anything.