Function Definitions

Functions are not only about organizing code or making it easier to reuse code, but provide a powerful means of **abstraction**!

Iffy Lang

$$b ::= \mathsf{T} \mid \mathsf{F} \mid \mathsf{if}\ b_1\ \mathsf{then}\ b_2\ \mathsf{else}\ b_3$$

Iffy Lang

$$p ::= \text{func}\ name(x_1, \ldots, x_i)\{body\}$$

$$body ::= \text{return}\ b$$

$$b ::= x \mid name(b_1, \ldots, b_i) \mid \text{T} \mid \text{F} \mid \text{if}\ b_1\ \text{then}\ b_2\ \text{else}\ b_3$$

Example Program: Negation

```
func not(x) {
  ?
}
```

Example Program: Negation

```
func not(x) {
  return (if x
            then F
            else T)
}
```

Example Program: And

```
func and(x1, x2) {
  ?
}
```

Example Program: And

```
func and(x1, x2) {
 return (if x1
           then x2
           else F)
}
```

Example Program: And

```
func or(x1, x2) {
  ?
}
```

Example Program: And

```
func or(x1, x2) {
 return (if x1
          then T
          else x2)
}
```

Example Program: Implication

```
func implies(x1, x2) {
  ?
}
```

Example Program: Implication

```
func implies(x1, x2) {
  return or(not(x1),x2)
}
```

Example Program: Necessary and Sufficient

```
func iff(x1, x2) {
  ?
}
```

Example Program: Necessary and Sufficient

```
func iff(x1, x2) {
 return and(implies(x1,x2),implies(x2,x1))
}
```

Example Program: Exclusive-Or

```
func xor(x1, x2) {
  ?
}
```

Example Program: Exclusive-Or

```
func xor(x1, x2) {
  return or(and(x1,not(x2)),and(not(x1),x2))
}
```

Syntax Trees

Syntax trees are a tree representation of a syntactical expression.

Example Syntax Tree

if F then T else (if F then T else T)

$$\begin{tikzcd}
 & \text{if} \arrow[dl] \arrow[d] \arrow[dr] & & \\
T & F & \text{if} \arrow[dl] \arrow[d] \arrow[dr] & \\
 & T & F & T
\end{tikzcd}$$

Example Syntax Tree

if (if F then T else T) then F else (if F then T else T)

Example Syntax Tree

if (if F then T else T) then F else (if F then T else T)

$$\begin{tikzcd}
 & & \text{if} \arrow[dll] \arrow[dd] \arrow[drrr] & & & & \\
F & & & & & \text{if} \arrow[dl] \arrow[d] \arrow[dr] & \\
 & & \text{if} \arrow[dl] \arrow[d] \arrow[dr] & & T & F & T \\
 & T & F & T & & &
\end{tikzcd}$$

Example Syntax Tree

if x then T else y

$$\begin{tikzcd}
& \text{if} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{T} & x & y
\end{tikzcd}$$

Syntax Trees for Function Definitions

```
func not(x) {
  return (if x
             then F
             else T)
}
```

$$\begin{tikzcd}
 & \mathit{not}(x_1, x_2) \arrow[d] & \\
 & \mathit{return} \arrow[d] & \\
 & \mathit{if} \arrow[dl] \arrow[d] \arrow[dr] & \\
\mathrm{F} & x_1 & \mathrm{T}
\end{tikzcd}$$

Syntax Trees for Function Definitions

```
func and(x1, x2) {
  return (if x1
          then x2
          else F)
}
```

Syntax Trees for Function Definitions

```
func and(x1, x2) {
 return (if x1
          then x2
          else F)
}
```

$$\begin{tikzcd}
 & and(x_1, x_2) \arrow[d] & \\
 & return \arrow[d] & \\
 & if \arrow[dl] \arrow[d] \arrow[dr] & \\
\mathrm{x}_2 & x_1 & \mathrm{F}
\end{tikzcd}$$

Syntax Trees for Function Application

$name(b_1, \ldots, b_i)$

$$\begin{tikzcd}
& name \arrow[dl] \arrow[d] \arrow[dr] & \\
b_1 & \cdots & b_i
\end{tikzcd}$$

Syntax Trees for Function Application

```
func implies(x1,
x2) {
  return
  or(not(x1),x2)
}
```

$$\begin{tikzcd}
& implies(x_1, x_2) \arrow[d] & \\
& return \arrow[d] & \\
& or \arrow[dl] \arrow[dr] & \\
\text{not} \arrow[d] & & x_2 \\
x_1 & &
\end{tikzcd}$$