Outline

- Programming with Functions
- Defining a Language
- Defining Type Rules
- Type Soundness

Programming with Functions

- A program comprises function definitions and applications

$$\mathbf{f}(\mathbf{x}) \equiv (\mathbf{x} \times \mathbf{x}) + 10$$

$$\mathbf{f}(2) = 14$$

Programming with Functions

- A program comprises function definitions and applications

$$\mathbf{f}(\mathbf{x}) \equiv (\mathbf{x} \times \mathbf{x}) + 10$$

$$\mathbf{g}(\mathbf{y}) \equiv 3 \times \mathbf{y}$$

$$\mathbf{g}(\mathbf{f}(2)) = 42$$

Programming with Functions

- Functions consume and produce more than numbers

$$\textbf{mkpair}(\mathbf{x}, \mathbf{y}) \equiv \langle \mathbf{x}, \mathbf{y} \rangle$$

$$\textbf{mkpair}(1, 2) = \langle 1, 2 \rangle$$

Programming with Functions

- Functions consume and produce more than numbers

$$\mathbf{mkpair}(\mathbf{x}, \mathbf{y}) \equiv \langle \mathbf{x}, \mathbf{y} \rangle$$

$$\mathbf{mklist}(\mathbf{x}, \mathbf{y}) \equiv \mathbf{mkpair}(\mathbf{x}, \mathbf{mkpair}(\mathbf{y}, \text{empty}))$$

$$\mathbf{mklist}(1, 2) = \langle 1, \langle 2, \text{empty} \rangle \rangle$$

Programming with Functions

- Functions consume and produce more than numbers

$$\mathbf{mkpair}(\mathbf{x}, \mathbf{y}) \equiv \langle \mathbf{x}, \mathbf{y} \rangle$$

$$\mathbf{mklist}(\mathbf{x}, \mathbf{y}) \equiv \mathbf{mkpair}(\mathbf{x}, \mathbf{mkpair}(\mathbf{y}, \mathrm{empty}))$$

$$\mathbf{fst}(\langle \mathbf{x}, \mathbf{y} \rangle) \equiv \mathbf{x}$$

$$\mathbf{fst}(\mathbf{mklist}(1, 2)) = 1$$

Programming with Functions

- Use functions to build complex data from simple constructs
- Implement branches with conditional functions

$$\textbf{add}(\textbf{n}, \textbf{N}, \textbf{pb}) \equiv \langle\langle \textbf{n}, \textbf{N} \rangle, \textbf{pb} \rangle$$

$$\textbf{lookup}(\textbf{n}, \langle\langle \textbf{n2}, \textbf{N} \rangle, \textbf{pb} \rangle) \equiv \begin{cases} \textbf{n} = \textbf{n2} \;\; \textbf{N} \\ \textbf{n} \neq \textbf{n2} \;\; \textbf{lookup}(\textbf{n}, \textbf{pb}) \end{cases}$$

lookup("Jack", **add**("Jack", "x1212", empty)) = "x1212"

Computation as Algebra

- Compute using algebraic equivalences

$$\mathbf{f}(\mathbf{x}) \equiv (\mathbf{x} \times \mathbf{x}) + 10$$

$$\mathbf{f}(2) \quad =$$

Computation as Algebra

- Compute using algebraic equivalences

$$\mathbf{f}(\mathbf{x}) \equiv (\mathbf{x} \times \mathbf{x}) + 10$$

$$\begin{aligned} \mathbf{f}(2) &= (2 \times 2) + 10 \\ &= 4 + 10 \\ &= 14 \end{aligned}$$

Computation as Algebra

- Equivalence is pattern matching...

$$\mathbf{mkpair}(\mathbf{x}, \mathbf{y}) \equiv \langle \mathbf{x}, \mathbf{y} \rangle$$

$$\mathbf{mklist}(\mathbf{x}, \mathbf{y}) \equiv \mathbf{mkpair}(\mathbf{x}, \mathbf{mkpair}(\mathbf{y}, \text{empty}))$$

$$\mathbf{mklist}(1, 2) \quad =$$

Computation as Algebra

- Equivalence is pattern matching...

$$\textbf{mkpair}(\mathbf{x}, \mathbf{y}) \equiv \langle \mathbf{x}, \mathbf{y} \rangle$$

$$\textbf{mklist}(\mathbf{x}, \mathbf{y}) \equiv \textbf{mkpair}(\mathbf{x}, \textbf{mkpair}(\mathbf{y}, \text{empty}))$$

$$\begin{aligned} \textbf{mklist}(1, 2) &= \textbf{mkpair}(1, \textbf{mkpair}(2, \text{empty})) \\ &= \langle 1, \textbf{mkpair}(2, \text{empty}) \rangle \\ &= \langle 1, \langle 2, \text{empty} \rangle \rangle \end{aligned}$$

$$\begin{aligned} \textit{or} \quad &= \textbf{mkpair}(1, \textbf{mkpair}(2, \text{empty})) \\ &= \textbf{mkpair}(1, \langle 2, \text{empty} \rangle) \\ &= \langle 1, \langle 2, \text{empty} \rangle \rangle \end{aligned}$$

Computation as Algebra

- ... and matching with conditionals

$$\textbf{add}(\textbf{n}, \textbf{N}, \textbf{pb}) \equiv \langle\langle \textbf{n}, \textbf{N}\rangle, \textbf{pb}\rangle$$

$$\textbf{lookup}(\textbf{n}, \langle\langle \textbf{n2}, \textbf{N}\rangle, \textbf{pb}\rangle) \equiv \begin{cases} \textbf{n} = \textbf{n2} \quad \textbf{N} \\ \textbf{n} \neq \textbf{n2} \quad \textbf{lookup}(\textbf{n}, \textbf{pb}) \end{cases}$$

$$\textbf{lookup}(\text{"Jack"}, \textbf{add}(\text{"Jack"}, \text{"x1212"}, \text{empty}))$$

$$= \quad \textbf{lookup}(\text{"Jack"}, \langle\langle \text{"Jack"}, \text{"x1212"}\rangle, \text{empty}\rangle)$$

$$= \quad \text{"x1212"}$$

Computation as Algebra

- ... and matching with conditionals

$$\textbf{add}(\textbf{n}, \textbf{N}, \textbf{pb}) \equiv \langle\langle \textbf{n}, \textbf{N}\rangle, \textbf{pb}\rangle$$

$$\textbf{lookup}(\textbf{n}, \langle\langle \textbf{n2}, \textbf{N}\rangle, \textbf{pb}\rangle) \equiv \begin{cases} \textbf{n} = \textbf{n2} & \textbf{N} \\ \textbf{n} \neq \textbf{n2} & \textbf{lookup}(\textbf{n}, \textbf{pb}) \end{cases}$$

lookup("Jill", **add**("Jack", "x1212", empty))

= **lookup**("Jill", ⟨⟨"Jack", "x1212"⟩, empty⟩)

= **lookup**("Jill", empty)

stuck implies an error

Higher-Order Functions

- A ***higher-order function*** is one that consumes or produces functions

$$\mathbf{f}(\mathbf{x}) \equiv \mathbf{x} \times \mathbf{x}$$

$$\mathbf{twice}(\mathbf{g}, \mathbf{x}) \equiv \mathbf{g}(\mathbf{g}(\mathbf{x}))$$

$$\begin{aligned} \mathbf{twice}(\mathbf{f}, 2) &= \mathbf{f}(\mathbf{f}(2)) \\ &= \mathbf{f}(2 \times 2) \\ &= \mathbf{f}(4) \\ &= 4 \times 4 \\ &= 16 \end{aligned}$$

Higher-Order Functions

- A ***higher-order function*** is one that consumes or produces functions

$$\mathbf{fst}(\langle \mathbf{x}, \mathbf{y} \rangle) \equiv \mathbf{x}$$

$$\mathbf{twice}(\mathbf{g}, \mathbf{x}) \equiv \mathbf{g}(\mathbf{g}(\mathbf{x}))$$

$$\begin{aligned} \mathbf{twice}(\mathbf{fst}, \langle\langle 1, 2\rangle, 3\rangle) &= \mathbf{fst}(\mathbf{fst}(\langle\langle 1, 2\rangle, 3\rangle)) \\ &= \mathbf{fst}(\langle 1, 2\rangle) \\ &= 1 \end{aligned}$$

The Direction of Evaluation

$$3 + 4 = ?$$

The Direction of Evaluation

$$3 + 4 = 3 + (2 + 2)$$

The Direction of Evaluation

$$\begin{aligned} \mathbf{f}(2) &= -1 + \mathbf{f}(2) + 1 \\ &= -1 + \mathbf{f}(\mathbf{sqrt}(4)) + 1 \\ &= \mathbf{\ldots} \end{aligned}$$

- For programming, we want an evaluation direction that produces ***values***

Expressions and Values

- Many possible ***expressions***

$$8$$

$$2 + 7 + \textbf{sqrt}(9)$$

$$\textbf{fst}$$

$$\langle 1, \textbf{fst}(\langle \text{empty}, \text{empty} \rangle) \rangle$$

- Certain expressions are designated as ***values***

$$8$$

$$\textbf{fst}$$

$$\langle 1, \text{empty} \rangle$$

Evaluation

- Define evaluation to ***reduce*** expressions to values

$$(2+7)+8 \;\rightarrow\; 9+8$$
$$\rightarrow\; 17$$

Evaluation with Higher-Order Functions

- Problem: creating new function values

$$\mathbf{f}(\mathbf{x}) \equiv \mathbf{x} + 1$$

$$\mathbf{g}(\mathbf{y}) \equiv \mathbf{y} + 2$$

$$\mathbf{compose}(\mathbf{a}, \mathbf{b}) \equiv \mathbf{...}$$

can't put **a**(**b**(**...**)) in place of **...**

Evaluation with Higher-Order Functions

- Problem: creating new function values

$$\mathbf{f}(\mathbf{x}) \equiv \mathbf{x} + 1$$

$$\mathbf{g}(\mathbf{y}) \equiv \mathbf{y} + 2$$

$$\mathbf{compose}(\mathbf{a}, \mathbf{b}) \equiv \mathbf{...}$$

$$\mathbf{compose}(\mathbf{f}, \mathbf{g}) \quad \rightarrow \quad \mathbf{...}$$

$$\rightarrow \quad \mathbf{h}$$

where

$$\mathbf{h}(\mathbf{z}) = \mathbf{f}(\mathbf{g}(\mathbf{z}))$$

Evaluation with Higher-Order Functions

- Redunction-friendly function notation:

Replace

$$\mathbf{f}(\mathbf{x}) \equiv \mathbf{x} + 1$$

with

$$\mathbf{f} \equiv (\lambda\, \mathbf{x}\, .\, \mathbf{x} + 1)$$

Evaluation with Higher-Order Functions

- Definition with $\equiv$ merely creates a shorthand

$$\mathbf{f} \equiv (\lambda\, \mathbf{x}\, .\, \mathbf{x} + 1)$$

- Apply functions through λ-application reduction

$$(\lambda\, \mathbf{x}\, .\, \mathbf{M})(\mathbf{v}) \rightarrow \mathbf{M} \text{ with } \mathbf{x} \text{ replaced by } \mathbf{v}$$

Evaluation with Higher-Order Functions

- Definition with $\equiv$ merely creates a shorthand

$$\mathbf{f} \equiv (\lambda\, \mathbf{x} \,.\, \mathbf{x} + 1)$$

- Apply functions through λ-application reduction

$$(\lambda\, \mathbf{x} \,.\, \mathbf{M})(\mathbf{v}) \rightarrow \mathbf{M}[\mathbf{v}/\mathbf{x}]$$

$$\begin{aligned} \mathbf{f}(10) &= (\lambda\, \mathbf{x} \,.\, \mathbf{x} + 1)(10) \\ &\rightarrow 10 + 1 \\ &\rightarrow 11 \end{aligned}$$

Evaluation with Higher-Order Functions

- Simple functions as values

$$\textbf{mkadder} \equiv (\lambda\, \textbf{m} \,.\, (\lambda\, \textbf{n} \,.\, \textbf{m} + \textbf{n}))$$

$$\textbf{add1} \equiv \textbf{mkadder}(1)$$

$$\textbf{add5} \equiv \textbf{mkadder}(5)$$

$$\begin{aligned} \textbf{add5} \quad &= \quad (\lambda\, \textbf{m} \,.\, (\lambda\, \textbf{n} \,.\, \textbf{m} + \textbf{n}))(5) \\ &\rightarrow \quad (\lambda\, \textbf{n} \,.\, 5 + \textbf{n}) \end{aligned}$$

Evaluation with Higher-Order Functions

- Simple functions as values

$$\mathbf{mkadder} \equiv (\lambda\, \mathbf{m}\, .\, (\lambda\, \mathbf{n}\, .\, \mathbf{m} + \mathbf{n}))$$

$$\mathbf{add1} \equiv \mathbf{mkadder}(1)$$

$$\mathbf{add5} \equiv \mathbf{mkadder}(5)$$

$$\begin{aligned} \mathbf{add5}(1) \quad &= \quad (\lambda\, \mathbf{m}\, .\, (\lambda\, \mathbf{n}\, .\, \mathbf{m} + \mathbf{n}))(5)(1) \\ &\rightarrow \quad (\lambda\, \mathbf{n}\, .\, 5 + \mathbf{n})(1) \\ &\rightarrow \quad 5 + 1 \\ &\rightarrow \quad 6 \end{aligned}$$

Evaluation with Higher-Order Functions

- Returning to the definition of **compose**

$$\mathbf{f} \equiv (\lambda\, \mathbf{x} \,.\, \mathbf{x} + 1)$$

$$\mathbf{g} \equiv (\lambda\, \mathbf{y} \,.\, \mathbf{y} + 2)$$

$$\mathbf{compose} \equiv (\lambda\, (\mathbf{a}, \mathbf{b}) \,.\, (\lambda\, \mathbf{z} \,.\, \mathbf{a}(\mathbf{b}(\mathbf{z}))))$$

$$\begin{aligned} \mathbf{compose}(\mathbf{f}, \mathbf{g}) \quad &= \quad (\lambda\, (\mathbf{a}, \mathbf{b}) \,.\, (\lambda\, \mathbf{z} \,.\, \mathbf{a}(\mathbf{b}(\mathbf{z}))))(\mathbf{f},\mathbf{g}) \\ &\rightarrow \quad (\lambda\, \mathbf{z} \,.\, \mathbf{f}(\mathbf{g}(\mathbf{z}))) \end{aligned}$$

Abbreviations

$$\mathbf{fac} \equiv \lambda \mathtt{n}.\,\mathtt{if0}\ \mathtt{n}\ \mathtt{then}\ \lceil 1 \rceil\ \mathtt{else}\ \mathtt{n} \times \mathbf{fac}(\mathtt{n} - \lceil 1 \rceil)$$

Illegal: **fac** isn't merely a shorthand because it mentions itself

$$\mathbf{mkfac} \equiv \lambda \mathtt{f}.\,\lambda \mathtt{n}.\,\mathtt{if0}\ \mathtt{n}\ \mathtt{then}\ \lceil 1 \rceil\ \mathtt{else}\ \mathtt{n} \times (\mathbf{f}(\mathbf{f}))(\mathtt{n} - \lceil 1 \rceil)$$

$$\mathbf{fac} \equiv \mathbf{mkfac}(\mathbf{mkfac})$$

Outline

- Programming with Functions
- Defining a Language
- Defining Type Rules
- Type Soundness

Definining a Functional Language

Steps to defining a language:

- Define the syntax for expressions
- Designate certain expressions as values
- Define the reduction rules on expressions

Syntax: Expressions

$$
\begin{array}{lcl}
\mathbf{M} & = & \lceil \mathbf{n} \rceil \\
 & | & \mathbf{x} \\
 & | & \mathbf{M} - \mathbf{M} \\
 & | & \mathbf{M} \times \mathbf{M} \\
 & | & \texttt{if0}\ \mathbf{M}\ \texttt{then}\ \mathbf{M}\ \texttt{else}\ \mathbf{M} \\
 & | & \lambda \mathbf{x} . \mathbf{M} \\
 & | & \mathbf{M}\ \mathbf{M} \\
\mathbf{n} & = & \text{an integer} \\
\mathbf{x} & = & \text{a variable}
\end{array}
$$

where parentheses can be put around any **M**

$\lceil 5 \rceil$ represents 5

Syntax: Expressions

$$
\begin{array}{lcl}
\mathbf{M} & = & \lceil \mathbf{n} \rceil \\
 & | & \mathbf{x} \\
 & | & \mathbf{M} - \mathbf{M} \\
 & | & \mathbf{M} \times \mathbf{M} \\
 & | & \texttt{if0}\ \mathbf{M}\ \texttt{then}\ \mathbf{M}\ \texttt{else}\ \mathbf{M} \\
 & | & \lambda\, \mathbf{x}\, .\, \mathbf{M} \\
 & | & \mathbf{M}\ \mathbf{M} \\
\mathbf{n} & = & \text{an integer} \\
\mathbf{x} & = & \text{a variable}
\end{array}
$$

where parentheses can be put around any **M**

$\lceil 5 \rceil - \lceil 3 \rceil$ represents the subtraction of 3 from 5

Syntax: Expressions

$$
\begin{aligned}
\mathbf{M} \quad &= \quad \lceil \mathbf{n} \rceil \\
&\mid \quad \mathbf{x} \\
&\mid \quad \mathbf{M} - \mathbf{M} \\
&\mid \quad \mathbf{M} \times \mathbf{M} \\
&\mid \quad \texttt{if0}\ \mathbf{M}\ \texttt{then}\ \mathbf{M}\ \texttt{else}\ \mathbf{M} \\
&\mid \quad \lambda\, \mathbf{x}\, .\, \mathbf{M} \\
&\mid \quad \mathbf{M}\ \mathbf{M} \\
\mathbf{n} \quad &= \quad \text{an integer} \\
\mathbf{x} \quad &= \quad \text{a variable}
\end{aligned}
$$

where parentheses can be put around any **M**

$\lambda\, \mathbf{x}\, .\, \mathbf{x}$ represents the identity function

Syntax: Expressions

$$
\begin{aligned}
\mathbf{M} \;&=\; \lceil \mathbf{n} \rceil \\
&\mid\; \mathbf{x} \\
&\mid\; \mathbf{M} - \mathbf{M} \\
&\mid\; \mathbf{M} \times \mathbf{M} \\
&\mid\; \texttt{if0}\ \mathbf{M}\ \texttt{then}\ \mathbf{M}\ \texttt{else}\ \mathbf{M} \\
&\mid\; \lambda\, \mathbf{x}\, .\, \mathbf{M} \\
&\mid\; \mathbf{M}\ \mathbf{M} \\
\mathbf{n} \;&=\; \text{an integer} \\
\mathbf{x} \;&=\; \text{a variable}
\end{aligned}
$$

where parentheses can be put around any **M**

$(\lambda\, \mathbf{x}\, .\, \mathbf{x})(\lceil 5 \rceil)$ represents applying the identity function to 5

Syntax: Values

$$\mathbf{V} \quad = \quad \lceil \mathbf{n} \rceil$$
$$\quad | \quad \lambda\, \mathbf{x} \,.\, \mathbf{M}$$

$\lceil 5 \rceil$ — a value

$\lambda\, \mathbf{x} \,.\, \mathbf{x}$ — a value

$\lceil 5 \rceil - \lceil 3 \rceil$ — **not** a value

$(\lambda\, \mathbf{x} \,.\, \mathbf{x})(\lceil 5 \rceil)$ — **not** a value

$\lambda\, \mathbf{y} \,.\, ((\lambda\, \mathbf{x}.\, \mathbf{x})(\mathbf{y}))$ — a value

Reductions

$\lceil \mathbf{n}_1 \rceil - \lceil \mathbf{n}_2 \rceil$	$\rightarrow$	$\lceil \mathbf{n}_1 - \mathbf{n}_2 \rceil$
$\lceil \mathbf{n}_1 \rceil \times \lceil \mathbf{n}_2 \rceil$	$\rightarrow$	$\lceil \mathbf{n}_1 \times \mathbf{n}_2 \rceil$
$\texttt{if0}\, \lceil 0 \rceil\, \texttt{then}\, \mathbf{M}_1\, \texttt{else}\, \mathbf{M}_2$	$\rightarrow$	$\mathbf{M}_1$
$\texttt{if0}\, \lceil \mathbf{n} \rceil\, \texttt{then}\, \mathbf{M}_1\, \texttt{else}\, \mathbf{M}_2$	$\rightarrow$	$\mathbf{M}_2$ if $\mathbf{n} \neq 0$
$(\lambda\, \mathbf{x}\, .\, \mathbf{M})(\mathbf{V})$	$\rightarrow$	$\mathbf{M}[\mathbf{V}/\mathbf{x}]$

$$\lceil 5 \rceil - \lceil 3 \rceil \rightarrow \lceil 2 \rceil$$

Reductions

$$\lceil \mathbf{n}_1 \rceil - \lceil \mathbf{n}_2 \rceil \quad \rightarrow \quad \lceil \mathbf{n}_1 - \mathbf{n}_2 \rceil$$

$$\lceil \mathbf{n}_1 \rceil \times \lceil \mathbf{n}_2 \rceil \quad \rightarrow \quad \lceil \mathbf{n}_1 \times \mathbf{n}_2 \rceil$$

$$\texttt{if0}\, \lceil 0 \rceil\, \texttt{then}\, \mathbf{M}_1\, \texttt{else}\, \mathbf{M}_2 \quad \rightarrow \quad \mathbf{M}_1$$

$$\texttt{if0}\, \lceil \mathbf{n} \rceil\, \texttt{then}\, \mathbf{M}_1\, \texttt{else}\, \mathbf{M}_2 \quad \rightarrow \quad \mathbf{M}_2 \quad \text{if } \mathbf{n} \neq 0$$

$$(\lambda\, \mathbf{x} \,.\, \mathbf{M})(\mathbf{V}) \quad \rightarrow \quad \mathbf{M}[\mathbf{V}/\mathbf{x}]$$

$$\texttt{if0}\, \lceil 0 \rceil\, \texttt{then}\, \lceil 5 \rceil\, \texttt{else}\, (\lambda\, \mathbf{x} \,.\, \mathbf{x}) \rightarrow \lceil 5 \rceil$$

Reductions

$$\lceil \mathbf{n}_1 \rceil - \lceil \mathbf{n}_2 \rceil \rightarrow \lceil \mathbf{n}_1 - \mathbf{n}_2 \rceil$$

$$\lceil \mathbf{n}_1 \rceil \times \lceil \mathbf{n}_2 \rceil \rightarrow \lceil \mathbf{n}_1 \times \mathbf{n}_2 \rceil$$

$$\texttt{if0}\,\lceil 0 \rceil\,\texttt{then}\,\mathbf{M}_1\,\texttt{else}\,\mathbf{M}_2 \rightarrow \mathbf{M}_1$$

$$\texttt{if0}\,\lceil \mathbf{n} \rceil\,\texttt{then}\,\mathbf{M}_1\,\texttt{else}\,\mathbf{M}_2 \rightarrow \mathbf{M}_2 \quad \text{if } \mathbf{n} \neq 0$$

$$(\lambda\,\mathbf{x}\,.\,\mathbf{M})(\mathbf{V}) \rightarrow \mathbf{M}[\mathbf{V}/\mathbf{x}]$$

$$\texttt{if0}\,\lceil 1 \rceil\,\texttt{then}\,\lceil 5 \rceil\,\texttt{else}\,(\lambda\,\mathbf{x}\,.\,\mathbf{x}) \rightarrow (\lambda\,\mathbf{x}\,.\,\mathbf{x})$$

Reductions

$$\lceil \mathbf{n}_1 \rceil - \lceil \mathbf{n}_2 \rceil \rightarrow \lceil \mathbf{n}_1 - \mathbf{n}_2 \rceil$$

$$\lceil \mathbf{n}_1 \rceil \times \lceil \mathbf{n}_2 \rceil \rightarrow \lceil \mathbf{n}_1 \times \mathbf{n}_2 \rceil$$

$$\mathtt{if0}\ \lceil 0 \rceil\ \mathbf{then}\ \mathbf{M}_1\ \mathtt{else}\ \mathbf{M}_2 \rightarrow \mathbf{M}_1$$

$$\mathtt{if0}\ \lceil \mathbf{n} \rceil\ \mathbf{then}\ \mathbf{M}_1\ \mathtt{else}\ \mathbf{M}_2 \rightarrow \mathbf{M}_2 \quad \text{if } \mathbf{n} \neq 0$$

$$(\lambda\, \mathbf{x} \,.\, \mathbf{M})(\mathbf{V}) \rightarrow \mathbf{M}[\mathbf{V}/\mathbf{x}]$$

$$(\lambda\, \mathbf{x} \,.\, \mathbf{x} \times \lceil 10 \rceil)(\lceil 8 \rceil) \rightarrow \lceil 8 \rceil \times \lceil 10 \rceil$$

Reductions in Context

$$\mathbf{M}_1 - \mathbf{M}_2 \quad \longrightarrow \quad \mathbf{M}_1' - \mathbf{M}_2$$

$$\text{where } \mathbf{M}_1 \longrightarrow \mathbf{M}_1'$$

$$\mathbf{V} - \mathbf{M}_2 \quad \longrightarrow \quad \mathbf{V} - \mathbf{M}_2'$$

$$\text{where } \mathbf{M}_2 \longrightarrow \mathbf{M}_2'$$

$$\mathbf{M}_1 \times \mathbf{M}_2 \quad \longrightarrow \quad \mathbf{M}_1' \times \mathbf{M}_2$$

...

$$(\lceil 5 \rceil \times \lceil 2 \rceil) - (\lceil 3 \rceil \times \lceil 4 \rceil) \longrightarrow \lceil 10 \rceil - (\lceil 3 \rceil \times \lceil 4 \rceil)$$

Reductions in Context

$$\mathbf{M}_1 - \mathbf{M}_2 \quad \to \quad \mathbf{M}_1' - \mathbf{M}_2$$

$$\text{where } \mathbf{M}_1 \to \mathbf{M}_1'$$

$$\mathbf{V} - \mathbf{M}_2 \quad \to \quad \mathbf{V} - \mathbf{M}_2'$$

$$\text{where } \mathbf{M}_2 \to \mathbf{M}_2'$$

$$\mathbf{M}_1 \times \mathbf{M}_2 \quad \to \quad \mathbf{M}_1' \times \mathbf{M}_2$$

...

$$\lceil 10 \rceil - (\lceil 3 \rceil \times \lceil 4 \rceil) \to \lceil 10 \rceil - \lceil 12 \rceil$$

Reductions in Context

$$\texttt{if0}\ \mathbf{M}\ \texttt{then}\ \mathbf{M}_1\ \texttt{else}\ \mathbf{M}_2 \ \longrightarrow\ \texttt{if0}\ \mathbf{M}'\ \texttt{then}\ \mathbf{M}_1\ \texttt{else}\ \mathbf{M}_2$$

where $\mathbf{M} \longrightarrow \mathbf{M}'$

$$\mathbf{M}_1\ \mathbf{M}_2 \ \longrightarrow\ \mathbf{M}_1'\ \mathbf{M}_2$$

where $\mathbf{M}_1 \longrightarrow \mathbf{M}_1'$

$$\mathbf{V}\ \mathbf{M}_2 \ \longrightarrow\ \mathbf{V}\ \mathbf{M}_2'$$

where $\mathbf{M}_2 \longrightarrow \mathbf{M}_2'$

$$(\lambda\,\mathbf{x}\,.\,\mathbf{x})(\lceil 2 \rceil \times \lceil 2 \rceil) \ \longrightarrow\ (\lambda\,\mathbf{x}\,.\,\mathbf{x})(\lceil 4 \rceil)$$

Reductions in Context

$$\texttt{if0}\ \mathbf{M}\ \texttt{then}\ \mathbf{M}_1\ \texttt{else}\ \mathbf{M}_2 \ \longrightarrow\ \texttt{if0}\ \mathbf{M'}\ \texttt{then}\ \mathbf{M}_1\ \texttt{else}\ \mathbf{M}_2$$

where $\mathbf{M} \longrightarrow \mathbf{M'}$

$$\mathbf{M}_1\ \mathbf{M}_2 \ \longrightarrow\ \mathbf{M}'_1\ \mathbf{M}_2$$

where $\mathbf{M}_1 \longrightarrow \mathbf{M}'_1$

$$\mathbf{V}\ \mathbf{M}_2 \ \longrightarrow\ \mathbf{V}\ \mathbf{M}'_2$$

where $\mathbf{M}_2 \longrightarrow \mathbf{M}'_2$

$$((\lambda\,\mathbf{x}\,.\,\mathbf{x})(\lambda\,\mathbf{y}\,.\,\mathbf{y}))(\lceil 2 \rceil \times \lceil 2 \rceil) \ \longrightarrow\ (\lambda\,\mathbf{y}\,.\,\mathbf{y})(\lceil 2 \rceil \times \lceil 2 \rceil)$$

Reductions in Context

A simpler way: define context

$$
\begin{array}{rcl}
\mathbf{E} & = & \texttt{[]} \\
 & | & \mathbf{E} - \mathbf{M} \\
 & | & \mathbf{V} - \mathbf{E} \\
 & | & \mathbf{E} \times \mathbf{M} \\
 & | & \mathbf{V} \times \mathbf{E} \\
 & | & (\mathbf{E}\ \mathbf{M}) \\
 & | & (\mathbf{V}\ \mathbf{E}) \\
 & | & \texttt{if0}\ \mathbf{E}\ \texttt{then}\ \mathbf{M}\ \texttt{else}\ \mathbf{M}
\end{array}
$$

$$\mathbf{E}\texttt{[}\mathbf{M}\texttt{]} \quad \to \quad \mathbf{E}\texttt{[}\mathbf{M}'\texttt{]} \quad \text{where } \mathbf{M} \to \mathbf{M}'$$

$\mathbf{E}\texttt{[}\mathbf{M}\texttt{]}$ means $\mathbf{E}$ with $\texttt{[]}$ replaced by $\mathbf{M}$

Reductions in Context

A simpler way: define context

$$
\begin{array}{rcl}
\mathbf{E} & = & \mathtt{[\,]} \\
 & | & \mathbf{E} - \mathbf{M} \\
 & | & \mathbf{V} - \mathbf{E} \\
 & | & \mathbf{E} \times \mathbf{M} \\
 & | & \mathbf{V} \times \mathbf{E} \\
 & | & (\mathbf{E}\ \mathbf{M}) \\
 & | & (\mathbf{V}\ \mathbf{E}) \\
 & | & \mathtt{if0}\ \mathbf{E}\ \mathtt{then}\ \mathbf{M}\ \mathtt{else}\ \mathbf{M}
\end{array}
$$

$$
\mathbf{E}\mathtt{[}\mathbf{M}\mathtt{]} \quad \rightarrow \quad \mathbf{E}\mathtt{[}\mathbf{M}'\mathtt{]} \quad \text{where } \mathbf{M} \rightarrow \mathbf{M}'
$$

$$
\begin{array}{lcl}
\mathbf{E} & = & \ulcorner 4 \urcorner - (\mathtt{[\,]} \times (\ulcorner 2 \urcorner + \ulcorner 1 \urcorner)) \\
\mathbf{E}\mathtt{[}(\ulcorner 4 \urcorner - \ulcorner 5 \urcorner)\mathtt{]} & = & \ulcorner 4 \urcorner - ((\ulcorner 4 \urcorner - \ulcorner 5 \urcorner) \times (\ulcorner 2 \urcorner + \ulcorner 1 \urcorner))
\end{array}
$$

Reductions

$$\lceil \mathbf{n}_1 \rceil - \lceil \mathbf{n}_2 \rceil \quad \rightarrow \quad \lceil \mathbf{n}_1 - \mathbf{n}_2 \rceil$$

$$\lceil \mathbf{n}_1 \rceil \times \lceil \mathbf{n}_2 \rceil \quad \rightarrow \quad \lceil \mathbf{n}_1 \times \mathbf{n}_2 \rceil$$

$$\mathtt{if0}\ \lceil 0 \rceil\ \mathtt{then}\ \mathbf{M}_1\ \mathtt{else}\ \mathbf{M}_2 \quad \rightarrow \quad \mathbf{M}_1$$

$$\mathtt{if0}\ \lceil \mathbf{n} \rceil\ \mathtt{then}\ \mathbf{M}_1\ \mathtt{else}\ \mathbf{M}_2 \quad \rightarrow \quad \mathbf{M}_2 \quad \text{if } \mathbf{n} \neq 0$$

$$(\lambda\, \mathbf{x}\, .\, \mathbf{M})(\mathbf{V}) \quad \rightarrow \quad \mathbf{M}[\mathbf{V}/\mathbf{x}]$$

$$\mathbf{E}[\mathbf{M}] \quad \rightarrow \quad \mathbf{E}[\mathbf{M}'] \quad \text{where } \mathbf{M} \rightarrow \mathbf{M}'$$

Is this language deterministic?

Deterministic Reduction

Theorem: For any **M**, at most one reduction rule applies.

Proof: By induction on the structure of **M**.

... requires a lemma ...

Lemma: There exists at most one **E** and $\mathbf{M}_0$ such that $\mathbf{E}[\mathbf{M}_0] = \mathbf{M}$ where $\mathbf{M}_0$ is reducible by one of the first five reduction rules.

Proof: By induction on the structure of **M**.

Induction on Expressions

$$
\begin{aligned}
\mathbf{M} \quad = \quad & \ulcorner \mathbf{n} \urcorner \\
\mid \quad & \mathbf{x} \\
\mid \quad & \mathbf{M} - \mathbf{M} \\
\mid \quad & \mathbf{M} \times \mathbf{M} \\
\mid \quad & \texttt{if0}\ \mathbf{M}\ \texttt{then}\ \mathbf{M}\ \texttt{else}\ \mathbf{M} \\
\mid \quad & \lambda\, \mathbf{x}\, .\, \mathbf{M} \\
\mid \quad & \mathbf{M}\ \mathbf{M}
\end{aligned}
$$

base case   inductive case

Base Cases

$$\begin{array}{lll} \mathbf{E} & = & [\,] \mid \mathbf{E} - \mathbf{M} \mid \mathbf{V} - \mathbf{E} \mid \mathbf{E} \times \mathbf{M} \mid \mathbf{V} \times \mathbf{E} \\ & \mid & (\mathbf{E}\ \mathbf{M}) \mid (\mathbf{V}\ \mathbf{E}) \mid \mathtt{if0}\ \mathbf{E}\ \mathtt{then}\ \mathbf{M}\ \mathtt{else}\ \mathbf{M} \end{array}$$

- Assume $\mathbf{M} = \lceil \mathbf{n} \rceil$
  - The only way to match the grammar for **E** is $\mathbf{E} = [\,]$ and $\mathbf{M}_0 = \lceil \mathbf{n} \rceil$. But that $\mathbf{M}_0$ is not reducible, so there are no matches.
- Assume $\mathbf{M} = \mathbf{x}$
  - The only way to match the grammar for **E** is $\mathbf{E} = [\,]$ and $\mathbf{M}_0 = \mathbf{x}$...

Inductive Cases

$$\mathbf{E} = [\,] \mid \mathbf{E} - \mathbf{M} \mid \mathbf{V} - \mathbf{E} \mid \mathbf{E} \times \mathbf{M} \mid \mathbf{V} \times \mathbf{E}$$
$$\mid \quad (\mathbf{E}\ \mathbf{M}) \mid (\mathbf{V}\ \mathbf{E}) \mid \texttt{if0}\ \mathbf{E}\ \texttt{then}\ \mathbf{M}\ \texttt{else}\ \mathbf{M}$$

- Assume $\mathbf{M} = \mathbf{M}_1 - \mathbf{M}_2$
    - Assume $\mathbf{M}_1 \neq \mathbf{V}_1$. The only match is $\mathbf{E} = \mathbf{E}_1 - \mathbf{M}_2$. By induction, there is a unique $\mathbf{E}_1[\mathbf{M}_0'] = \mathbf{M}_1$, and $\mathbf{M}_0' = \mathbf{M}_0$.
    - Assume $\mathbf{M}_1 = \mathbf{V}_1$. This matches $\mathbf{E} = \mathbf{E}_1 - \mathbf{M}_2$, but $\mathbf{E}_1$ would have to be $[\,]$ and $\mathbf{M}_0$ would have to be $\mathbf{V}_1$, which is not reducible. So $\mathbf{E} = \mathbf{V}_1 - \mathbf{E}_2$. By induction, there is a unique $\mathbf{E}_2[\mathbf{M}_0'] = \mathbf{M}_2$, and $\mathbf{M}_0' = \mathbf{M}_0$.

Inductive Cases

$$\begin{array}{lll}\mathbf{E} & = & \mathbf{[\,]} \mid \mathbf{E} - \mathbf{M} \mid \mathbf{V} - \mathbf{E} \mid \mathbf{E} \times \mathbf{M} \mid \mathbf{V} \times \mathbf{E} \\ & \mid & (\mathbf{E}\ \mathbf{M}) \mid (\mathbf{V}\ \mathbf{E}) \mid \mathtt{if0}\ \mathbf{E}\ \mathtt{then}\ \mathbf{M}\ \mathtt{else}\ \mathbf{M}\end{array}$$

- Assume $\mathbf{M} = \mathbf{M}_1 \times \mathbf{M}_2$.
  - Analogous to the subtraction case.

Inductive Cases

$$\begin{array}{lll} \mathbf{E} & = & \mathtt{[\,]} \mid \mathbf{E} - \mathbf{M} \mid \mathbf{V} - \mathbf{E} \mid \mathbf{E} \times \mathbf{M} \mid \mathbf{V} \times \mathbf{E} \\ & \mid & (\mathbf{E}\ \mathbf{M}) \mid (\mathbf{V}\ \mathbf{E}) \mid \mathtt{if0}\ \mathbf{E}\ \mathtt{then}\ \mathbf{M}\ \mathtt{else}\ \mathbf{M} \end{array}$$

- Assume $\mathbf{M} = \mathbf{M}_1\ \mathbf{M}_2$.
  - Analogous to the subtraction case.

Inductive Cases

$$\begin{array}{lll} \mathbf{E} & = & \mathbf{[\,]} \mid \mathbf{E} - \mathbf{M} \mid \mathbf{V} - \mathbf{E} \mid \mathbf{E} \times \mathbf{M} \mid \mathbf{V} \times \mathbf{E} \\ & \mid & (\mathbf{E}\ \mathbf{M}) \mid (\mathbf{V}\ \mathbf{E}) \mid \mathtt{if0}\ \mathbf{E}\ \mathtt{then}\ \mathbf{M}\ \mathtt{else}\ \mathbf{M} \end{array}$$

- Assume $\mathbf{M} = \lambda\, \mathbf{x}\, .\, \mathbf{M}_1$.
  - Analogous to the number case.

Inductive Cases

$$\begin{array}{rcl} \mathbf{E} & = & [\,] \mid \mathbf{E} - \mathbf{M} \mid \mathbf{V} - \mathbf{E} \mid \mathbf{E} \times \mathbf{M} \mid \mathbf{V} \times \mathbf{E} \\ & \mid & (\mathbf{E}\ \mathbf{M}) \mid (\mathbf{V}\ \mathbf{E}) \mid \texttt{if0}\ \mathbf{E}\ \texttt{then}\ \mathbf{M}\ \texttt{else}\ \mathbf{M} \end{array}$$

- Assume $\mathbf{M} = \texttt{if0}\ \mathbf{M}_1\ \texttt{then}\ \mathbf{M}_2\ \texttt{else}\ \mathbf{M}_3$. The only match is $\mathbf{E} = \texttt{if0}\ \mathbf{E}_1\ \texttt{then}\ \mathbf{M}_2\ \texttt{else}\ \mathbf{M}_3$.
  - Assume $\mathbf{M}_1 = \mathbf{V}_1$. Then $\mathbf{E}_1 = [\,]$ and there is no non-value $\mathbf{M}_0$.
  - Assume $\mathbf{M}_1 \neq \mathbf{V}_1$. By induction, there is a unique $\mathbf{E}_1[\mathbf{M}_0'] = \mathbf{M}_1$, and $\mathbf{M}_0' = \mathbf{M}_0$.

Inductive Cases

$$\mathbf{E} \;=\; \texttt{[]} \mid \mathbf{E} - \mathbf{M} \mid \mathbf{V} - \mathbf{E} \mid \mathbf{E} \times \mathbf{M} \mid \mathbf{V} \times \mathbf{E}$$
$$\mid \;\; (\mathbf{E}\;\mathbf{M}) \mid (\mathbf{V}\;\mathbf{E}) \mid \texttt{if0}\;\mathbf{E}\;\texttt{then}\;\mathbf{M}\;\texttt{else}\;\mathbf{M}$$

Since we have covered every possible shape of **M**, the lemma is proved.

Handling State

$$
\begin{array}{lcl}
\mathbf{M} & = & \mathbf{...} \\
& | & \mathbf{newref\ M} \\
& | & \mathbf{defref\ M} \\
& | & \mathbf{setref\ M} = \mathbf{M}
\end{array}
$$

$$
\begin{array}{lcl}
\mathbf{E} & = & \mathbf{...} \\
& | & \mathbf{newref\ E} \\
& | & \mathbf{deref\ E} \\
& | & \mathbf{setref\ E} = \mathbf{M} \\
& | & \mathbf{setref\ V} = \mathbf{E}
\end{array}
$$

Handling State

Possible reduction rules:

$$\mathbf{V} \quad = \quad \mathbf{... \mid newref\ V}$$

$$\mathbf{deref}\ (\mathbf{newref\ V}) \quad \to \quad \mathbf{V}$$

$$\mathbf{setref}\ (\mathbf{newref\ V_1}) = \mathbf{V_2} \quad \to \quad \mathbf{newref\ V_2}$$

Example:

$$(\lambda\ \mathbf{r\ .\ deref\ r})(\mathbf{setref}\ (\mathbf{newref} \ulcorner 5 \urcorner) = \ulcorner 12 \urcorner)$$

$$= \quad (\lambda\ \mathbf{r\ .\ deref\ r})(\mathbf{newref} \ulcorner 12 \urcorner)$$

$$= \quad \mathbf{deref}\ (\mathbf{newref} \ulcorner 12 \urcorner)$$

$$= \quad \ulcorner 12 \urcorner$$

Handling State

Possible reduction rules:

$$\begin{array}{lcl} \mathbf{V} & = & \mathbf{\ldots \mid newref\ V} \\ \mathbf{deref}\ (\mathbf{newref\ V}) & \rightarrow & \mathbf{V} \\ \mathbf{setref}\ (\mathbf{newref\ V_1}) = \mathbf{V_2} & \rightarrow & \mathbf{newref\ V_2} \end{array}$$

Problem:

$$\begin{array}{ll} & (\lambda\, \mathbf{r}\ .\ (\lambda\, \mathbf{d}\ .\ \mathbf{deref\ r})(\mathbf{setref\ r} = \lceil 10 \rceil))(\mathbf{newref} \lceil 5 \rceil) \\ = & (\lambda\, \mathbf{d}\ .\ \mathbf{deref}\ (\mathbf{newref}\ 5))(\mathbf{setref}\ (\mathbf{newref} \lceil 5 \rceil) = \lceil 10 \rceil) \\ = & (\lambda\, \mathbf{d}\ .\ \mathbf{deref}\ (\mathbf{newref}\ 5))(\mathbf{newref} \lceil 10 \rceil) \\ = & \mathbf{deref}\ (\mathbf{newref} \lceil 5 \rceil) \\ = & \lceil 5 \rceil \end{array}$$

Handling State

Correct reduction requires a ***store***

$$\sigma = \text{a store address}$$
$$\mathbf{S} = \text{a mapping from } \sigma \text{ to } \mathbf{V}$$
$$\mathbf{V} = \mathbf{...} \mid \sigma$$

$$\langle \mathbf{S}, \lceil \mathbf{n}_1 \rceil - \lceil \mathbf{n}_2 \rceil \rangle \rightarrow \langle \mathbf{S}, \lceil \mathbf{n}_1 - \mathbf{n}_2 \rceil \rangle$$

...

$$\langle \mathbf{S}, \mathbf{newref}\ \mathbf{V} \rangle \rightarrow \langle \mathbf{S}[\sigma = \mathbf{V}], \sigma \rangle$$

where σ is not in **S**

$$\langle \mathbf{S}[\sigma = \mathbf{V}], \mathbf{deref}\ \sigma \rangle \rightarrow \langle \mathbf{S}[\sigma = \mathbf{V}], \mathbf{V} \rangle$$
$$\langle \mathbf{S}[\sigma = \mathbf{V}_1], \mathbf{setref}\ \sigma = \mathbf{V}_2 \rangle \rightarrow \langle \mathbf{S}[\sigma = \mathbf{V}_2], \sigma \rangle$$

Handling State

$\langle \mathbf{S}, \lceil \mathbf{n}_1 \rceil - \lceil \mathbf{n}_2 \rceil \rangle \quad \rightarrow \quad \langle \mathbf{S}, \lceil \mathbf{n}_1 - \mathbf{n}_2 \rceil \rangle$

...

$\langle \mathbf{S}, \mathbf{newref\ V} \rangle \quad \rightarrow \quad \langle \mathbf{S}[\sigma = \mathbf{V}], \sigma \rangle$

where σ is not in **S**

$\langle \mathbf{S}[\sigma = \mathbf{V}], \mathbf{deref}\ \sigma \rangle \quad \rightarrow \quad \langle \mathbf{S}[\sigma = \mathbf{V}], \mathbf{V} \rangle$

$\langle \mathbf{S}[\sigma = \mathbf{V}_1], \mathbf{setref}\ \sigma = \mathbf{V}_2 \rangle \quad \rightarrow \quad \langle \mathbf{S}[\sigma = \mathbf{V}_2], \sigma \rangle$

$\langle \{\}, (\lambda\ \mathbf{r}\ .\ (\lambda\ \mathbf{d}\ .\ \mathbf{deref\ r})(\mathbf{setref\ r} = \lceil 10 \rceil))(\mathbf{newref} \lceil 5 \rceil) \rangle$

$= \quad \langle \{\sigma = \lceil 5 \rceil\}, (\lambda\ \mathbf{r}\ .\ (\lambda\ \mathbf{d}\ .\ \mathbf{deref\ r})(\mathbf{setref\ r} = \lceil 10 \rceil))(\sigma) \rangle$

$= \quad \langle \{\sigma = \lceil 5 \rceil\}, (\lambda\ \mathbf{d}\ .\ \mathbf{deref}\ \sigma)(\mathbf{setref}\ \sigma = \lceil 10 \rceil) \rangle$

$= \quad \langle \{\sigma = \lceil 10 \rceil\}, (\lambda\ \mathbf{d}\ .\ \mathbf{deref}\ \sigma)(\sigma) \rangle$

$= \quad \langle \{\sigma = \lceil 10 \rceil\}, \mathbf{deref}\ \sigma \rangle$

$= \quad \langle \{\sigma = \lceil 10 \rceil\}, \lceil 10 \rceil \rangle$

Handling State

After changing the language, we have to go back and fix the proofs (in principle).

Outline

- Programming with Functions
- Defining a Language
- ➡ Defining Type Rules
- Type Soundness

Type Rules

$$\lceil 5 \rceil : \mathbf{int}$$

$$\lceil 6 \rceil - \lceil 1 \rceil : \mathbf{int}$$

$$(\lambda\, \mathbf{x} . \mathbf{x})(\lceil 8 \rceil) : \mathbf{int}$$

$$(\lambda\, \mathbf{x} . \mathbf{x}) - \lceil 10 \rceil : \textit{no type}$$

$$\mathbf{if0}\, \lceil 0 \rceil\, \mathbf{then}\, \lceil 1 \rceil\, \mathbf{else}\, (\lambda\, \mathbf{x} . \mathbf{x}) : \textit{no type}$$

Type Rules

- arithmetic expressions produce integers

$$\lceil \mathbf{n} \rceil : \mathtt{int}$$

$$\frac{\mathbf{M_1} : \mathtt{int} \qquad \mathbf{M_2} : \mathtt{int}}{\mathbf{M_1} - \mathbf{M_2} : \mathtt{int}}$$

$$\frac{\lceil 5 \rceil : \mathtt{int} \qquad \dfrac{\lceil 3 \rceil : \mathtt{int} \qquad \lceil 1 \rceil : \mathtt{int}}{\lceil 3 \rceil - \lceil 1 \rceil : \mathtt{int}}}{\lceil 5 \rceil - (\lceil 3 \rceil - \lceil 1 \rceil) : \mathtt{int}}$$

Type Rules

- **`if0`**: assume both branches have the same type

$$\frac{\mathbf{M} : \mathtt{int} \qquad \mathbf{M}_1 : \mathbf{T} \qquad \mathbf{M}_2 : \mathbf{T}}{\mathtt{if0}\ \mathbf{M}\ \mathtt{then}\ \mathbf{M}_1\ \mathtt{else}\ \mathbf{M}_2 : \mathbf{T}}$$

$$\frac{\lceil 0 \rceil : \mathtt{int} \qquad \dfrac{\lceil 2 \rceil : \mathtt{int} \qquad \lceil 3 \rceil : \mathtt{int}}{\lceil 2 \rceil + \lceil 3 \rceil : \mathtt{int}} \qquad \lceil 1 \rceil : \mathtt{int}}{\mathtt{if0}\ \lceil 0 \rceil\ \mathtt{then}\ (\lceil 2 \rceil + \lceil 3 \rceil)\ \mathtt{else}\ \lceil 1 \rceil : \mathtt{int}}$$

Type Rules

- What about variables?

$$\mathbf{x}$$

shouldn't have a type

$$\lambda\, \mathbf{x.x}$$

x needs a type, used towards the expression type

- Accumulate variable context in an environment, Γ

$$\Gamma \vdash \mathbf{x} : \mathbf{T} \qquad \text{if } \Gamma(\mathbf{x}) = \mathbf{T}$$

$$\{\mathbf{x} = \mathbf{int}\} \vdash \mathbf{x} : \mathbf{int}$$

Type Rules

- Fix up old rules

$$\Gamma \vdash \lceil \mathbf{n} \rceil : \mathtt{int}$$

$$\frac{\Gamma \vdash \mathbf{M}_1 : \mathtt{int} \qquad \Gamma \vdash \mathbf{M}_2 : \mathtt{int}}{\Gamma \vdash \mathbf{M}_1 - \mathbf{M}_2 : \mathtt{int}}$$

$$\frac{\Gamma \vdash \mathbf{M} : \mathtt{int} \qquad \Gamma \vdash \mathbf{M}_1 : \mathbf{T} \qquad \Gamma \vdash \mathbf{M}_2 : \mathbf{T}}{\Gamma \vdash \mathtt{if0}\ \mathbf{M}\ \mathtt{then}\ \mathbf{M}_1\ \mathtt{else}\ \mathbf{M}_2 : \mathbf{T}}$$

$$\frac{\{\mathbf{x}=\mathtt{int}\} \vdash \lceil 9 \rceil : \mathtt{int} \qquad \{\mathbf{x}=\mathtt{int}\} \vdash \mathbf{x} : \mathtt{int}}{\{\mathbf{x}=\mathtt{int}\} \vdash \lceil 9 \rceil - \mathbf{x} : \mathtt{int}}$$

Type Rules

- Function type: $\mathbf{T}_1 \rightarrow \mathbf{T}_2$

$$\frac{\Gamma\{\mathbf{x}{=}\mathbf{T}'\} \vdash \mathbf{M} : \mathbf{T}}{\Gamma \vdash (\lambda\, \mathbf{x}\, .\, \mathbf{M}) : \mathbf{T}' \rightarrow \mathbf{T}}$$

$$\frac{\Gamma \vdash \mathbf{M}_1 : \mathbf{T}' \rightarrow \mathbf{T} \qquad \Gamma \vdash \mathbf{M}_2 : \mathbf{T}'}{\Gamma \vdash (\mathbf{M}_1\ \mathbf{M}_2) : \mathbf{T}}$$

$$\frac{\dfrac{\{\mathbf{x}{=}\mathtt{int}\} \vdash \mathbf{x} : \mathtt{int}}{\{\} \vdash (\lambda\, \mathbf{x}\, .\, \mathbf{x}) : \mathtt{int} \rightarrow \mathtt{int}} \qquad \lceil 5 \rceil : \mathtt{int}}{\{\} \vdash (\lambda\, \mathbf{x}\, .\, \mathbf{x})(\lceil 5 \rceil) : \mathtt{int}}$$

Type Rules

- One more function example (abbreviate **`int`** with **`i`**)

$$
\dfrac{
\dfrac{
\dfrac{
\begin{array}{c}
\{\mathbf{f}{=}\mathtt{i}{\to}\mathtt{i}\} \vdash \mathbf{f} : \mathtt{i}{\to}\mathtt{i} \\
\{\mathbf{f}{=}\mathtt{i}{\to}\mathtt{i}\} \vdash 5 : \mathtt{i}
\end{array}
}{
\{\mathbf{f}{=}\mathtt{i}{\to}\mathtt{i}\} \vdash \mathbf{f} \lceil 5 \rceil : \mathtt{i}
}
}{
\{\} \vdash (\lambda\, \mathbf{f} \,.\, \mathbf{f} \lceil 5 \rceil) : (\mathtt{i}{\to}\mathtt{i}){\to}\mathtt{i}
}
\qquad
\dfrac{
\dfrac{
\begin{array}{c}
\{\mathbf{y}{=}\mathtt{i}\} \vdash \mathbf{y} : \mathtt{i} \\
\{\mathbf{y}{=}\mathtt{i}\} \vdash \lceil 1 \rceil : \mathtt{i}
\end{array}
}{
\{\mathbf{y}{=}\mathtt{i}\} \vdash \mathbf{y} - \lceil 1 \rceil : \mathtt{i}
}
}{
\{\} \vdash (\lambda\, \mathbf{y} \,.\, \mathbf{y} - \lceil 1 \rceil) : \mathtt{i}{\to}\mathtt{i}
}
}{
\{\} \vdash (\lambda\, \mathbf{f} \,.\, \mathbf{f} \lceil 5 \rceil)(\lambda\, \mathbf{y} \,.\, \mathbf{y} - \lceil 1 \rceil) : \mathtt{i}
}
$$

Type Rules

$$\frac{\Gamma \vdash \mathbf{M} : \mathbf{T}}{\Gamma \vdash \mathbf{newref}\ \mathbf{M} : \mathbf{ref}\ \mathbf{T}} \qquad \frac{\Gamma \vdash \mathbf{M} : \mathbf{ref}\ \mathbf{T}}{\Gamma \vdash \mathbf{deref}\ \mathbf{M} : \mathbf{T}}$$

$$\frac{\Gamma \vdash \mathbf{M}_1 : \mathbf{ref}\ \mathbf{T} \qquad \Gamma \vdash \mathbf{M}_2 : \mathbf{T}}{\Gamma \vdash \mathbf{setref}\ \mathbf{M}_1 = \mathbf{M}_2 : \mathbf{ref}\ \mathbf{T}}$$

$$\dfrac{\dfrac{\dfrac{\{\} \vdash \lceil 5 \rceil : \mathtt{int}}{\{\} \vdash \mathbf{newref}\ \lceil 5 \rceil : \mathbf{ref}\ \mathtt{int}} \qquad \{\} \vdash \lceil 7 \rceil : \mathtt{int}}{\{\} \vdash \mathbf{setref}\ (\mathbf{newref}\ \lceil 5 \rceil) = \lceil 7 \rceil : \mathbf{ref}\ \mathtt{int}}}{\{\} \vdash \mathbf{deref}\ (\mathbf{setref}\ (\mathbf{newref}\ \lceil 5 \rceil) = \lceil 7 \rceil) : \mathtt{int}}$$

Outline

- Programming with Functions
- Defining a Language
- Defining Type Rules
- Type Soundness

Soundness

Theorem: If $\{\} \vdash \mathbf{M} : \mathbf{T}$ then either

- There exists $\mathbf{S}'$ and $\mathbf{V}$ such that $\langle \{\}, \mathbf{M} \rangle \rightarrow \textbf{...} \rightarrow \langle \mathbf{S}', \mathbf{V} \rangle$
- For all $\mathbf{S}'$ and $\mathbf{M}'$, if $\langle \{\}, \mathbf{M} \rangle \rightarrow \textbf{...} \rightarrow \langle \mathbf{S}', \mathbf{M}' \rangle$ then there exists $\mathbf{S}''$ and $\mathbf{M}''$ such that $\langle \mathbf{S}', \mathbf{M}' \rangle \rightarrow \langle \mathbf{S}'', \mathbf{M}'' \rangle$

In other words, an evaluation never gets stuck.

The proof relies on two lemmas: a **preservation lemma** and a **progress lemma**.

Soundness: Preservation

Lemma (Preservation): If

- $\langle \mathbf{S}, \mathbf{M} \rangle \rightarrow \langle \mathbf{S}', \mathbf{M}' \rangle$ and
- $||\mathbf{S}|| \vdash \mathbf{M} : \mathbf{T}$,

then

- $||\mathbf{S}'|| \vdash \mathbf{M}' : \mathbf{T}$

where $||\mathbf{S}||(\sigma) = \mathbf{T}$ if $\mathbf{S}(\sigma) = \mathbf{V}$ and $\{\} \vdash \mathbf{V} : \mathbf{T}$.

Proof: By induction on **M**.

Soundness: Progress

Lemma (Progress): If

- $\mathbf{M}$ is not a $\mathbf{V}$ and
- and $\|\mathbf{S}\| \vdash \mathbf{M} : \mathbf{T}$,

then

- there exist $\mathbf{M}'$ and $\mathbf{S}'$ such that $\langle \mathbf{S}, \mathbf{M} \rangle \rightarrow \langle \mathbf{S}', \mathbf{M}' \rangle$.

Proof: By induction on $\mathbf{M}$.

Soundness Proof Sketch

Lemma: If $\| \mathbf{S} \| \vdash \mathbf{M} : \mathbf{T}$ then either

- There exists $\mathbf{S}'$ and $\mathbf{V}$ such that $\langle \mathbf{S}, \mathbf{M} \rangle \rightarrow \mathbf{...} \rightarrow \langle \mathbf{S}', \mathbf{V} \rangle$
- For all $\mathbf{S}'$ and $\mathbf{M}'$, if $\langle \mathbf{S}, \mathbf{M} \rangle \rightarrow \mathbf{...} \rightarrow \langle \mathbf{S}', \mathbf{M}' \rangle$ then there exists $\mathbf{S}''$ and $\mathbf{M}''$ such that $\langle \mathbf{S}', \mathbf{M}' \rangle \rightarrow \langle \mathbf{S}'', \mathbf{M}'' \rangle$

Proof sketch:

- The Progress Lemma says that we can take a step if we're not yet to a value.
- The Preservation Lemma says that the step preserves the type, so we'll be able to take another step.

Conclusion

- Programming languages are formally defined using algebra
- A language definition comprises
  - a grammar
  - a set of reduction rules
  - an optional set of typing rules
- Soundness ensures that the type rules and reduction rules are consistent