Name:

CS 421 — Type Semantics Activity (Monotype Version)

Mattox Beckman

The Rules

The Language

$L ::=$	$\lambda x.L$	abstractions
$\mid$	$L\ L$	applications
$\mid$	$\texttt{let}\ x = L\ \texttt{in}\ L$	Let expressions
$\mid$	$\texttt{if}\ L\ \texttt{then}\ L\ \texttt{else}\ L$	If expressions
$\mid$	E	expressions
$E ::=$	x	variables
$\mid$	n	integers
$\mid$	b	booleans
$\mid$	$E \oplus E$	integer operations
$\mid$	$E \sim E$	integer comparisons
$\mid$	$E\ \&\&\ E$	boolean and
$\mid$	$E \mid\mid E$	boolean or

The Type Rules

Arithmetic
$$\frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 \oplus e_2 : \texttt{int}}$$

Relations
$$\frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 \sim e_2 : \texttt{bool}}$$

Booleans
$$\frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \texttt{bool}}{\Gamma \vdash e_1\texttt{and}\, e_2 : \texttt{bool}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \texttt{bool}}{\Gamma \vdash e_1\texttt{or}\, e_2 : \texttt{bool}}$$

If
$$\frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \alpha \quad \Gamma \vdash e_3 : \alpha}{\Gamma \vdash \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 : \alpha}$$

Application
$$\frac{\Gamma \vdash e_1 : \alpha_2 \to \alpha \quad \Gamma \vdash e_2 : \alpha_2}{\Gamma \vdash e_1\ e_2\ : \alpha}$$

Abstraction
$$\frac{\Gamma \cup \{x : \alpha_1\} \vdash e : \alpha_2}{\Gamma \vdash \lambda x.e : \alpha_1 \to \alpha_2}$$

Let
$$\frac{\Gamma \vdash e_1 : \alpha_1 \quad \Gamma \cup \{x : \alpha_1\} \vdash e_2 : \alpha_2}{\Gamma \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2\ : \alpha_2}$$

Reductions

Reduce the following programs according to the semantic rules given.

Problem 1)

$\{\texttt{x:Int,y:Int}\} \vdash \textbf{if } x * y > 2 \textbf{ then } x \textbf{ else } y \ : \ Int$

Problem 2)

$\{\texttt{x:Int,y:Int}\} \vdash \textbf{let } m = x * y \textbf{ in } m - x \ : \ Int$

Problem 3)

$\{\} \vdash (\lambda f.\lambda x.f\ x)\ (\lambda x.x)\ 10 \ : \ Int$

Make your own rules!

Problem 4)

Try to write the type rules for HASKELL's `head` and `tail` functions.

Problem 5)

The logical rule for *Modus Ponens* looks like this:

$$\frac{A \to B \quad A}{B}$$

Is there a programming language equivalent to this?[1] Talk to a neighbor and see if you can find a semantic rule that mirrors this.

Problem 6) What happens when you try to type check this code? Try to derive α.

`{y:Int,z:String}` $\vdash (\lambda f.(f y, f z))\,(\lambda x.x) : \ \alpha$

[1] Hint, the answer is "yes".