

# CS415 — Discussion Section Notes 8

Pieter Hooimeijer

2007-03-25

## Outline

1. Housekeeping
2. Operational Semantics

## Housekeeping

- *Written Assignment 6* is due this Thursday at 1pm.
- *Programming Assignment 4* is due this Friday at 11:59pm.

Submit something before **11:59pm on Tuesday** to get an early run on some random testcases. Then resubmit before the Friday deadline (or not).

## Operational Semantics

So far, we've seen two types of judgments:

$$(1) \frac{\begin{array}{c} O \vdash e_0 : T \\ T \leq T_0 \\ O[T_0/x] \vdash e_1 : T_1 \end{array}}{O \vdash \text{let } x : T_0 \leftarrow e_0 \text{ in } e_1 : T_1} \quad (2) \frac{W \vdash T : \text{type}}{W \vdash P\langle T \rangle : \text{type}}$$

We've been using these rules to prove properties of programs, e.g. "this piece of code has type  $T_1$ " or just "this type is valid."

Guiding question: how did these individual rules help us type check entire programs?

Ok. Now we're interested in saying things about the *meaning* of programs. So instead of " : SomeTypeName" we'll say stuff like " $5 + 7 : 12, S$ ." We've already ruled out syntax and type errors, so we can assume any program we see is 'legal.'



Let's do method dispatch as well (while we're here).

$$\begin{array}{l}
so, E, S \vdash e_1 : v_1, S_1 \\
so, E, S_1 \vdash e_2 : v_2, S_2 \\
\vdots \\
so, E, S_{n-1} \vdash e_n : v_n, S_n \\
so, E, S_n \vdash e_0 : v_0, S_{n+1} \\
v_0 = X(a_1 = l_1, \dots, a_m = l_m) \\
imp(X, f) = (x_1, \dots, x_n, e_{body}) \\
l_{x_i} = newloc(S_{n+1}) \quad \text{for } i = 1, \dots, n \\
E' = \emptyset[x_1 : l_{x_1}, \dots, x_n : l_{x_n}, a_1 : l_1, \dots, a_m : l_m] \\
S_{n+2} = S_{n+1}[v_1/l_{x_1}, \dots, v_n/l_{x_n}] \\
v_0, E', S_{n+2} \vdash e_{body} : v, S_{n+3} \\
\hline
so, E, S \vdash e_0.f(e_1, \dots, e_n) : v, S_{n+3}
\end{array}$$