CMSC 22100/32100: Programming Languages

Comments on Homework 1

M. Blume **October 13, 2008**

Here are some things that caught my eye when looking through last week's submissions. This list is not necessarily complete; most likely it is not. But hopefully it will help somewhat.

- In question 1, notice that both arguments to function *append* are lists. (Don't be confused by my poor choice of the letter m for one of the list variables.)

- For question 2: $\rightarrow$ is an "arbitrary" relation in the sense that there *are* inference rules for it, but you don't know what they are. For the proof it is not necessary to know the rules for $\rightarrow$, all you need is the two rules for $\rightarrow^*$ and the knowledge that those are the *only* rules for $\rightarrow *$.

- When doing a proof by induction on the derivation of some judgment j, be careful that you get your steps in the right order: First you do a case-split by considering, in turn, every rule that could have been the last rule used in the derivation for j. In each case you then determine what special properties must hold for j. Also, by inverting the particular rule in question, you find what premises needed to be derived in order to derive j. If such a premise is an instance of j, then you are ready to use the induction hypothesis at that point.

  In any case, my point is: You do the case-split first (determining which rule you consider), then you invert that rule. I have seen at least one attempted solution that somehow first inverted a rule (without justifying why that rule and not some other rule), and which then proceeded to do a case-split (*i.e.*, one which at that point was no longer needed).

- When giving inference rules for the syntactic structure of a language, in order to say that e is an expression we usually do not want to derive e itself but rather some judgment of the form e **exp**.

- When giving an inductive definition of a relation R, don't forget to write the phrase "R is the smallest set such that...."

- When giving an evaluation semantics for a language, it is common that most—if not all—rules have a conclusion of the form $e \Downarrow n$ where e itself is a template for one of the possible syntactic forms. Example: Let the BNF-style rules for some small language be:

$$\begin{array}{rcl} n & \in & \mathbb{Z} \\ e & ::= & n \mid \mathbf{plus}(e,e) \mid \mathbf{times}(e,e) \end{array}$$

Then there ought to be 3 rules in the evaluation semantics: one for n, one for $\mathbf{plus}(e_1, e_2)$, and one for $\mathbf{times}(e_1, e_2)$. For example, it could look like this:

$$\frac{}{n \Downarrow n}\ \textsc{constant} \qquad \frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2 \qquad n = n_1 + n_2}{\mathbf{plus}(e_1, e_2) \Downarrow n}\ \textsc{plus}$$

$$\frac{e_1 \Downarrow n_1 \qquad e_2 \Downarrow n_2 \qquad n = n_1 * n_2}{\mathbf{times}(e_1, e_2) \Downarrow n}\ \textsc{times}$$