CMSC 22100/32100: Programming Languages

Sample solution to Homework 1

M. Blume — Due: October 7, 2008

1. Consider the rules:

$$\frac{}{\textbf{zero nat}}\ \text{ZERO} \qquad \frac{n\ \textbf{nat}}{\textbf{succ}(n)\ \textbf{nat}}\ \text{SUCC} \qquad \frac{}{\textbf{nil list}}\ \text{NIL}$$

$$\frac{n\ \textbf{nat} \qquad l\ \textbf{list}}{\textbf{cons}(n,l)\ \textbf{list}}\ \text{CONS}$$

These rules define a set of terms **nat** representing natural numbers in Peano encoding and a set of terms **list** representing lists of such numbers.

We can inductively (*i.e.*, recursively) define the following *append* function on lists:

$$\begin{array}{rcl} \mathit{append}(\textbf{nil}, m) & = & m \\ \mathit{append}(\textbf{cons}(n,l), m) & = & \textbf{cons}(n, \mathit{append}(l,m)) \end{array}$$

(a) Represent *append* as a ternary relation and give its definition inductively.

9pt

Solution:
Let the relation A be the smallest set such that

i. For every y such that y **list** we have $(\textbf{nil}, y, y) \in A$.

ii. If $(x,y,z) \in A$ and a **nat**, then $(\textbf{cons}(a,x), y, \textbf{cons}(a,z)) \in A$.

(b) Write down a set of inference rules that defines the same ternary relation.

10pt

Solution:

$$\frac{y\ \textbf{list}}{\textbf{append}(\textbf{nil}, y, y)}\ \text{R1} \qquad \frac{\textbf{append}(x,y,z) \qquad a\ \textbf{nat}}{\textbf{append}(\textbf{cons}(a,x), y, \textbf{cons}(a,z))}\ \text{R2}$$

(c) Prove that the so-defined relation is single-valued, *i.e.*, that it represents a binary function.

12pt

To show:
If $\textbf{append}(x, y, z)$ and $\textbf{append}(x, y, z')$, then $z = z'$.
Proof:
By induction on the derivation of $\textbf{append}(x, y, z)$.

Case 1: Rule R1 was used to derive $\textbf{append}(x, y, z)$, so $x = \textbf{nil}$ and $y = z$. Since $x = \textbf{nil}$, rule R1 must also have been used to derive $\textbf{append}(x, y, z')$. Thus, $z' = y = z$.

Case 2: Rule R2 was used to derive $\textbf{append}(x, y, z)$. Thus, $x = \textbf{cons}(a, x_0)$ for some a and x_0, and $z = \textbf{cons}(a, z_0)$ for some z_0. Furthermore, inversion of R2 gives $\textbf{append}(x_0, y, z_0)$. Since $x \neq \textbf{nil}$, R2 must also have been used to derive $\textbf{append}(x, y, z')$. Thus, we have $z' = \textbf{cons}(a, z_0')$ and $\textbf{append}(x_0, y, z_0')$ for some z_0'. Using the induction hypothesis we find that $z_0 = z_0'$. Therefore, $z = \textbf{cons}(a, z_0) = \textbf{cons}(a, z_0') = z'$ as required.

2. (See Chapter 2.1) Let $s \mapsto s'$ be some arbitrary binary relation and let $\mapsto^*$ be defined by the following two inference rules:

$$\frac{}{s \mapsto^* s}\ \textsc{refl} \qquad\qquad \frac{s \mapsto s' \qquad s' \mapsto^* s''}{s \mapsto^* s''}\ \textsc{trans}$$

Prove that $\mapsto^*$ is indeed transitive, *i.e.*, that $\forall s, s', s''.\ s \mapsto^* s' \wedge s' \mapsto^* s'' \Rightarrow s \mapsto^* s''$.

20pt

Solution:
By induction on the derivation of $s \mapsto^* s'$:

Case 1: Rule REFL was used last to derive $s \mapsto^* s'$, so $s = s'$. Thus, trivially, $s \mapsto^* s''$.

Case 2: Rule TRANS was used last to derive $s \mapsto^* s'$. Thus, by inversion of the rule there exists a t such that $s \mapsto t$ and $t \mapsto^* s'$. We use the IH on $t \mapsto^* s'$ and $s' \mapsto^* s''$, finding that $t \mapsto^* s''$. Using rule TRANS in forward direction on $s \mapsto t$ and $t \mapsto^* s''$ lets us conclude that $s \mapsto^* s''$ as required.

3. Consider a language where all values are Peano-encoded natural numbers given by the **nat** judgment from question 1. The expressions e of the language shall be of one of the following forms: **zero** representing the constant 0, $\textbf{succ}(e)$ representing the operation of producing the successor of a given argument, $\textbf{pred}(e)$ representing the operation of producing the *natural* predecessor [1] of a given argument, and $\textbf{if0}(e_1, e_2, e_3)$ representing

[1] The natural predecessor of $n + 1$ is n, and the natural predecessor of 0 is taken to be 0.

a tests of e_1 for being 0, returning the result of e_2 if it is or the result of e_3 if it is not.

6pt (a) Give a definition of e in BNF style.

Solution:

$$e ::= \mathbf{zero} \mid \mathbf{succ}(e) \mid \mathbf{pred}(e) \mid \mathbf{if0}(e, e, e)$$

(b) Give equivalent inference rules for a judgment e **exp** which holds if
8pt e is an expression of the language.

$$\frac{}{\mathbf{zero}\ \mathbf{exp}}\ \text{Z} \qquad \frac{e\ \mathbf{exp}}{\mathbf{succ}(e)\ \mathbf{exp}}\ \text{S} \qquad \frac{e\ \mathbf{exp}}{\mathbf{pred}(e)\ \mathbf{exp}}\ \text{P}$$

$$\frac{e_1\ \mathbf{exp} \qquad e_2\ \mathbf{exp} \qquad e_3\ \mathbf{exp}}{\mathbf{if0}(e_1, e_2, e_3)\ \mathbf{exp}}\ \text{C}$$

(c) Give a set of inference rules for judgments of the form $e \Rightarrow n$ where e is an expression and n is a natural number (in Peano-encoding). The judgment should express the "evaluates-to" relation in the style of a big-step operational semantics and must correspond to the informal
10pt description given above.

The tricky bits are:

- We need two rules for **pred**—one for the case that the argument evaluates to **zero** and one for the case where the argument evaluates to some **succ**(n).
- The rules for **if0** require an explicit premise of the form e_3 **exp** or e_2 **exp** for the sub-term that does not get evaluated. Otherwise the statement to be proved in part (d) would not be true.

Here are the rules:

$$\frac{}{\mathbf{zero} \Rightarrow \mathbf{zero}}\ \text{E-Z} \qquad \frac{e \Rightarrow n}{\mathbf{succ}(e) \Rightarrow \mathbf{succ}(n)}\ \text{E-S}$$

$$\frac{e \Rightarrow \mathbf{zero}}{\mathbf{pred}(e) \Rightarrow \mathbf{zero}}\ \text{E-P(Z)} \qquad \frac{e \Rightarrow \mathbf{succ}(n)}{\mathbf{pred}(e) \Rightarrow n}\ \text{E-P(S)}$$

$$\frac{e_1 \Rightarrow \mathbf{zero} \qquad e_2 \Rightarrow v \qquad e_3\ \mathbf{exp}}{\mathbf{if0}(e_1, e_2, e_3) \Rightarrow v}\ \text{E-C(Z)}$$

$$\frac{e_1 \Rightarrow \mathbf{succ}(n) \qquad e_2\ \mathbf{exp} \qquad e_3 \Rightarrow v}{\mathbf{if0}(e_1, e_2, e_3) \Rightarrow v}\ \text{E-C(S)}$$

(d) Prove that if $e \Rightarrow n$ is derivable, then so is e **exp** as well as n **nat**.

10pt

n **nat** The proof proceeds by induction on the derivation of $e \Rightarrow n$. The cases E-Z and E-P(Z) are trivial by rule ZERO. Case E-C(Z) follows directly from the IH for e_2. Similarly, case E-C(S) follows from the IH for e_3. For case E-S we use the IH on e and then use rule SUCC. For case E-P(S) we use the IH on e and then apply the inversion of rule SUCC. (As discussed in class, this inversion is an admissible rule.)

e **exp** Again, the proof proceeds by induction on the derivation of $e \Rightarrow n$. Case E-Z is immediate by rule Z. Case E-S uses the IH on e and then rule S; cases E-P(Z) and E-P(S) use the IH on e and then rule P. Case E-C(Z) uses the IH on e_1 and e_2 and then applies rule C. Notice that the last step requires to know that e_3 **exp**, which is given by inversion of E-C(Z). Case E-C(S) is analogous to E-C(Z), with the roles of e_2 and e_3 swapped.

15pt

(e) Prove that the relation $\Rightarrow$ defined by your rules is single-valued.

To show:
If $e \Rightarrow n$ and $e \Rightarrow n'$, then $n = n'$.
Proof:
By induction on the derivation of $e \Rightarrow n$.

E-Z: We have $e = \mathbf{zero}$ and $n = \mathbf{zero}$. E-Z must have been used to derive $e \Rightarrow n'$, so $n' = \mathbf{zero} = n$.

E-S: We have $e = \mathbf{succ}(e_0)$, $n = \mathbf{succ}(n_0)$, and $e_0 \Rightarrow n_0$. E-S must have been used to derive $e \Rightarrow n'$, so $n' = \mathbf{succ}(n_0')$ and $e_0 \Rightarrow n_0'$. By IH: $n_0 = n_0'$. Thus, $n = \mathbf{succ}(n_0) = \mathbf{succ}(n_0') = n'$.

E-P(Z): We have $e = \mathbf{pred}(e_0)$, $n = \mathbf{zero}$ and $e_0 \Rightarrow \mathbf{zero}$. Two sub-cases:

E-P(Z) used for $e \Rightarrow n'$: Here $n' = \mathbf{zero} = n$.

E-P(S) used for $e \Rightarrow n'$: Here $e_0 \Rightarrow \mathbf{succ}(n_0)$ for some n_0. By IH this means that $\mathbf{zero} = \mathbf{succ}(n_0)$, which is a contradiction. (This means that E-P(S) could not have been used for $e \Rightarrow n'$ after all.)

E-P(S): We have $e = \mathbf{pred}(e_0)$ and $e_0 \Rightarrow \mathbf{succ}(n)$. Two sub-cases:

E-P(Z) used for $e \Rightarrow n'$: $e_0 \Rightarrow \mathbf{zero}$, so by IH, $\mathbf{zero} = \mathbf{succ}(n)$, *i.e.*, contradiction.

E-P(S) used for $e \Rightarrow n'$: $e_0 \Rightarrow \mathbf{succ}(n')$. By IH: $\mathbf{succ}(n) = \mathbf{succ}(n')$, so $n = n'$.

E-C(Z): We have $e = \mathbf{if0}(e_1, e_2, e_3)$ and $e_1 \Rightarrow \mathbf{zero}$. By reasoning analogous to case E-P(Z) it must be that $e \Rightarrow n'$ also uses rule E-C(Z) (as opposed to E-C(S)). We use the IH on e_2, which gives the desired result.

E-C(S): We have $e = \mathbf{if0}(e_1, e_2, e_3)$ and $e_1 \Rightarrow \mathbf{succ}(n_1)$ for some n_1. By reasoning analogous to case E-P(S) it must be that $e \Rightarrow n'$ also uses rule E-C(S) (as opposed to E-C(S)). We use the IH on e_3, which gives the desired result.