Tutorial 3: Solutions

CSC 467 Compilers and Interpreters
Fall Semester, 2005

1. (a) The translation scheme is :

S → id:=E	{ S.type = ..., S.val = E.val }
S → if E then S_1	{ S.type = ..., S.val = S_1.val }
S → while E do S_1	{ S.type = ..., S.val = S_1.val }
S → S_1 ; S_2	{ S.type = ..., S.val = S_2.val }

   (b) The translation scheme is :

E → E_1 B E_2	{ E.type = if (E_1.type=boolean and E_2.type=boolean) then E_1.type else *type_error* }
B → and | or	{ }
E → not E_1	{ E.type = if (E_1.type=boolean) then E_1.type else *type_error* }
E → E_1 C E_2	{ E.type = if (E_1.type=E_2.type) then boolean else *type_error* }
C → < | > | == |! =	{ }

2. **Base case:** When we have only one procedure call, then there is only one activation record. $D[1]$ points to that activation record, so the procedure will see the correct display. **For** n **calls:** Suppose that at the nth call the procedure called sees the correct display. We will prove the statement for $n+1$ calls.

   For $n+1$ **calls:** We have two cases:

   (a) Procedure p at nesting depth i calls procedure q at nesting depth $j > i$. This means that q is nested within p. The first i elements of the display do not change (remain correct) and $D[j]$ (which is actually $D[i+1]$) points to the new activation record. So q will also see the correct display $D[j]$.

   (b) Procedure p at nesting depth i calls procedure q at nesting depth $j \leq i$. This means that p is nested within q. The first $j-1$ elements of the display as well as $D[i]$ do not change (remain correct). The old value of $D[j]$ is saved in the new activation record and then $D[j]$ points to the new activation record of q. Therefore, $D[j]$, too, points to the correct activation record and q sees the correct display.

3. (a) The values of i and $A[i]$ remain unchanged.

   (b) swap2 successfully swaps the values.

   (c) swap2 successfully swaps the values.

   (d) i is successfully set to $A[i]$ but instead of setting $A[i]$ to i, it sets $A[A[i]]$ to i.

4. (a) D.s

(b) F.w, F.v, D.v

(c) Yes, because D.s depends only on attributes of its siblings to its left (in $F \to FD$) or has no dependencies (in $F \to D$).

(d)
$$
\begin{array}{ll}
F \to & \{D.s = -1\} \\
& D \\
& \{F.w = 1,\ F.v = D.v\} \\
F \to & F_1 \\
& \{D.s = -F_1.w - 1\} \\
& D \\
& \{F.w = F_1.w + 1,\ F.v = F_1.v + D.v\} \\
D \to & 0\ \{D.v = 0\} \\
D \to & 1\ \{D.v = 2^{D.s}\}
\end{array}
$$

(e) Only the changes are shown:

$F \to ND$	$D.s = N.s$
$N \to \epsilon$	$N.s = -1$
$F \to F_1 MD$	$M.i = F_1.w$
	$D.s = M.s$
$M \to \epsilon$	$M.s = -M.i - 1$