CSE 130: Programming Languages – Principles and Paradigms Fall 2000

Problem Set 1 - Solutions

Instructor: Daniele Micciancio — Sept. 27, 2000

Problem 1 (10 points)

a. The following grammar is based on the grammar in the notes, where $+$ and $*$ are also right associative. A solution in which $+$ and $*$ are left associative as in the grammar defined in class is also correct as long as $\uparrow$ is right associative.

$$\begin{array}{lcl} E & \rightarrow & T + E \mid T \\ T & \rightarrow & F * T \mid F \\ F & \rightarrow & B \uparrow F \mid B \\ B & \rightarrow & 0 \mid 1 \mid ... \mid 9 \mid (E) \end{array}$$

b. The parsing trees are the following:

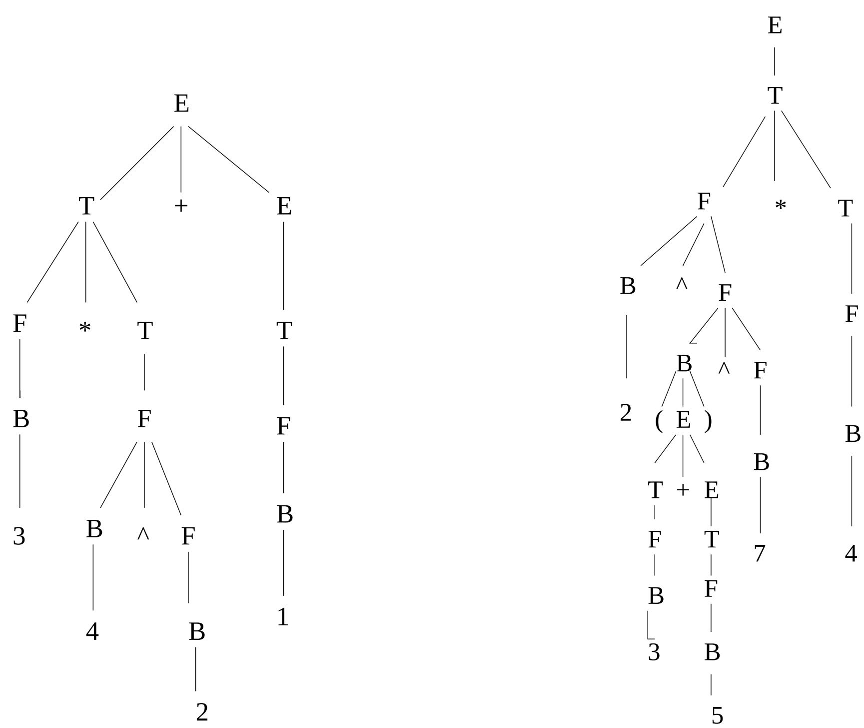

Figure 1: Parse trees for the two expressions

These are the derivation of the two expressions:

$3 * 4 \uparrow 2 + 1$

$$
\begin{array}{ll}
E & \rightarrow \\
T + E & \rightarrow \\
F * T + E & \rightarrow \\
B * T + E & \rightarrow \\
3 * T + E & \rightarrow \\
3 * F + E & \rightarrow \\
3 * B \uparrow F + E & \rightarrow \\
3 * 4 \uparrow F + E & \rightarrow \\
3 * 4 \uparrow B + E & \rightarrow \\
3 * 4 \uparrow 2 + E & \rightarrow \\
3 * 4 \uparrow 2 + T & \rightarrow \\
3 * 4 \uparrow 2 + F & \rightarrow \\
3 * 4 \uparrow 2 + B & \rightarrow \\
3 * 4 \uparrow 2 + 1 &
\end{array}
$$

$2 \uparrow (3 + 5) \uparrow 7 * 4$

$$
\begin{array}{ll}
E & \rightarrow \\
T & \rightarrow \\
F * T & \rightarrow \\
B \uparrow F * T & \rightarrow \\
2 \uparrow F * T & \rightarrow \\
2 \uparrow B \uparrow F * T & \rightarrow \\
2 \uparrow (E) \uparrow F * T & \rightarrow \\
2 \uparrow (T + E) \uparrow F * T & \rightarrow \\
2 \uparrow (F + E) \uparrow F * T & \rightarrow \\
2 \uparrow (B + E) \uparrow F * T & \rightarrow \\
2 \uparrow (3 + E) \uparrow F * T & \rightarrow \\
2 \uparrow (3 + T) \uparrow F * T & \rightarrow \\
2 \uparrow (3 + F) \uparrow F * T & \rightarrow \\
2 \uparrow (3 + B) \uparrow F * T & \rightarrow \\
2 \uparrow (3 + 5) \uparrow F * T & \rightarrow \\
2 \uparrow (3 + 5) \uparrow B * T & \rightarrow \\
2 \uparrow (3 + 5) \uparrow 7 * T & \rightarrow \\
2 \uparrow (3 + 5) \uparrow 7 * F & \rightarrow \\
2 \uparrow (3 + 5) \uparrow 7 * B & \rightarrow \\
2 \uparrow (3 + 5) \uparrow 7 * 4 &
\end{array}
$$

Problem 2 (10 points)

a. The semantics can be defined using the axiom:

(**for var(x) from** E_1 **to** E_2 **do S,** σ)

$\Rightarrow$ (**begin var(x):=**E_1**; while var(x)**$\leq E_2$ **do**

begin var(x):=var(x)+1;S end end, σ)

Notice, this is similar to the last axiom in the semantics given in the notes.

b. Consider the following program:

x:=2; **for** i:=1 **to** x **do** $x := x - 1$

If the upper limit for the counter is evaluated only once, the body of the loop is executed twice. The value of x when the program terminates is 0. If the upper limit is evaluated at each iteration, then the loop is executed only once and the value of x will be 1.

c. This can be seen from the execution of the program under the two different semantics. In both cases we use W, as an abbreviation for

"**while** $y \leq 2$ **do begin**$x := x - 1; y := y + 1$ **end**$''$

Using the semantics defined in the notes (for simplicity some steps are omitted) and using the semantics for :

$$
\begin{array}{ll}
 & (\ x := 2; \textbf{for}\ y\ \textbf{from}\ 1\ \textbf{to}\ x\ \textbf{do}\ x := x - 1, [x : \bot]) \\
\Rightarrow & (\ []; \textbf{for from}\ 1\ \textbf{to}\ x\ \textbf{do}\ x := x - 1, [x : 2]) \\
\Rightarrow & (\ \textbf{for}\ y := 1\ \textbf{to}\ x\ \textbf{do}\ x := x - 1; , [x : 2]) \\
\Rightarrow^2 & (\ \textbf{begin}\ y := 1;\ \textbf{while}\ y \leq 2\ \textbf{do begin}\ x := x - 1; y := y + 1\ \textbf{end end}, [x : 2]) \\
\Rightarrow & (\ \textbf{begin}\ [];\ \textbf{while end}\ y \leq 2\ \textbf{do begin}\ x := x - 1; y := y + 1\ \ \textbf{end}, \textbf{end}[x : 2, y : 1]) \\
\Rightarrow & (\ \textbf{begin while}\ y \leq 2\ \textbf{do begin}\ x := x - 1; y := y + 1\ \textbf{end end}, [x : 2, y : 1]) \\
\Rightarrow & (\ \textbf{while}\ y \leq 2\ \textbf{do begin}\ x := x - 1; y := y + 1\ , [x : 2, y : 1]\ \textbf{end}) \\
\Rightarrow & (\ \textbf{if}\ y \leq 2\ \textbf{then begin begin}\ x := x - 1; y := y + 1; \textbf{end}; W\textbf{end else}\ []\ \ ,\ [x : 2, y : 1]) \\
\Rightarrow^2 & (\ \textbf{if true then begin begin}\ x := x - 1; y := y + 1\ \textbf{end}; W\textbf{end else}\ []\ , [x : 2, y : 1]) \\
\Rightarrow & (\ \textbf{begin begin}\ x := x - 1; y := y + 1\ \textbf{end}; W\ \textbf{end}\ ,\ [x : 2, y : 1]) \\
\Rightarrow^4 & (\ \textbf{begin begin}\ y := y + 1; \textbf{end}\ W\ \textbf{end},\ [x : 1, y : 1]) \\
\Rightarrow & (\ \textbf{begin}\ y := y + 1; W\textbf{end},\ [x : 1, y : 1]) \\
\Rightarrow^3 & (\ \textbf{begin}\ []; W\ \textbf{end},\ [x : 1, y : 2]) \\
\Rightarrow & (\ \textbf{begin}\ W\ \textbf{end},\ [x : 1, y : 2]) \\
\Rightarrow & (W, [x : 1, y : 2]) = (\textbf{while}\ y \leq 2\ \textbf{do begin begin}\ x := x - 1; y := y + 1\ \textbf{end end}, [x : 1, y : 2]) \\
\Rightarrow & (\textbf{if}\ y \leq 2\ \textbf{then begin begin}x := x - 1; y := y + 1; \textbf{end}\ W\ \textbf{end else}\ [], [x : 1, y : 2] \\
\Rightarrow^2 & (\textbf{if true then begin begin}\ x := x - 1; y := y + 1\ \textbf{end}; W\ \textbf{end else}[], [x : 1, y : 2]) \\
\Rightarrow & (\textbf{begin begin}\ x := x - 1; y := y + 1\ \textbf{end}; W\ \textbf{end}, [x : 1, y : 2]) \\
\Rightarrow^4 & (\textbf{begin begin}y := y + 1\ \textbf{end}; W\ \textbf{end}, [x : 0, y : 2]) \\
\Rightarrow & (\textbf{begin}\ y := y + 1; W\ \textbf{begin}, [x : 0, y : 2]) \\
\Rightarrow^2 & (\textbf{begin}\ []; W\ \textbf{end}, [x : 0, y : 3]) \\
\Rightarrow & (\textbf{begin}\ W\ \textbf{end}, [x : 0, y : 3]) \\
\Rightarrow & (\ W\ , [x : 0, y : 3]) = (\textbf{while}\ y \leq 2\ \textbf{do begin}\ x := x - 1; y := y + 1\ \textbf{end}, [x : 0, y : 3]) \\
\Rightarrow & (\textbf{if}\ y \leq 2\ \textbf{then begin begin}\ x := x - 1; y := y + 1\textbf{end}; W\ \textbf{end else}\ [], [x : 0, y : 3] \\
\Rightarrow^2 & (\textbf{if false then begin begin}\ x := x - 1; y := y + 1\textbf{end}; W\ \textbf{end else}\ [], [x : 0, y : 3] \\
\Rightarrow & ([], [x : 0, y : 3]
\end{array}
$$

Using the semantics given at the previous point, and the abbreviation:

$W = \textbf{while } y \leq x \textbf{ do begin } x := x - 1; y := y + 1 \textbf{ end}$

we have:

$$
\begin{array}{ll}
 & (x := 2; \textbf{for } y := 1 \textbf{ to } x \textbf{ do } x := x - 1, [x : \bot]) \\
\Rightarrow & (\textbf{for } y := 1 \textbf{ to } x \textbf{ do } x := x - 1\textbf{end}, [x : 2]) \\
\Rightarrow & (\textbf{begin } y := 1; \textbf{while } y \leq x \textbf{ do begin } x := x - 1; y := y + 1 \textbf{ end}, [x : 2]) \\
\Rightarrow & (\textbf{begin while } y \leq x \textbf{ do begin } x := x - 1; y := y + 1 \textbf{ end end}, [x : 2, y : 1]) \\
\Rightarrow & (\textbf{while } y \leq x \textbf{ do begin } x := x - 1; y := y + 1 \textbf{ end}, [x : 2, y : 1]) \\
\Rightarrow & (\textbf{ if } y \leq x \textbf{ then begin begin } x := x - 1; y := y + 1 \textbf{ end}; W\textbf{end else}[]\ , [x : 2, y : 1]) \\
\Rightarrow^3 & (\textbf{ if true then begin begin } x := x - 1; y := y + 1; \textbf{end}; W \textbf{ end else}[], [x : 2, y : 1]) \\
\Rightarrow & (\textbf{ begin begin } x := x - 1; y := y + 1 \textbf{ end}; W\textbf{end}, [x : 2, y : 1]) \\
\Rightarrow^4 & (\textbf{ begin begin } y := y + 1 \textbf{ end}; W\textbf{end}, [x : 1, y : 1]) \\
\Rightarrow & (\textbf{ begin } y := y + 1; W\textbf{end}, [x : 1, y : 1]) \\
\Rightarrow^4 & (\textbf{ begin } W \textbf{ end}, [x : 1, y : 2]) \\
\Rightarrow & (W, [x : 1, y : 2]) = (\textbf{while } y \leq x \textbf{ do begin } x := x - 1; y := y + 1 \textbf{ end}, [x : 1, y : 2]) \\
\Rightarrow & (\textbf{ if } y \leq x \textbf{ then begin begin } x := x - 1; y := y + 1; \textbf{end}; W \textbf{ end else}[];, [x : 1, y : 2]) \\
\Rightarrow^3 & (\textbf{ if false then begin } x := x - 1; y := y + 1; \textbf{end}; W \textbf{ end else}[];, [x : 1, y : 2]) \\
\Rightarrow & ([], [x : 1, y : 2])
\end{array}
$$

Problem 3 (10 points)

a. $\textbf{wp}(b := a - 3, \{b < 0\}) = \{a - 3 < 0\} = \{a < 3\}$
$\textbf{wp}(a := 2 * b + 1, \{a < 3\}) = \{2 * b + 1 < 3\} = \{b < 1\}$

Therefore, $\textbf{wp}(a := 2 * b + 1; b := a - 3, \{b < 0\}) = \{b < 1\}$.

b. Consider the following annotation of the program:

$\{x \geq 0, y > 0\}$
q:=0;
r:=x;
$\{x = qy + r, 0 \leq r\}$
while (y≤ r) do
begin r:=r-y;
    q:=q+1;
end;
$\{x = qy + r, 0 \leq r < y\}$

First we show that $\{0 \leq x, 0 < y\} q := 0; r := x \{x = qy + r, 0 \leq r\}$ is valid.

Using twice the rule for assignment we have that:

$\{0 \leq x\} q := 0; r := x \{x = qy + r, 0 \leq r\}$ is valid.

Since $\{0 \leq x, 0 < y\} \Rightarrow \{x = 0y + x, 0 \leq x\} = \{0 \leq x\}$ then

$\{0 \leq x, 0 < y\} q := 0; r := x \{x = qy + r, 0 \leq r\}$ is indeed valid.

Next we show that $I = \{x = qy + r, 0 \leq r\}$ is an invariant for the while loop, i.e. we show that $\{I \wedge B\}\ S\ \{I\}$ i.e. we show that:

$\{x = qy + r \;\&\; 0 \leq r \;\&\; y \leq r\} r := r - q; q := q + 1 \{x = qy + r\}$ is valid.

By applying twice the rule for assignment we have that:

$\{x = (q+1)y + r - y, 0 \leq r - y\} r := r - y; q := q + 1 \{x = qy + r, 0 \leq r < y\}$ which is equivalent to:

$\{x = qy + r, y \leq r\} r := r - y; q := q + 1 \{x = qy + r, 0 \leq r < y\}$. And since $\{x = qy + r \;\&\; 0 \leq r \;\&\; y \leq r\} \Rightarrow \{x = qy + r, y \leq r\}$, the statement is true.

We now use the rule for the while statement, and obtain that:

$\{x = qy + r, 0 \leq r\}$
while$(y \leq r)$ do
begin
r:=r-y
q:=q+1
end
$\{x = qy + r, 0 \leq r, r < y\}$

is valid.

It follows that the program is partially correct. For total correctness (i.e. that the program terminates) we have to show the the while loop terminates. For this consider the function f which for state σ of the program returns the value of r in that state. Then $f(\sigma) \geq 0$ since r is grater than zero. Also, f is strictly decreasing. It follows that the loop terminates.

Observe that if the condition $0 < y$ is not satisfied then the function is not decreasing. Indeed, it is easy to see that if y is negative and the program enters the "while" loop then it does not terminate, $0 < y$ is a necessary condition. If x is negative then the program is not correct, take for example $x = -4$ and $y = 2$ the result of the program is not correct.