Assignment 2
Liveness Analysis and Parsing

15-411: Compiler Design
Ruy Ley-Wild (`rleywild@cs`)

Due: Thursday, September 24, 2008 (1:30 pm)

Reminder: Assignments are individual assignments, not done in pairs. The work must be all your own.

You may hand in a handwritten solution or a printout of a typeset solution at the beginning of lecture on Thursday, September 24. Please read the late policy for written assignments on the course web page. If you decide not to typeset your answers, make sure the text and pictures are legible and clear.

Problem 1

[20 points]

```
total :
    1 :  sum ← 0
    2 :  i ← 0
    3 :  j ← 0
loop :
    4 :  size ← length(list)
    5 :  if i ≥ size goto end
    6 :  i ← nth(list, i)
    7 :  j ← size + i
    8 :  k ← nth(list, j)
    9 :  sum ← sum + i
   10 :  goto loop
end :
   11 :  return sum
```

(a) Assuming that function calls have the general form $l : x \leftarrow f(a_1, \ldots, a_n)$, give new rules to define the $\mathsf{def}(l, x)$, $\mathsf{use}(l, x)$ and $\mathsf{succ}(l, l')$ predicates for this new instruction as on page L4.8 of the lectures notes on liveness analysis.

(b) Assuming that function calls always terminate and have no side-effects, give any additional rules you might need to specify $\mathsf{nec}(l, x)$ as on page L5.4 of the lecture notes on dataflow analysis. Briefly explain your answer.

(c) Perform liveness and neededness analysis on the program above, constructing a table showing which variables are live, which variables are necessary, and which variables are needed at each line. Assume function calls always terminate and have no side-effects, so you will need the rules from pages L5.4–5, and any additional rules from parts (a) and (b). Indicate which lines of code are "dead" and thus can be eliminated.

(d) Now suppose any function call may have a side-effect. Give any new rules you might need to specify $\texttt{nec}(l, x)$. Briefly explain your answer.

(e) Repeat part (c), now using rules from L5.4–5, parts (a) and (d).

Problem 2

[20 points]

Consider the following grammar $\mathcal{P}$ for regular expressions:

$$
\begin{aligned}
R &\to R\texttt{*} \\
R &\to R\texttt{|}R \\
R &\to R\texttt{.}R \\
R &\to \texttt{(}R\texttt{)} \\
R &\to \texttt{a} \\
R &\to \texttt{b}
\end{aligned}
$$

(a) Give three derivations of the string `a.b.a.b`: the left-most, the right-most, and one that is neither left- nor right-most.

(b) Give the FIRST and FOLLOW sets for $\mathcal{R}$. Explain whether $\mathcal{R}$ is LL(1).

(c) We can *left factor* $\mathcal{R}$ into an equivalent grammar $\mathcal{Q}$ by factoring the rules with a common prefix.

$$
\begin{aligned}
R &\to RQ \\
R &\to \texttt{(}R\texttt{)} \\
R &\to \texttt{a} \\
R &\to \texttt{b}
\end{aligned}
\qquad\qquad
\begin{aligned}
Q &\to \texttt{*} \\
Q &\to \texttt{|}R \\
Q &\to \texttt{.}R
\end{aligned}
$$

Furthermore we can *eliminate left recursion* from $\mathcal{Q}$ into an equivalent grammar $\mathcal{P}$ by preventing the nonterminal R from appearing on the left of its expansions.

$$
\begin{aligned}
R &\to \texttt{(}R\texttt{)}P \\
R &\to \texttt{a}P \\
R &\to \texttt{b}P
\end{aligned}
\qquad
\begin{aligned}
Q &\to \texttt{*} \\
Q &\to \texttt{|}R \\
Q &\to \texttt{.}R
\end{aligned}
\qquad
\begin{aligned}
P &\to \\
P &\to QP
\end{aligned}
$$

Give the FIRST and FOLLOW sets for $\mathcal{P}$. Explain whether $\mathcal{P}$ is LL(1).

Problem 3

[20 points]

For each grammar explain whether it is LR(1), and if it isn't explain why not.

(a)

$$S \rightarrow A\texttt{\$}$$

$$\begin{array}{lcl} A & \rightarrow & \texttt{x} \\ A & \rightarrow & \texttt{(}A\texttt{)\&(}A\texttt{)} \\ A & \rightarrow & \texttt{(}A\texttt{)|(}A\texttt{)} \\ A & \rightarrow & \texttt{(}A\texttt{)\^{}(}A\texttt{)} \end{array}$$

(b)

$$S \rightarrow T\texttt{\$}$$

$$\begin{array}{lcl} T & \rightarrow & \texttt{a}A\texttt{cd} \\ T & \rightarrow & \texttt{a}B\texttt{ce} \end{array}$$

$$A \rightarrow \texttt{b}$$

$$B \rightarrow \texttt{b}$$

(c)

$$S \rightarrow \texttt{x:=}A\texttt{\$}$$

$$\begin{array}{lcl} A & \rightarrow & \texttt{x:=}A \\ A & \rightarrow & \texttt{n} \end{array}$$

(d)

$$S \rightarrow A\texttt{\$}$$

$$\begin{array}{lcl} A & \rightarrow & \texttt{x} \\ A & \rightarrow & A\texttt{+x} \\ A & \rightarrow & A\texttt{x} \end{array}$$