School of Computer Science, University of Nottingham

COMP2012/G52LAC Languages and Computation, Spring 2019

Dr. Henrik Nilsson

Solutions to Coursework Problems, Set 2

20 March 2019

1. (a) i.

$$\begin{array}{lcll} S & \underset{G}{\Rightarrow} & XYX & \text{by } S \to XYX \\ & \underset{G}{\Rightarrow} & abYX & \text{by } X \to ab \\ & \underset{G}{\Rightarrow} & abcdX & \text{by } Y \to cd \\ & \underset{G}{\Rightarrow} & abcdab & \text{by } X \to ab \end{array}$$

When giving derivation sequences in a context-free grammar, it is normally *not* necessary to justify every single step as it usually is obvious which production was used. The justified derivation sequence above was given for explanatory purposes. Also, when the grammar is clear from the context, it is not necessary to explicitly state in which grammar the derivation is carried out. Thus, answers like the following are perfectly OK too:

$$S \underset{G}{\Rightarrow} XYX \underset{G}{\Rightarrow} abYX \underset{G}{\Rightarrow} abcdX \underset{G}{\Rightarrow} abcdab$$

or

$$S \Rightarrow XYX \Rightarrow abYX \Rightarrow abcdX \Rightarrow abcdab$$

ii.

$$\begin{array}{lcll} S & \underset{G}{\Rightarrow} & YXY & \text{by } S \to YXY \\ & \underset{G}{\Rightarrow} & cXdXY & \text{by } Y \to cXd \\ & \underset{G}{\Rightarrow} & caYbdXY & \text{by } X \to aYb \\ & \underset{G}{\Rightarrow} & cacdbdXY & \text{by } Y \to cd \\ & \underset{G}{\Rightarrow} & cacdbdaYbY & \text{by } X \to aYb \\ & \underset{G}{\Rightarrow} & cacdbdacdbY & \text{by } Y \to cd \\ & \underset{G}{\Rightarrow} & cacdbdacdbcd & \text{by } Y \to cd \end{array}$$

Note: the above is a left-most derivation as the left-most non-terminal is being expanded in each step. But any order of expanding the terminals is fine (unless a left-most or a right-most derivation is required explicitly).

[Marking: 3 and 7 marks respectively, for a total of 10 marks]

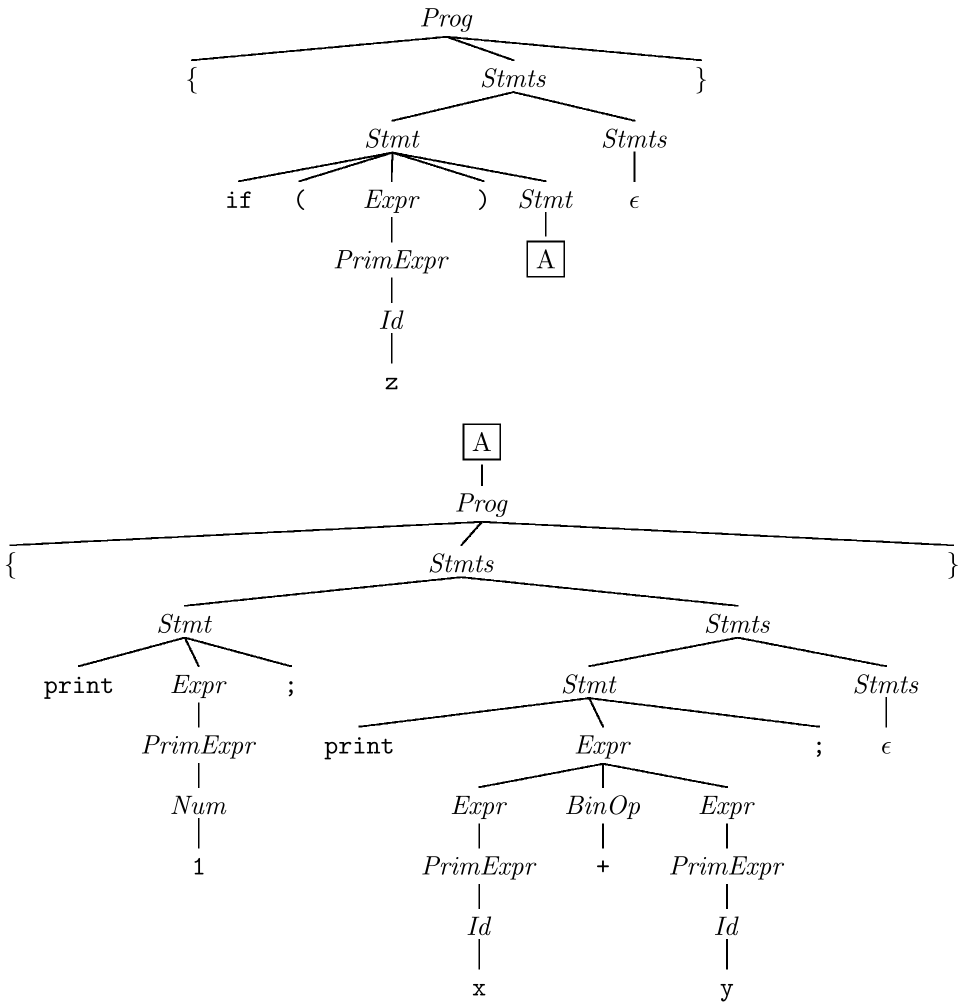
(b) No, $abccddab \notin L(G)$. From the start symbol S, we can either derive XYX or YXY. As the only way to derive words starting with an a is to use the productions for X, the first derivation step must be $S \Rightarrow XYX$. The word $abccddab$ both starts and ends with the word ab. As there is no way to derive ϵ from Y, the production $X \to aYb$ cannot be used to derive ab. Instead the production $X \to ab$ must be used in both cases. Thus we get a derivation $S \Rightarrow XYX \Rightarrow abYX \Rightarrow abYab$. That means that we now must derive $ccdd$ from Y. There are only two productions for Y. We cannot use $Y \to cd$ as we then only would get the word cd, not $ccdd$. And we cannot use $Y \to cXd$ either as any word derivable from X starts with an a and ends with a b. [Marking: 10 marks]

2. (a) The explanations are only for clarifying and not needed for full marks.

   i. Correct
   ii. Not correct (unbalanced parentheses)
   iii. Correct
   iv. Not correct (`x = 0` is a statement (*Stmt*), not an expression (*Expr*), in J).
   v. Correct
   vi. Not correct (missing semicolon)
   vii. Correct
   viii. Not correct (`print` must be followed by an expression (*Expr*))
   ix. Correct
   x. Not correct (every J program must start with an opening brace and end with a closing brace)

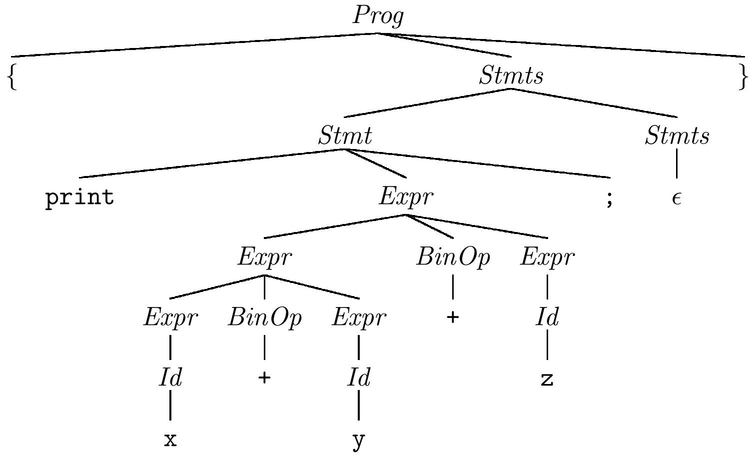
   [Marking: 1 mark each, 10 marks in total]

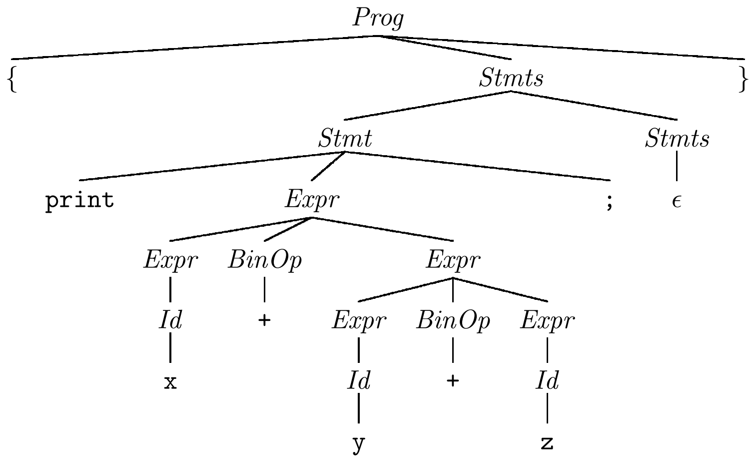
   (b) Derivation tree for { `if (z)` { `print 1 ; print x + y ;` } } :



   [Marking: 10 marks]

(c) Yes, the grammar is ambiguous. For example, { `print x + y + z ;` } is a syntactically valid J program with two different derivation trees:





[Marking: 10 marks]

3. First identify the immediately left-recursive non-terminals. Then group the productions for each such non-terminal into two groups: one where each RHS starts with the non-terminal in question, and one where they don't:

$$\begin{array}{rcl} A & \rightarrow & A\alpha_1 \mid \ldots \mid A\alpha_m \\ A & \rightarrow & \beta_1 \mid \ldots \mid \beta_n \end{array}$$

Then replace those productions with new productions for A and productions for A', where A' is a new name, as follows:

$$\begin{array}{rcl} A & \rightarrow & \beta_1 A' \mid \ldots \mid \beta_n A' \\ A' & \rightarrow & \alpha_1 A' \mid \ldots \mid \alpha_m A' \mid \epsilon \end{array}$$

There are two immediately left-recursive non-terminals in the given grammar: X and Y. The grammar is essentially already grouped as required. Applying the above transformation rule to both the X and Y productions yields:

$$\begin{array}{rcl} S & \rightarrow & aS \mid bX \\ X & \rightarrow & YXdX' \mid YX' \\ X' & \rightarrow & XcX' \mid \epsilon \\ Y & \rightarrow & gY' \\ Y' & \rightarrow & eY' \mid fY' \mid \epsilon \end{array}$$

[Marking: 15 marks]

4. (a) $N_\epsilon = \{S, A, B\}$. A is nullable because $A \to \epsilon$ is a production. B is nullable because $B \to \epsilon$ is a production. S is nullable because $S \to ABB$ is a production and both A and B are nullable. C is not nullable because the right-hand sides of all productions for C include one or more terminals (c or ef), meaning it is clear ϵ cannot be derived from C. For the same reasons as C, D is not nullable. [Marking: 5 marks]

(b) Keeping in mind which non-terminals are nullable, we obtain the following equations:

$$\begin{aligned}
\text{first}(A) &= \text{first}(aA) \cup \text{first}(\epsilon) \\
&= \{a\} \cup \emptyset \\
&= \{a\}
\end{aligned}$$

$$\begin{aligned}
\text{first}(B) &= \text{first}(BCDb) \cup \text{first}(\epsilon) \\
&= (\text{first}(B) \cup \text{first}(CDb)) \cup \emptyset \\
&= \text{first}(B) \cup (\text{first}(C) \cup \emptyset) \\
&= \text{first}(B) \cup \text{first}(C)
\end{aligned}$$

$$\begin{aligned}
\text{first}(C) &= \text{first}(cD) \cup \text{first}(ef) \\
&= \{c\} \cup \{e\} \\
&= \{c, e\}
\end{aligned}$$

$$\begin{aligned}
\text{first}(D) &= \text{first}(dC) \cup \text{first}(fe) \\
&= \{d\} \cup \{f\} \\
&= \{d, f\}
\end{aligned}$$

The solutions of the equations for first(A), first(C), and first(D) are manifest. Recall that an equation of the form $X = X \cup Y$, in the absence of other constraints on X, simplifies to $X = Y$ when we are looking for the smallest solution. The equation for first(B) has the form $X = X \cup Y$ and there are no other constraints on first(B). The smallest solution is thus given by $\text{first}(B) = \text{first}(C) = \{c, e\}$.

Now we can turn to setting up and solving the equation for first(S), again keeping in mind which non-termainals are nullable:

$$\begin{aligned}
\text{first}(S) &= \text{first}(AAS) \cup \text{first}(ABB) \\
&= (\text{first}(A) \cup \text{first}(A) \cup \text{first}(S)) \cup (\text{first}(A) \cup \text{first}(B) \cup \text{first}(B)) \\
&= \text{first}(S) \cup \text{first}(A) \cup \text{first}(B) \\
&= \text{first}(S) \cup \{a\} \cup \{c, e\} \\
&= \text{first}(S) \cup \{a, c, e\}
\end{aligned}$$

Again, an equatiom of the form $X = X \cup Y$, with no further constraints on first(S), meaning that the smallest solution is simply $\text{first}(S) = \{a, c, e\}$.

[Marking: 10 marks]

(c) Note: very detailed account below for clarity. It is sufficient to just state the constraints *according to the definitions* and then simplify.

Constraints for follow(S). Note that S only appear in one RHS, of the production $S \to AAS$, where it appears last; i.e. the string following S is just ϵ, and by definition we have nullable(ϵ). The constraints for S are thus:

$$\begin{aligned}
\{\$\} &\subseteq \text{follow}(S) \\
\text{first}(\epsilon) &\subseteq \text{follow}(S) \\
\text{follow}(S) &\subseteq \text{follow}(S)
\end{aligned}$$

Constraints for follow(A) follow from the productions where A occurs in the RHS, i.e.

$$\begin{array}{rcl} S & \rightarrow & AAS \\ S & \rightarrow & ABB \\ A & \rightarrow & aA \end{array}$$

(note: nullable(S), nullable(B), and nullable(ϵ)):

$$\begin{array}{rcl} \text{first}(A) & \subseteq & \text{follow}(A) \\ \text{follow}(S) & \subseteq & \text{follow}(A) \\ \text{first}(S) & \subseteq & \text{follow}(A) \\ \text{follow}(S) & \subseteq & \text{follow}(A) \\ \text{first}(B) & \subseteq & \text{follow}(A) \\ \text{follow}(S) & \subseteq & \text{follow}(A) \\ \text{first}(\epsilon) & \subseteq & \text{follow}(A) \\ \text{follow}(A) & \subseteq & \text{follow}(A) \end{array}$$

Constraints for follow(B) follow from the productions where B occurs in the RHS, i.e.

$$\begin{array}{rcl} S & \rightarrow & ABB \\ B & \rightarrow & BCDb \end{array}$$

(note: nullable(B), nullable(ϵ), $\neg$nullable(CDb)):

$$\begin{array}{rcl} \text{first}(B) & \subseteq & \text{follow}(B) \\ \text{follow}(S) & \subseteq & \text{follow}(B) \\ \text{first}(\epsilon) & \subseteq & \text{follow}(B) \\ \text{follow}(S) & \subseteq & \text{follow}(B) \\ \text{first}(CDb) & \subseteq & \text{follow}(B) \end{array}$$

Constraints for follow(C) follow from the productions where C occurs in the RHS, i.e.

$$\begin{array}{rcl} B & \rightarrow & BCDb \\ D & \rightarrow & dC \end{array}$$

(note: $\neg$nullable(Db) and nullable(ϵ)):

$$\begin{array}{rcl} \text{first}(Db) & \subseteq & \text{follow}(C) \\ \text{first}(\epsilon) & \subseteq & \text{follow}(C) \\ \text{follow}(D) & \subseteq & \text{follow}(C) \end{array}$$

Constraints for follow(D) follow from the productions where D occurs in the RHS, i.e.

$$\begin{array}{rcl} B & \rightarrow & BCDb \\ C & \rightarrow & cD \end{array}$$

(note: $\neg$nullable(b) and nullable(ϵ)):

$$\begin{array}{rcl} \text{first}(b) & \subseteq & \text{follow}(D) \\ \text{first}(\epsilon) & \subseteq & \text{follow}(D) \\ \text{follow}(C) & \subseteq & \text{follow}(D) \end{array}$$

Using

$$\begin{aligned}
\text{first}(\epsilon) &= \emptyset \\
\text{first}(S) &= \{a, c, e\} \\
\text{first}(A) &= \{a\} \\
\text{first}(B) &= \{c, e\} \\
\text{first}(CDb) &= \text{first}(C) \cup \emptyset \\
&= \{c, e\} \cup \emptyset = \{c, e\} \\
\text{first}(Db) &= \text{first}(D) \cup emptyset \\
&= \{d, f\} \cup \emptyset = \{d, f\} \\
\text{first}(b) &= \{b\}
\end{aligned}$$

and eliminating trivial constraints (of the types $\emptyset \subseteq X$ and $X \subseteq X$) yields:

$$\begin{aligned}
\{\$\} &\subseteq \text{follow}(S) \\
\\
\{a\} &\subseteq \text{follow}(A) \\
\text{follow}(S) &\subseteq \text{follow}(A) \\
\{a, c, e\} &\subseteq \text{follow}(A) \\
\{c, e\} &\subseteq \text{follow}(A) \\
\\
\{c, e\} &\subseteq \text{follow}(B) \\
\text{follow}(S) &\subseteq \text{follow}(B) \\
\{c, e\} &\subseteq \text{follow}(B) \\
\\
\{d, f\} &\subseteq \text{follow}(C) \\
\text{follow}(D) &\subseteq \text{follow}(C) \\
\\
\{b\} &\subseteq \text{follow}(D) \\
\text{follow}(C) &\subseteq \text{follow}(D)
\end{aligned}$$

Noting that $\text{follow}(C) \subseteq \text{follow}(D) \wedge \text{follow}(D) \subseteq \text{follow}(C)$ implies $\text{follow}(C) = \text{follow}(D)$, this is equivalent to:

$$\begin{aligned}
\{\$\} &\subseteq \text{follow}(S) \\
\{a\} \cup \text{follow}(S) \cup \{a, c, e\} \cup \{c, e\} &\subseteq \text{follow}(A) \\
\{c, e\} \cup \text{follow}(S) \cup \{c, e\} &\subseteq \text{follow}(B) \\
\{d, f\} \cup \{b\} &\subseteq \text{follow}(C) = \text{follow}(D)
\end{aligned}$$

which can be further simplified to the final constraints:

$$\begin{aligned}
\{\$\} &\subseteq \text{follow}(S) \\
\{a, c, e\} \cup \text{follow}(S) &\subseteq \text{follow}(A) \\
\{c, e\} \cup \text{follow}(S) &\subseteq \text{follow}(B) \\
\{b, d, f\} &\subseteq \text{follow}(C) = \text{follow}(D)
\end{aligned}$$

[Marking: 15 marks]

(d) The smallest set satisfying the constraint for $\text{follow}(S)$ is obviously just $\{\$\}$. Substituting this into the remaining constraints makes the smallest sets satisfying those obvious too. Thus:

$$\begin{array}{rcl}
\text{follow}(S) & = & \{\$\} \\
\text{follow}(A) & = & \{a, c, e\} \cup \{\$\} = \{a, c, e, \$\} \\
\text{follow}(B) & = & \{c, e\} \cup \{\$\} = \{c, e, \$\} \\
\text{follow}(C) & = & \{b, d, f\} \\
\text{follow}(D) & = & \{b, d, f\}
\end{array}$$

[Marking: 5 marks]