Problem Set 9

Spring 10

Due: Tues 27 April at 2pm in class before the lecture.

Please follow the homework format guidelines posted on the class web page:

`http://www.cs.uiuc.edu/class/sp10/cs373/`

1. CFG design [**Category**: Comprehension, **Points**: 20]

   Consider this *straight line* programming language without any support for loops or conditional statements: A typical straight line program consists of several *sentences* separated by semicolon(;). Three different kinds of sentences are supported:

   - Variable definitions: using keyword *var* we can define a variable, for example:
     var abc
     defines a variable with name abc. Only letters $a \ldots z$ and $A \ldots Z$ are allowed in variable names.
   - Assignments: using symbol *:=* we can assign value of some *expression* to a variable. Left hand side of this symbol is always a variable name and right hand side is always an expression. An expression is composion of variable names and natural numbers using arithmetic operations +, -, * and /. A single natural number or a single variable name is also an expression. Some samples:
     abc := -23
     a:= 2
     v := 12 * v / abc - 4
     v := uvy
   - Output statements: using keyword *print* we can print a variable value to the output. For example:
     print abc
     prints the value of variable abc to the output.

   The language is case sensitive and the only allowed characters are +, -, *, /, ;, :, =, $a \ldots z$, $A \ldots Z$, $0 \ldots 9$, ␣ (space character), $\sim$ (new line).

   Design a context free grammer for this language (that is, construct a grammer G such that $L(G)$ is the set of all valid programs in this language. Don't forget to mention all four important pieces of a CFG explicitly). Note that a program is valid if and only if it conforms to the previous defined rules for this language. Becareful not to add any rules by your intuition, for example we never required a variable name to be declared by the keyword *var* before being used, so for example this is a valid program:
   abc :=5; var Acd ;
   Acd := abc * 6 * pqr; print fg
   Checking those kind of restrictions that are usually needed for serious programming languages requires more machinery.

Solution:

Instead of using capital letters for naming non-terminals, we use this kind of notation: $\langle abc \rangle$ to give a more readable name to a terminal. In the following grammar, $\langle prog \rangle$ is the start symbol. As you are reading the following productions, look back at the previous page and compare them to the given descriptions of the language.

$$
\begin{aligned}
\langle prog \rangle &\to \langle space \rangle \langle prog \rangle \langle space \rangle ; \langle space \rangle \langle prog \rangle \langle space \rangle \mid \langle statement \rangle \mid \langle space \rangle \\
\langle statement \rangle &\to \mathsf{var} \langle space \rangle \langle var\_name \rangle \\
\langle var\_name \rangle &\to \langle var\_name \rangle \langle letter \rangle \mid \langle letter \rangle \\
\langle statement \rangle &\to \langle var\_name \rangle \langle space \rangle \mathsf{:=} \langle space \rangle \langle expression \rangle \\
\langle expression \rangle &\to - \langle space \rangle \langle expression \rangle \mid \langle expression \rangle \langle space \rangle \langle op \rangle \langle space \rangle \langle expression \rangle \\
\langle op \rangle &\to - \mid + \mid * \mid / \\
\langle var\_num \rangle &\to \langle var\_name \rangle \mid \langle number \rangle \\
\langle statement \rangle &\to \mathsf{print} \langle space \rangle \langle var\_name \rangle \\
\langle number \rangle &\to \langle number \rangle \langle digit \rangle \mid \langle digit \rangle \\
\langle digit \rangle &\to \mathsf{0} \mid \cdots \mid \mathsf{9} \\
\langle letter \rangle &\to \mathsf{a} \mid \cdots \mid \mathsf{z} \mid \mathsf{A} \mid \cdots \mid \mathsf{Z} \\
\langle space \rangle &\to \langle space \rangle \langle space \rangle \mid \textvisiblespace \mid \sim
\end{aligned}
$$

2. CFG decoding [**Category**: Comprehension, **Points**: 20]

   (a) Consider the grammar G_1 with the set of productions shown below (S is the start variable). What is $L(G_1)$?

$$S \implies \# \mid 0S1 \mid 1S0$$

Solution:

$L(G_1) = \{w\#(w^R)^c \mid w \in \{0,1\}^*\}$ (x^c represents the complement of binary string x and x^R its reverse).

   (b) Consider the grammar G_2 with the set of productions shown below (S is the start variable). What is $L(G_2)$?

$$
\begin{aligned}
S &\implies \# \mid ASA \\
A &\implies 0 \mid 1
\end{aligned}
$$

Solution:

$L(G_2) = \{w\#x \mid w, x \in \{0,1\}^*, |w| = |x|\}$.

(c) Consider the grammar G_3 with the set of productions shown below (S is the start variable). What is $L(G_3)$?

$$
\begin{aligned}
S &\implies 1B1 \mid 0B0 \mid ASA \\
B &\implies ABA \mid \# \\
A &\implies 0 \mid 1
\end{aligned}
$$

Solution:

$L(G_3) = \{w\#x \mid w, x \in \{0,1\}^*, |w| = |x|, x \neq (w^R)^c\}$.

(Note: B always generates some string of the form $w\#x$ with $|w| = |x|$. So by looking at the productions of the non-terminal S, it will generate something of the form $0w\#x0$ or $1w\#x1$ by at least one of the rules $S \Rightarrow 0B0 \mid 1B1$ and finally produces some string of the form $w'0w\#x0x'$ or $w'1w\#x1x'$ after (possibly many times applying) rule $S \implies ASA$. The matching 0 or 1 prohibits the two strings to be complement of the reverse of each other and that is the only restriction.)

(d) What is the relation between $L(G_1)$, $L(G_2)$ and $L(G_3)$?

Solution:

$L(G_2) = L(G_1) \cup L(G_3)$.

3. CYK [**Category**: Comprehension, **Points**: 20]

Use CYK algorithm to determine whether or not the given string belongs to the grammar. Your answer should include either "yes" or "no" and a chart that you built using CYK.

(a) Which of the following words belong to $L(G_2)$: $aabbb$, $aabab$?

$$
\begin{aligned}
S &\implies AP \mid AB \\
E &\implies AP \mid EB \mid b \\
P &\implies EB \\
A &\implies a \\
B &\implies b
\end{aligned}
$$

Solution:

aabbb - yes, *aabab* - no.

S,E				
∅	S,E,P			
∅	S,E	E,P		
∅	S	E,P	E,P	
A	A	B,E	B,E	B,E
a	a	b	b	b

∅				
∅	∅			
∅	∅	∅		
∅	S	∅	S	
A	A	B,E	A	B,E
a	a	b	a	b

(b) Which of the following words belong to $L(G_1)$: *cadba*, *cbaad*?

$$
\begin{aligned}
S &\implies PE \mid CQ \mid a \\
E &\implies PE \mid CQ \mid a \\
P &\implies EB \\
Q &\implies ED \\
B &\implies b \\
C &\implies c \\
D &\implies d
\end{aligned}
$$

Solution:

cadba - yes, *cbaad* - no.

S,E				
P	∅			
S,E	∅	∅		
∅	Q	∅	∅	
C	S,E	D	B	S,E
c	a	d	b	a

∅				
∅	∅			
∅	∅	∅		
∅	∅	∅	Q	
C	B	S,E	S,E	D
c	b	a	a	d

4. Decidability [**Category**: Proof, **Points**: 20]

Prove that given a CFG G checking whether or not $L(G) \subseteq a^*b^*$ is a decidable problem.

Solution:

$L(G) \subseteq a^*b^* \iff L(G) \cap \overline{a^*b^*} = \emptyset$

We know that testing whether or not the language of a grammar is empty is a decidable problem, so we only need to show that given G we can construct $L(G) \cap \overline{a^*b^*}$ in a finite amount of steps.

To construct a grammar for this intersection we can, for example:

- construct a DFA D for a^*b^*,
- construct a DFA $\overline{D}$ for $\overline{a^*b^*}$
- convert G to PDA P, s.t. $L(G) = L(P)$
- construct PDA P' accepting intersection of a^*b^* and $L(G)$, using P and $\overline{D}$
- convert P' to a grammar G'

5. CNF Conversion [**Category**: Proof., **Points**: 20]

Begin with the grammar G:

$$
\begin{array}{lcl}
S & \rightarrow & aAa \mid bBb \mid \epsilon \\
A & \rightarrow & C \mid a \\
B & \rightarrow & C \mid b \\
C & \rightarrow & CD \mid \epsilon \\
D & \rightarrow & A \mid B \mid ab
\end{array}
$$

(a) Eliminate ϵ-productions, obtaining G_1. (8 Points)

(b) Eliminate any unit productions in G_1, obtaining G_2. (6 Points)

(c) Put G_2 into Chomsky Normal Form G_3. (6 Points)

Solution:

(a) First of all, add a new start variable S_0 with $S_0 \rightarrow S$. The set of nullable variables are $\{S_0, S, A, B, C, D\}$. Adding productions that replace each appearance of nullable variables by ϵ obtains G_1:

$$
\begin{array}{lcl}
S_0 & \rightarrow & S \mid \epsilon \\
S & \rightarrow & aAa \mid bBb \mid aa \mid bb \\
A & \rightarrow & C \mid a \\
B & \rightarrow & C \mid b \\
C & \rightarrow & CD \mid D \\
D & \rightarrow & A \mid B \mid ab
\end{array}
$$

(b) The unit rules in G_1 are: $S_0 \to S, A \to C, B \to C, C \to D, D \to A, D \to B$. After elimination we get G_2:

$$\begin{array}{lcl} S_0 & \to & aAa \mid bBb \mid aa \mid bb \mid \epsilon \\ S & \to & aAa \mid bBb \mid aa \mid bb \\ A & \to & b \mid CD \mid ab \mid a \\ B & \to & a \mid CD \mid ab \mid b \\ C & \to & CD \mid a \mid b \mid ab \\ D & \to & a \mid b \mid CD \mid ab \end{array}$$

(c) We introduce two new rules $P \to a, Q \to b$ to eliminate mixing rules. Then we also introduce $X \to AP, Y \to BQ$ to eliminate long rules:

$$\begin{array}{lcl} S_0 & \to & PX \mid QY \mid PP \mid QQ \mid \epsilon \\ S & \to & PX \mid QY \mid PP \mid QQ \\ A & \to & b \mid CD \mid PQ \mid a \\ B & \to & a \mid CD \mid PQ \mid b \\ C & \to & CD \mid a \mid b \mid PQ \\ D & \to & a \mid b \mid CD \mid PQ \\ X & \to & AP \\ Y & \to & BQ \\ P & \to & a \\ Q & \to & b \end{array}$$

6. Closure Property [**Category**: Proof., **Points**: 20]

Prove the language $L_s = \{a^n b^n c^m \mid m = n - 1 \textit{ or } m = n + 1\}$ is not context-free using only closure properties. (You may assume that $L = \{a^n b^n c^n \mid n \geq 0\}$ is not context-free.)

Solution:

Assume for the sake of contradiction that L_s is context-free, then by closure under concatenation, $L'_s = L_s \circ \{c\} = \{a^n b^n c^m \mid m = n \textit{ or } m = n + 2, m \geq 1\}$ is also context-free. Note that $L_3 = \{a^i b^j c^k \mid (i + j + k) \textit{ is divided by } 3\}$ is a regular language. Since the intersection of a context-free language and a regular language is context-free, we have $(L'_s \cap L_3) \cup \{\epsilon\} = L$ is also context-free. Contradiction!