CpSc 421 Midterm 2 **November 5, 2008**

Do problem 0 and any three of problems 1-4.
If you attempt more than three of problems 1-4, please indicate which ones you want graded – otherwise, I'll make an arbitrary choice.

Graded on a scale of 100 points.
You can attempt from 105 to 110 points depending on which problems you choose. If you score over 100, you get to keep the extra credit.

0. (**5 points**) Your name: Mark Greenstreet Your student #: $9.1093188 \times 10^{-31}$

Question	Score
0	
TOTAL	

1. (**30 points**) Let $G = (V, \Sigma, R, \mathit{Expr})$ be the CFG with

$$\begin{array}{rcl} V & = & \{\mathit{Expr}, \mathit{Variable}, \mathit{Constant}, \mathit{Letter}, \mathit{Digit}\} \\ \Sigma & = & \{\texttt{0}, \texttt{1}, \ldots, \texttt{9}, \texttt{a}, \texttt{b}, \ldots, \texttt{z}, \texttt{+}, \texttt{*}\} \end{array}$$

and rules

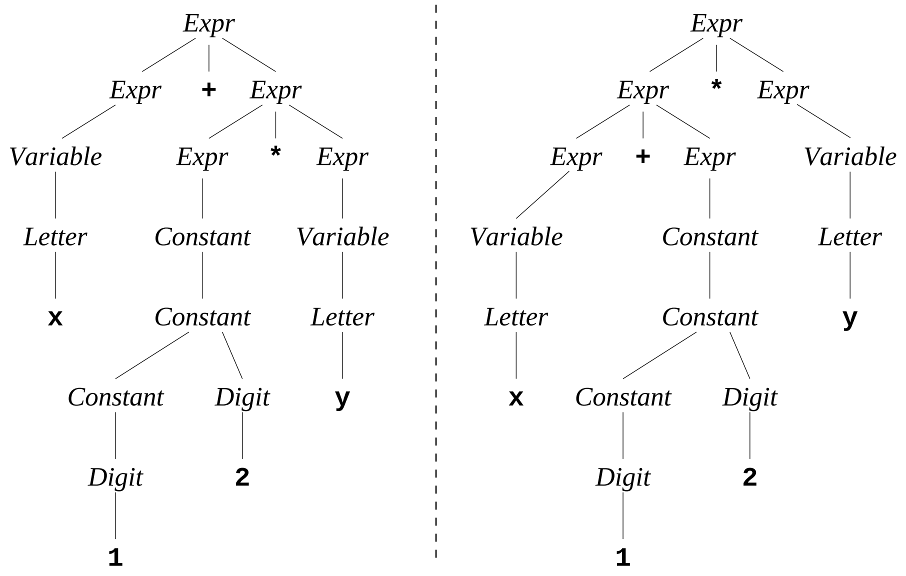
$$\begin{array}{rclclclcl} \mathit{Expr} & \rightarrow & \mathit{Variable} & \mid & \mathit{Constant} & \mid & \mathit{Expr}\ \texttt{+}\ \mathit{Expr} & \mid & \mathit{Expr}\ \texttt{*}\ \mathit{Expr} \\ \mathit{Variable} & \rightarrow & \mathit{Letter} & \mid & \mathit{Variable}\ \mathit{Letter} & \mid & \mathit{Variable}\ \mathit{Digit} & & \\ \mathit{Constant} & \rightarrow & \mathit{Digit} & \mid & \mathit{Constant}\ \mathit{Digit} & & & & \\ \mathit{Letter} & \rightarrow & \texttt{a} & \mid & \texttt{b} & \mid & \ldots & \mid & \texttt{z} \\ \mathit{Digit} & \rightarrow & \texttt{0} & \mid & \texttt{1} & \mid & \ldots & \mid & \texttt{9} \end{array}$$

(a) (**1**5 points) This grammar is ambiguous. Demonstrate this by drawing two different parse trees that generate the string

`x+12*y`

If you use the back side of one of these pages or one of the blank pages at the back, please write "*See page #*" where # is the page number here.

Solution:



(b) (**1**5 points) Write an unambiguous grammar that generates the same language as G. You can just write the rules. If a variable in your grammar has the same rules as a variable in the grammar above, you can write

$$V \quad \rightarrow \quad \text{same as } G$$

Solution:

$$\begin{array}{rclcl} \mathit{Expr} & \rightarrow & \mathit{Term} & \mid & \mathit{Expr}\texttt{+}\mathit{Term} \\ \mathit{Term} & \rightarrow & \mathit{Factor} & \mid & \mathit{Term}\texttt{*}\mathit{Factor} \\ \mathit{Factor} & \rightarrow & \mathit{Variable} & \mid & \mathit{Constant} \\ \mathit{Variable} & \rightarrow & \text{same as } G & & \\ \mathit{Constant} & \rightarrow & \text{same as } G & & \\ \mathit{Letter} & \rightarrow & \text{same as } G & & \\ \mathit{Digit} & \rightarrow & \text{same as } G & & \end{array}$$

The "rules" for Letter and Digit do not need to be included to get full credit.

2. (**35 points**) One of the two languages below is context-free (**20 points**), and the other is not (**15 points**). Identify which is which and justify your answers. For both languages, $\Sigma = \{\mathtt{a}, \mathtt{b}\}$.

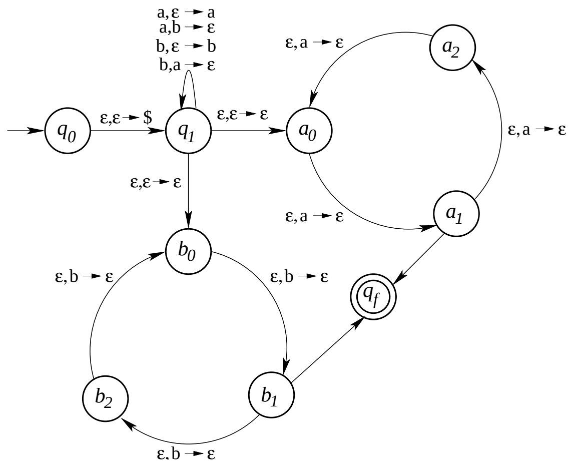
$$\begin{array}{rcl} B_1 & = & \{s \mid (abs(\#a(s) - \#b(s)) \bmod 3) = 1\} \\ B_2 & = & \{s \mid \exists n.\ s = a^n b^{2n} a^n\} \end{array}$$

where $\#a(s)$ indicates the number of a's in s, $\#b(s)$ indicates the number of b's, and $abs(x)$ denotes the absolute value of x.

Solution:

B_1 **is** context free.

Justification 1: We can construct a PDA that recognizes B_1 as shown below:



In state q_1, this machine pushes a's onto the stack to indicate how many more a's there are than b's, or b's onto the stack to indicate how many more b's there are than a's. At the end of the string it transitions to state a_0 if there are more a's than b's; otherwise, it transitions to state b_0. States a_0, a_1 and a_2 determine the number of excess a's (mod 3). If the machine reaches the stack end marker in state a_1, then there were more a's than b's, and the excess has a remainder of 1 when divided by three. This means that the input string is in B_1, and the PDA accepts. The operation when the number of b's exceeds the number of a's is similar.

Justification 2: The language $B_3 = \{s \mid \#a(s) \geq \#b(s)\}$ is context free. For example, it is generated by the rules

$$\begin{array}{lllllll} S & \rightarrow & \mathtt{a}T & \mid & T\mathtt{a} & \mid & SS \\ T & \rightarrow & \epsilon & \mid & \mathtt{a}T\mathtt{b} & \mid & \mathtt{b}T\mathtt{a} \quad \mid \quad TT \end{array}$$

The CFLs are closed under intersection with regular languages and the language

$$B_4 = \{s \mid ((\#a(s) - \#b(s) \bmod 3) = 1\}$$

is regular. Thus, $B_5 = B_3 \cap B_4$ is context free, and we observe that $B_5 = B_1 \cap B_3$ as well. By a similar argument, the language

$$B_6 = \{s \mid \#a(s) \leq \#b(s)\}$$

is context-free, and $B_1 \cap B_6$ is context free as well. Because the CFLs are closed under union,

$$\begin{array}{rl} & (B_1 \cap B_3) \cup (B_1 \cap B_6) \\ = & B_1 \cap (B_3 \cup B_6) \\ = & B_1 \cap \Sigma^* \\ = & B_1 \end{array}$$

is context-free.

B_2 **is not** context free. Let p be a proposed pumping lemma constant.

Let $s = \mathsf{a}^p\mathsf{b}^{2p}\mathsf{a}^p$.

Let u, v, x, y, z be any strings such that $s = uvxyz$, $|vxy| \leq p$ and $|vy| > 1$.

Because $|vxy| \leq p$, it cannot include a's from the first a^p substring and from the last a^p substring. Thus, $uv^0xy^0z = uxz$ is of the form $\mathsf{a}^i\mathsf{b}^j\mathsf{a}^k$ where one of the following three conditions holds:

vy contain symbols from the first a^p. In this case, $i < k$, and $uxz \notin B_2$.

vy contain symbols from the last a^p. In this case, $i > k$, and $uxz \notin B_2$.

vy contains no a's. In this case, $j < 2i$, and $uxz \notin B_2$.

Language B_2 does not satisfy the conditions of the pumping lemma for context-free languages. Therefore B_2 is not context-free.

3. (**35 points**) Define

$$BalancedConcat(A, B) \quad = \quad \{s \mid \exists x \in A, y \in B.\ (|x| = |y|) \wedge s = xy\}$$

(a) (**15 points**) Show an example of a regular language for A and a regular language for B such that $BalancedConcat(A, B)$ is not regular. Give a short justification for your answer (my justification is one sentence long).

Solution: Let $\Sigma = \{\mathsf{a}, \mathsf{b}\}$, $A = L(\mathsf{a}^*)$, and $B = L(\mathsf{b}^*)$. For this choice of A and B, $BalancedConcat(A, B)$ is the set of all strings of the form $\mathsf{a}^n\mathsf{b}^n$ for any $n \geq 0$ and is not context free as we have shown many times.

(b) (**20 points**) Show that for any regular languages A and B, $BalancedConcat(A, B)$ is context free. (my solution is eight sentences long).

Solution: Because A and B are regular, there are DFAs that recognize them. Build a PDA whose finite control includes the transitions for these two DFAs. The PDA starts by pushing an endmarker symbol onto the stack and moving to the initial state of A. While performing transitions of A, the PDA pushes one marker onto the stack for each symbol that it reads. Whenver the PDA is in an accepting state of A, it can make an ϵ-move to the start state of B (with no change to the stack). While performing transitions of B, the PDA pops one symbol off the stack for each symbol read. If the PDA is in an accepting state of B, it can pop the stack-end-marker off of the stack and accept. This machine recognizes $BalancedConcat(A, B)$; therefore, $BalancedConcat(A, B)$ is context free.

4. (**35 points**) Let

$$B = \{M\#w\#n \mid M \text{ describes a Turing machine, } w \text{ describes a string, and } n \text{ is the binary representation of an integer, such that TM } M \text{ halts after at most } n \text{ steps when run with input } w.\}$$

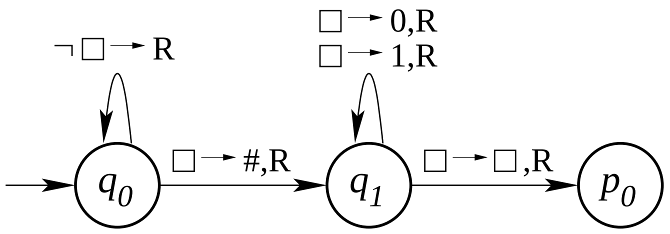
(a) (**1**0 points)
Show that B is Turing decidable. You don't need a detailed proof. It is sufficient to sketch an algorithm for deciding whether or not a string is in B.
(My solution has four sentences.)

Solution: A TM, M_B can first check to make sure that its input is of the form $M\#w\#n$ where M describes a TM, w describes an input string for M, and n is the binary encoding of an integer. It then simulates M running with input w for at most n steps. If M halts by the end of this simulation, M_B accepts. Otherwise, M_B rejects.

Because B is Turing decidable, there is a TM, M_B that decides B. Now, consider a non-deterministic TM, N_H, that on input $M\#w$ scans to the end of the input and appends $\#n$, where n is a string selected non-deterministically from $\{\mathtt{0}, \mathtt{1}\}^*$. Machine N_H then returns its head to the left end of the tape and runs machine M_B on the tape. If M_B accepts, then N_H accepts and if N_H rejects, then N_H rejects.

(b) (**10 points**) Draw the state transition diagram for a non-deterministic TM that appends $\#n$ to the end of its tape, where n can be the binary encoding of *any* integer. Your TM should start in state q_0 and transition to state p_0 when it has finished writing n.

Solution:



(c) (**15 points**) Machine N_H accepts $M\#w$ iff machine M halts when run with input w. Thus, N_H recognizes language $HALT$. We can construct a deterministic Turing machine, M_H, that simulates N_H. We also know that $HALT$ is not decidable. Why isn't M_H (equivalently, N_H) a decider for $HALT$?

Solution: N_H can loop. In particular, it can write an arbitrarily long string for n and never get around to invoking M_B. Likewise, a deterministic TM that simulates N_H can loop, because it can simulate N_H testing longer and longer strings for n.