CSE 30151 Fall 2017 Exam 2

1. (20pt) Multiple Choice/Short Answer (2 points each). For (a) to (e) use as reference the following CFG. Assume lower case letters are terminals.

(a) _____________ What is $\Sigma$?

(b) _____________ What is $V$?

(c) _____________ What is the start variable?

(d) _____________ Show upper bound for $p$.

(e) _________________________ Describe as a set expression $L = \{\ldots\}$ what language is generated by this grammar.

(f) _________________________ Write what a transition for a PDA would look like that requires an “a” on the input, a “b” on the top of the stack and then replaces the top of the stack by a “c”.

(g) _____________ Does a transition for a PDA that looks like $a, \varepsilon \rightarrow b$ represent a push or a pop?

(h) _____________ Multiple transitions from the same PDA state that require the same input and top of stack mean that (a) its an error, (b) you take all of them, (c) you can arbitrarily take any one of them, (d) you take whichever one that is on a path to an accept state, (e) none of the above.

(i) _____________ (True/False) All computer languages can be described by a CFG.

(j) _____________ (True/False) A Turing Machine with non-determinism, multiple tapes, and a stack can solve problems a simple 1-tape TM can’t.
Solutions:

(a) $\Sigma = \{a, b, c, d\}$

(b) $V = \{S, A, B\}$

(c) $S$

(d) $b^{|V|+1} = 3^4 = 81$

(e) $\{a^n b^n c^m d^m | n \geq 0, m \geq 0\}$

(f) $a, b \rightarrow c$

(g) Push

(h) (d)

(i) False

(j) False
2. (20 points - CFLs) Consider the grammar from problem 1: \( S \rightarrow AB, \ A \rightarrow aAb \ | \ \varepsilon, \ B \rightarrow cBd \ | \ \varepsilon. \)

(a) (5 points) Show either a parse tree or a left-derivation sequence for \( abccdd. \)

(b) (5 points) If \( L \) is a CFL, define \( L^R = \{ w^R \mid w \text{ is in } L \} \) (i.e. the set of all strings that are the reversal of some string in \( L \)). Show that \( L^R \) is also a CFL. It is sufficient to describe how to construct \( L^R \) via a CFG constructed from one for \( L \).

(c) (5 points) Assume two language \( L_1 \) and \( L_2 \) are both CFLs. Show that the language \( L_3 = \{ w_a w_b \mid \text{either } w_a \text{ is in } L_1 \text{ and } w_b \text{ is in } L_2, \text{ or } w_a \text{ is in } L_2 \text{ and } w_b \text{ is in } L_1 \} \) is also CF by referring to closure rules for CFGs. Describe how would you construct a CFG of \( L_3. \)

(d) (5 points) Now assume \( L_1 \) is the \( L \) from problem 1, and \( L_2 \) is the \( L^R \) of \( L \) as in part (b) above. Build a grammar for an \( L_3 \) as described in part (c) above.

Solution:
(a) \( S \rightarrow AB \rightarrow aAbB \rightarrow abB \rightarrow abcBd \rightarrow abccBdd \rightarrow abccdd \)

(b) Assume we have a CFG for \( L \). For \( L^R \), reverse the order of all right hand sides of all rules. Note that all rules with one terminal or nonterminal on the right hand side are unchanged. The parse tree will be identical but reversed.

(c) \( L_3 \) is the union of two sets, each of which is the concatenation of two CFLs. Since CFLs are closed under both union and concatenation, \( L_3 \) is CF.

\( \Sigma_3 = \Sigma_1 \cup \Sigma_2. \) Rename the non-terminals so there is no overlap between nonterminals. Assume the start variable for \( L_1 \) is \( S_1 \) and that for \( L_2 \) is \( S_2 \). Invent another nonterminal \( S_3 \) to represent the start variable for \( L_3 \). Then the grammar for \( L_3 \) is all the rules for \( L_1 \) and \( L_2 \) and add:

\[
S_3 \rightarrow S_1 S_2 \\
S_3 \rightarrow S_2 S_1
\]

(d) \( \Sigma \) is the same. Use a ‘ for the \( L^R \) nonterminals. Add new start terminal \( P \):

\[
P \rightarrow SS' \mid S'S \\
S \rightarrow AB \\
A \rightarrow aAb \mid \varepsilon \\
B \rightarrow cBd \mid \varepsilon \\
S' \rightarrow B'A' \\
A' \rightarrow bA'a \mid \varepsilon \\
B' \rightarrow dB'c \mid \varepsilon
\]
3. (20 points - PDA). P is the language over $\Sigma = \{(, )\}$ of matching parenthesis where in any prefix of the string there are never more “)” than “(”, and in the whole string the numbers of “(” and “)” are equal. Give a formal description of a PDA that accepts this. For reference, a CFG for this is as follows, but you do not need to use the CFG to PDA algorithm if you don’t want to! - you can simply write a far simpler PDA directly from the language description. You do not need to include any transitions to reject or trap states.

$$
\begin{align*}
P & \rightarrow ( P ) \\
P & \rightarrow P P \\
P & \rightarrow \varepsilon
\end{align*}
$$

Solution: $\Gamma = \{(, ), \$$\}, states as specified on diagrams. Here’s the simple solution:

Here’s the one based on the book’s algorithm:
Imagine a bird whose nest is on a very tall tree, and who in any one time step can fly the same distance either north (N), south (S), east (E), west (W), up (U), or down (D). We can define different sets of strings of characters from \( \Sigma = \{N, S, E, W, U, D\} \) as languages of different kinds of paths taken by the bird. For each of the following languages, decide if the bird needs only a PDA or a TM for its navigation system, i.e. is the language of paths CF or not. If it is CF, give a CFG. If not, prove via pumping lemma. In either case, describe the language as a set.

(a) (10 points) One such language \( L_1 \) is the set of strings where the bird takes a trip to some location and then backtracks exactly in reverse to get back to the starting point. For example if the path out was UNNDENE, the path back is WSWUSSD, and the string for that path is UNNDENEWSWUSSD.

(b) (10 points) Now consider the language \( L_2 \) that represents ANY path from the bird’s nest that returns the bird back to its nest. The outgoing and return paths may be different. Thus there are equal numbers of Ns and Ss, Es and Ws, and Us and Ds.

Solution:
(a) \( L_1 = \{ww^R|w \text{ from}(N, S, E, W, U, D)^* \text{ and } w^R \text{ is the reverse of } w\} \) is CF: Consider the grammar with start variable \( P \):

\[
\begin{align*}
P & \rightarrow NP_S | SP_N \\
P & \rightarrow EP_W | WP_E \\
P & \rightarrow UP_D | DP_U \\
P & \rightarrow \varepsilon
\end{align*}
\]

Thus the bird needs only a PDA.

(b) \( L_2 = \{w| w \text{ is in } (N, S, E, W, U, D)^* \text{ and there are equal numbers of Ns as Ss, Es and Ws, Us and Ds}\} \). L is not context free.

Consider the string \( s = N^p E^p U^p S^p W^p D^p \). Note any N is separated by at least 2p from any S, and similar for the other combinations.

If L is CF, then any \(|vxy| \leq p\) and there are only a few choices of where \( vxy \) could be:

- Totally within the same stretch of \( p \) symbols that are the same. If here then, pumping up will add to that symbol without adding equally to the matching reverse direction. So that can’t be possible.

- \( vxy \) splits the boundary between two symbol types. Note because each substring is \( p \) long, \( vxy \) can cover only one boundary. Pumping will thus add to sets of two directions. However, opposing directions are around 2p characters apart, so the matching directions in either case cannot be pumped in the same amount.

Thus no such partition exists and the bird needs a TM.
5. (20 points - TM) Consider a Turing Machine that sorts a tape initially holding a string of 0s and 1s so that all 0s’ are moved to the left of the tape and all 1s’ are on the right. For simplicity assume the leftmost character on the tape is always a # and that # appears only there. Length of string is arbitrary (could be just the single #), as is the initial mix of 1’s and 0’s. Thus #0110001 becomes #000111, and #0111 is unchanged.

(a) (5 points) Give a simple “implementation” description of how your machine works.

(b) (10 points) Give a formal description of your machine: your choice of either a state diagram or a transition table. Include a description of what transitioning to accept means. Hint: a solution exists in as little as 6 states. Don’t bother with any transitions to trap or error states.

(c) (5 points) Show a configuration sequence for the initial tape #10.

Solution: Part (a) First a high-level description

(a) Move to the right of the #
(b) Scan right skipping 0s until either a 1 or blank is found. If the latter, accept (string is all 0s).
(c) if a 1 is found scan right past any more 1s to either a 0 or a blank. If the latter, accept (we have no more 0s after the 1).
(d) If a 0, change it to a 1 and scan left to either # or a 0. Move right one square and change to a 0.
(e) Go back to step (b).

Part (b) $\Sigma = \{\#, 0, 1\}, \Gamma = \{\#, 0, 1, \_\}$. Remember $\Sigma$ is a subset of $\Gamma$, and blank is in $\Gamma$

Transition from $q_1$ to accept when we find no 1s in the string.
Transition from $q_2$ to accept where there are no more 0s after the first 1 (i.e. sorted)

Part (c) $q_0 \#10 \implies \# q_1 10 \implies \#1 q_2 0 \implies \# q_3 11 \implies q_3 \#11 \implies \# q_4 11 \implies \#0 q_1 1 \implies \#01 q_2 \implies \#01 accept$

Common errors:
• Forgetting $\Sigma, \Gamma$
• Missed an early blank such as a string that is just # or #11111
• Avoid accepting strings that start with multiple #s or accept # in middle
• Correct syntax for part c: LeftTape state RightTape
• use of $\varepsilon$ in transitions

Several interesting alternative designs. Here is a part a alternative

(a) Scan right past # (reject if not #)
(b) Scan right to 1st 1 or blank. If a 1 change it to “x”. If a blank, accept.
(c) Continue right to a 0 or blank.
   • If a 0, change to a 1. Scan left to x, and change it to 0. Go back to second step.
   • If a blank, scan back to x, change back to 1 and accept.

Here is another slow but simple alternative

(a) Scan right past # (reject if not #)
(b) Scan right past 0s to 1st 1 or blank. If a blank, accept.
(c) At a 1, scan right to 1st 0 or blank.
   • If a blank, accept.
   • If a 0, replace it with a 1. Move left one spot and change that 1 to a 0, and move right
(d) Go back to step 2

Another approach was to copy all 0s to end of tape, then all 1s. Then copy string back over the start of the tape.
6. Blank Sheet. If you use this page identify what problems are here.
Since I am taking this exam later than the other students, I acknowledge that I have neither solicited or received any discussion or information about the test in any form from any other student, and am taking this test fully within the bounds of the Notre Dame Honor Code.

Date: 

Name: