CSE 34151 Spring 2018 Exam 2

- SIGN YOUR NAME ON THE TOP OF ALL PAGES!!!!!!
- Sit so there is an empty chair on both sides of you.
- Do all problems and return all sheets.
- This is an in-class exam, with only open book and this year's class notes permitted. If you have an e-book you may use your computer for that but under the ND Honor Code you may not use the computer for any other purpose, i.e. computer searches, emails, texts, cell phones, or communications with or help from others. All other aspects of the ND Honor code apply. Note you CANNOT include materials such as answer keys from non-class or prior years sources.
- Show all you work in the spaces supplied, including auxiliary equations or definitions that you may have used. **You may use the space below and the blank page at the end for spill-over. Indicate on the original problem page if you use either.**
- Short, clear pictures and diagrams are fine - you need not add lots of explanations.

Problem	Max Points	Points Off
1	30	
2	30	
3	20	
4	20	
Total	100	

BACKGROUND for Problems 1 and 2: Congratulations on being promoted to Chief Scientist of Brainy Birds, Inc. - the world's leader an avian avionics (computing for the birds). Your first responsibility is to determine for each of several new projects what is the minimum type of computing is needed (DFA, PDA, or TM) for bird-borne use. You may assume that all birds so equipped with BB gear are measured each second as to which direction they travel: **u**p, **d**own, **e**ast, **w**est, **n**orth, and **s**outh. Further, the distance all birds fly in a second is always the same regardless of direction. Thus a path taken by a bird is expressed to BB gear as a string, such as *ueeenuwdd*. (Go Up, then East 3 time units, then turn North, then Up again, then West, then Down for 2 time units.)

For each project, answer the following:

1. (5pt) Describe in English the "accepted" set of strings.
2. If you believe the accepted set of strings is a regular language:
   (a) (10 pt) Describe the language formally as a regex
   (b) (12 pt) Implement a matching DFA or NFA. Describe briefly the purpose of the different states.
   (c) (3pt) Indicate the number of states needed.
3. If you believe the accepted set of strings IS NOT regular, but IS a context free language, then:
   (a) (5 pt) Prove the language is NOT regular (using the pumping lemma)
   (b) (5 pt) Define formally a CFG that describes the language.
   (c) (12 pt) Develop a PDA that accepts it. (The PDA need not follow the algorithm for conversions from CFGs, but could be "ad hoc" as this is liable to be simpler and faster). Describe briefly the purpose of the different states.
   (d) (3 pt)How big of a stack is needed?
4. If you believe the accepted set of strings IS NEITHER regular NOR context free, then:
   (a) (10 pt) Prove the language is NOT a CFL (using the pumping lemma)
   (b) (12 pt) Develop a TM that accepts it. Define this TM in **both an implementation description and a formal description**. The formal description may use either a state diagram or a transition table. You can assume that any transition not expressed goes to a trap state. Describe briefly the purpose of the different states.
   (c) (3 pt) How long can the tape grow to?
5. In general, **if you have sets of similar states and/or transitions, feel free to use some shorthand, but explain**.

1. (30pt) Project 1: The CP-1 is a GPS-add-on for **carrier pigeons** that track the path of the bird from its roost to its destination, where the bird signals arrival by pecking the # key on the CP-1. Then when the bird takes off to return, the CP-1 tracks the bird's movements and lights up Green ("accept") and releases a food pellet only if, when the bird lands, it followed the *exact opposite* path of the out-going path. If at any point on the return path, the bird is not following the exact opposite path, the CP-1 should signal "reject." Thus a valid string "accepted" by the CP-1 looks like *unede#wuwsd*, while *uend#suwd* would be rejected. A path of 0 length (string is just #) is acceptable.

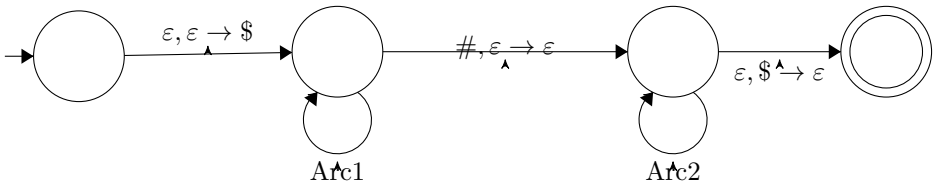
   Solution

   Language: $L = \{t\#t^R | \text{t is a string from } \Sigma^* \text{ and } t^R \text{ is the string reversal of t, with each character "reversed"}\}$ and $\Sigma = \{n, s, e, w, u, d\}$

   The language is not regular. Proof via Pumping Lemma:

   Choose a string $s = U^p \# D^p$. Proof then follows the $0^n 1^n$ path.

   A CFG: $S \rightarrow nSs \mid sSn \mid eSw \mid wSe \mid uSd \mid dSu \mid \#$



   Arc 1 has the following transitions on it

$$n, \varepsilon \rightarrow n$$
$$s, \varepsilon \rightarrow s$$
$$e, \varepsilon \rightarrow e$$
$$w, \varepsilon \rightarrow w$$
$$u, \varepsilon \rightarrow u$$
$$d, \varepsilon \rightarrow d$$

   Arc 1 has the following transitions on it

$$n, s \rightarrow \varepsilon$$
$$s, n \rightarrow \varepsilon$$
$$e, w \rightarrow \varepsilon$$
$$w, e \rightarrow \varepsilon$$
$$u, d \rightarrow \varepsilon$$
$$d, u \rightarrow \varepsilon$$

   Points off:

   - 2 description of string does not include recognition of either complement or reversal
   - 1 Σ must include #
   - 1 pattern for CFG did not start at origin
   - 2 PDA did not include all pairs
   - 1 formal description missing Σ

2. (30 pt) Project 2: The Eagle-1 is designed as a ground-based verifier for a route planner program for blind eagles that will verify that a "fly about" path (no stops) planned to be flown by the eagle *always returns* back to its nest, but with no intermediate constraints on the path as in CP-1. Thus for example the path *unwuwseedd* is accepted, while *wuseedn* is not (not enough w's).

   Solution

   Note for future reuse of problem remind no special character at start of tape

   Language: $L = \{t|$t is a string from Σ^* where the number of n's = number of s's, the number of e's = number of w's, and the number of u's = number of d's $\}$ and $\Sigma = \{n, s, e, w, u, d\}$

   Use the pumping lemma to show it is not context free. Choose $t = u^p e^p d^p w^p$ is in L. Given that $|vxy| \leq p$,

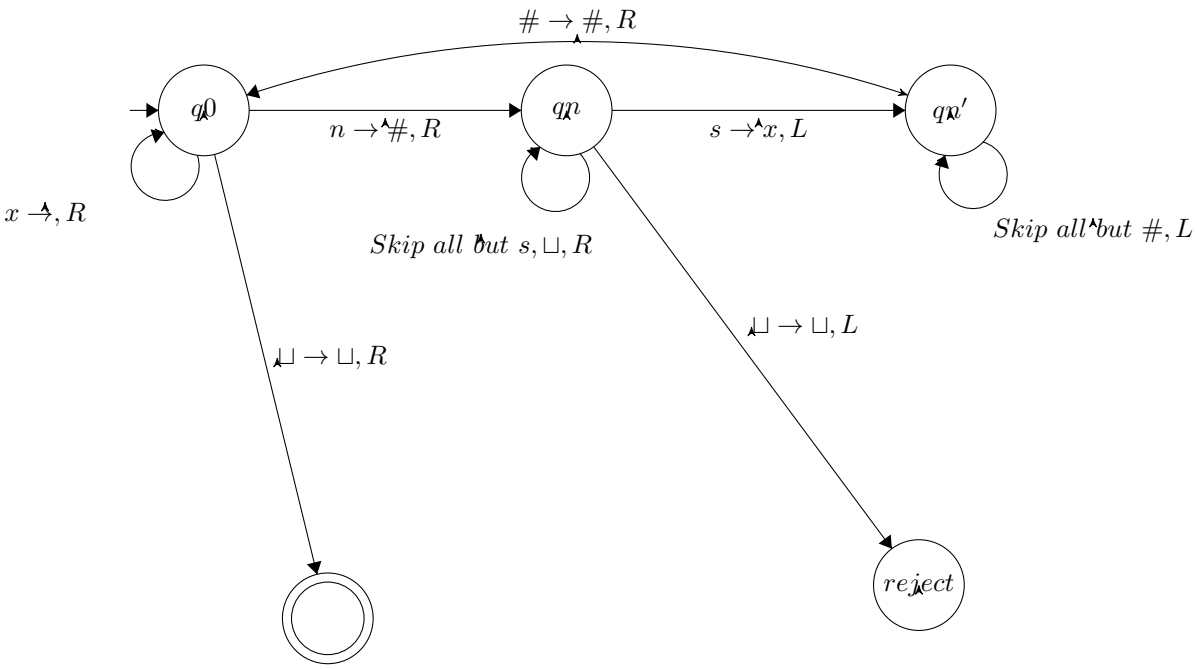
   - vxy could be in any of the sub-strings of the same direction. In this case v and y are the same direction, and pumping it will destroy the balance with the opposite direction
   - vxy could cross the boundary between two neighboring sub-strings, for example $u^p e^p$. Now v starts with a u and y ends with a e. Pumping them changes the number of u's and e's but NOT the matching d's and w's

   Thus L not CFG, and we need a TM.

   Implementation level description of TM:

   (a) Scan from right until first character from Σ. If hit a blank, stop and accept.
   (b) Replace the character by an "#".
   (c) Scan forward looking for the opposite of that character. If not found before a blank, reject.
   (d) If found replace character by a "x"
   (e) Scan left until a #. Stop, move right one cell and restart at step 1.

   State diagram: there are 5 other states like qn for each of the 5 other directions.



   Points off:

   - 2 if you assumed some sort of special leading char on left
   - for pumping something like $n^p s^p$doesn't work.

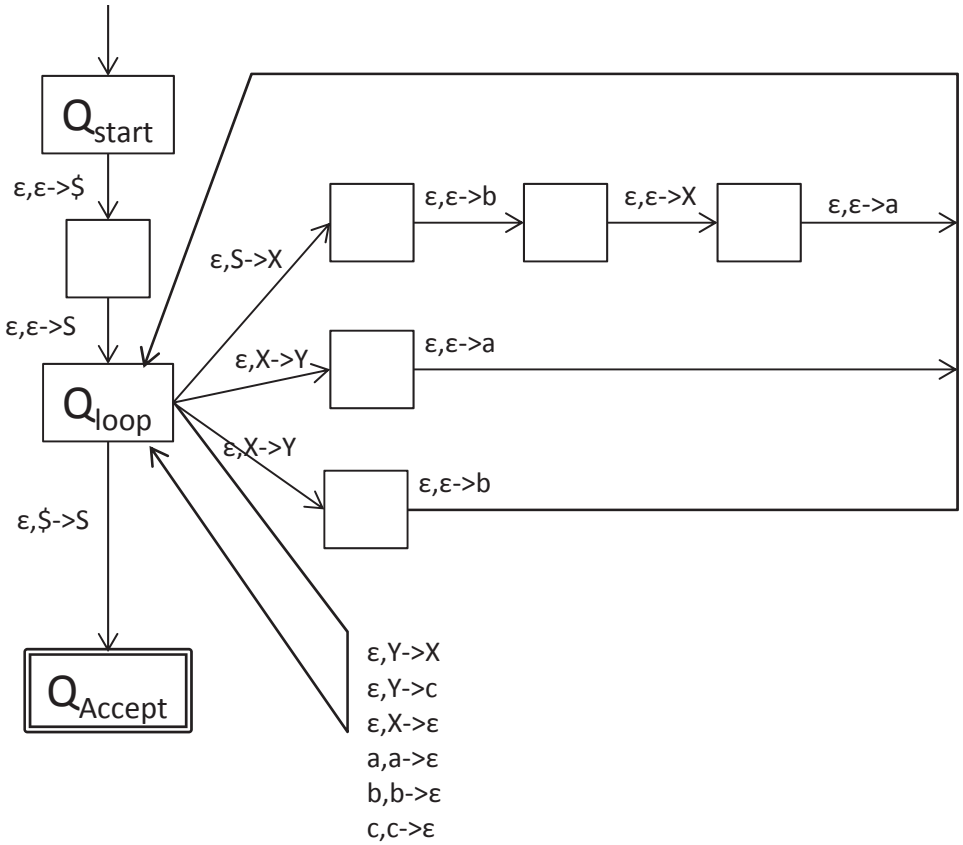
- note that acceptable strings have no required order
- 2 if tape length not the same as original string (infinite not an answer)
- 1 if you assumed an initial blank and then didn't check for 2 blanks
- 1 each for no Σ, Γ
- need to id how you found the left end of the tape

3. (20pt - PDA from CFG) For the grammar below, define formally a PDA that accepts it. You do not need to show any transitions to trap states. You may either draw a state diagram or a table (I suggest the state diagram). You do not need to list all the state names if you draw a diagram. Hint: consider the procedure of Fig. 2.24. Show all states; don't use the shorthand that has multiple stack pushes on one arc. Mark with a "+" all transitions that "push"; mark with a "*" any transition that "pops"; and circle any transition that does both.

$$S \to bXaX$$
$$X \to bY \mid aY \mid \varepsilon$$
$$Y \to X \mid c$$

_____ How may states did you need?

Solution: 9 states. $\Sigma = \{a, b, c\}$, $\Gamma = \{a, b, c, S, X, Y, \$\}$, $F = Q_{accept}$, $q_0 = Q_{start}$.



There is also an 8 state answer.

Points off:

- 2 if not circled or +* push-pop transitions
- 1 for each missing arc
- 2 for pushing backwards
- 3 no markings on transitions
- 2 for extra states with no labels on transitions

4. (20pt - Derivations and Induction) Consider the following CFG G:

$$S \to abBh$$
$$A \to dBg$$
$$B \to cAe \mid m$$

- (5pt) List a left-most derivation of the string $w = abcdcdmgegeh = ab(cd)^2m(ge)^2h$.
- (5pt) Draw a parse tree for w rooted at S.
- (10pt) Then prove by induction that any string of the form $ab(cd)^im(ge)^ih, i \geq 0$ is also a member of $L(G)$ (i.e. this string really is pumpable). Make sure you explicitly state all three steps of the induction (prove basis, state hypothesis, prove hypothesis) Suggestion: to guide your proof look at your derivation or parse tree for where the pattern starts to emerge. Hint: make your induction prove something like showing all strings of the form uv^kXy^kz, where u,v,y, and z are some specific strings and X is a non-terminal, are *derivable* from S. Use that to prove the desired strings are in L(G).

Solution:

Derivation:

$S \Rightarrow abBh \Rightarrow abcAeh \Rightarrow abcdBgeh \Rightarrow abcdcAegeh \Rightarrow abcdcdBgegeh \Rightarrow abcdcdmgegeh = ab(dc)^2m(ge)^2h$

Proof:

Inductive hypothesis: Let $P(k)$ denote that $ab(cd)^kB(ge)^kh$ is derivable from S for all $k \geq 0$.

Basis step: $P(0)$. Show that $ab(cd)^0B(ge)^0h = abxh$ is in $L(G)$.

$$S \Rightarrow abBh$$

Induction hypothesis. Assume for all $i \leq k$, P(i) is derivable from S. Note that this NOT really the same as saying you can get from P(k) to P(k+1), since that is what you want to prove in the next step, i.e. if you know P(i) is true for all i between 0 and k, and you can prove that it is also true for k+1, THEN by induction it must be true for all $i \geq 0$.

Inductive Step: Prove P(k+1) is derivable from S.

We know $P(k) = ab(cd)^kB(ge)^kh$ is derivable from S. Using the rules for B and A we get:

$$ab(dc)^kB(ge)^kh \Rightarrow ab(dc)^kcAe(ge)^kh \Rightarrow ab(dc)^kcdBge(ge)^kh \;=\; ab(dc)^{k+1}Bge(ge)^{k+1}h$$

So we have proved hypothesis. Thus for all k: $ab(cd)^kB(ge)^kh$ is derivable from S. But the last B rule then allows us to also say that $ab(cd)^km(ge)^kh$ is derivable from S and thus all such strings are in L(G).

Thus $P(k+1)$ is in $L(G)$ because we can show a parse tree from S.

I took off 3 points if you did not show how the strings were generated and then went back together again (esp. where x comes from and how v and y are re-assembled from the various pieces).

Note also that there was one clever alternative presented by a student who

- started with the string $ab(bbaa)^{k+1}bba(ba)^{k+1}$,
- expanded it to $ab(bbaa)^kbbaabbaba(ba)^k$
- noted that $bbaabbaba$ in the middle is derivable from $B \to bbAa \to bbaaBba \to bbaabbAaba \to bbaabbaba$
- thus it can be written as derivable from $ab(bbaa)^kB(ba)^k$
- but the alternate derivation of B yields $ab(bbaa)^kbbAa(ba)^k \to ab(bbaa)^kbba(ba)^k$ which is exactly P(k) which we know is in L.

- then P(k+1) is also in L.

Points off:

- 1 if induction doesn't state with 0
- 3 for bad basis
- 2 for bad hypothesis
- up to 5 for bad proof

This is a blank sheet for your use. Label which problem you are using it for.