Extra Credit

Note: All problems on this homework set are extra-credit. You may turn in solutions for up to four of the problems below. Turning in a solution for any part of a problem counts as attempting the entire problem.
Have fun!

1. (**20 points**) Let A be a CFL and B be regular. Prove that $A \cap B$ is a CFL.

2. (**30 points**) Let A be the language $\{x \mid \exists w.\ x = ww\}$. We showed in class that A is not a CFL. Prove that $\overline{A}$ *is* a CFL.

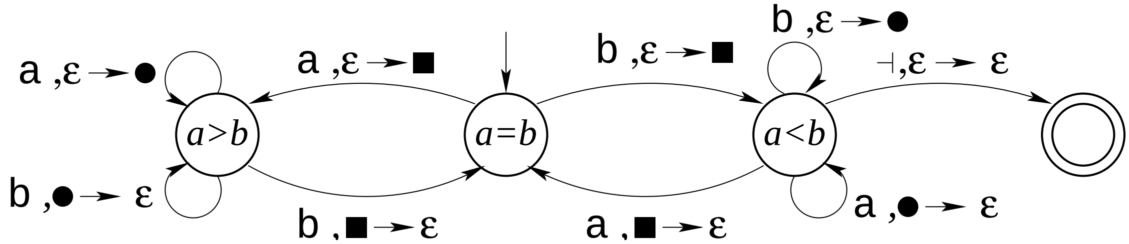
3. (**30 points**) A DPDA is a deterministic, pushdown automaton. Formally, a DPDA, $D = (Q, \Sigma, \Gamma, \delta, q_0, F)$. From any configuration, a DPDA has exactly one possible move. We need to allow ϵ moves so that in appropriate situations the DPDA can push more symbols onto the stack than it pops off. If the DPDA has an ϵ move possible, then there must only be one such move, and no other move may be possible. Finally, the input alphabet includes a special symbol, $\dashv$. This is an endmarker – the last symbol of any input string is $\dashv$ and $\dashv$ may not appear before the last symbol of the string.

   We can describe DPDAs by drawing transition diagrams just as we did for (non-deterministic) PDAs. For a DPDA, there may not be multiple arrows out of a state that could be taken for the same input symbol and stack symbol.

   (a) (**10 points**) Draw the transition diagram for a DPDA that accepts

   $$\{w \in \{\mathsf{a}, \mathsf{b}\}^* \mid \#\mathsf{a}(w) < \#b(w)\}$$

   Solution:



   The PDA has three states to keep track of whether the string read so far more a's than b's (state $a > b$), the same number (state $a = b$), or fewer (state $a < b$). If the PDA is in state $a < b$ and encounters the end of the input string, it accepts.

   (b) (**10 points**) Prove that the class of languages accepted by DPDAs is closed under complement.

   Solution: Let $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a DPDA (note that δ is a function, $\delta : Q \times \Sigma \times \Gamma \ \rightarrow\ Q \times \Gamma$). Let $\overline{P} = (Q, \Sigma, \Gamma, \delta, q_0, \overline{F})$. Because P can only reach one state when reading any given string, $L(\overline{P}) = \overline{L(P)}$.

   (c) (**10 points**) Prove that the class of languages recognized by DPDAs is not closed under union.

   Solution: A DCFL is a language recognized by a DPDA. Clearly every DCFL is recognized by an ordinary (i.e. non-deterministic) PDA as well and is therefore a CFL. Let

$$\begin{array}{rcll} A & = & \mathsf{a}^i\mathsf{b}^j\mathsf{c}^*, & i \neq j \\ B & = & \mathsf{a}^i\mathsf{b}^*\mathsf{c}^j, & i \neq j \\ C & = & \overline{A \cup B} \ \cap\ \mathsf{a}^*\mathsf{b}^*\mathsf{c}^* & \end{array}$$

Language C is the same as $\mathsf{a}^n\mathsf{b}^n\mathsf{c}^n$ which is not a CFL. As shown in probem 1, the CFLs are closed under intersection with regular languages; thus, $\overline{A \cup B}$ is not a CFL, and therefore not a DCFL. As shown in part (b), the DCFLs are closed under complement; thus $A \cup B$ is not a DCFS. Languages A and B are both DCFLs – the DPDAs that recognize them are similar to the one from part (a). This shows that the DCFSs are note closed under union.

4. (**20 points**) Let $A_c = \{M\#w \mid \text{ TM } M \text{ writes symbol } c \text{ on its tape when run with input } w\}$. Show that the language A_c is Turing undecidable.

Solution: I will reduce the A_{TM} to A_c. Let M be a description of a Turing Machine, and w be a string. We use M to create the description of a new Turing machine as follows:

: We add a new symbol, c' to the tape alphabet.

: For any transition of M that writes a c to the tape, M' will write c'. Machine M' does the same thing when reading a c' as M does when reading c.

: We add a new state, q_c to the set of states. We replace transitions of M that move to its accept state with a transition to q_c.

: In state q_c, machine M' writes a c on the tape, moves the head to the right, and moves to the accept state.

Machine M' writes a c on its tape when run with input w iff machine M accepts w. The construction of the description of M' from the description of M is computable by a Turing machine. Thus, we have reduced A_{TM} to A_c. A_{TM} is undecidable; therefore, A_c is undecidable as well.

5. (**30 points**) Let $\mathit{NoWriteOverInput}$ be the class of Turing machines that may not alter their input strings. If $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ is a Turing machine, then $M \in \mathit{NoWriteOverInput}$ iff for every $c \in \Sigma$ and $q \in Q$:

$$\delta(q, c) \quad = \quad (q', c, d)$$

for some $q' \in Q$ and some $d \in \{L, R\}$ (note that M writes the same symbol that it read).

(a) (**15 points**) Let M be a Turing machine in $\mathit{NoWriteOverInput}$. Prove that $L(M)$ is regular.

Solution: I'll show that $L(M)$ has a finite number of Myhill-Nerode equivalence classes (see the Sept. 29 notes). This shows that $L(M)$ is regular.
Let Q be the set of state of M. Let $C = Q \times (Q \rightarrow Q)$. Each equivalence class of $L(M)$ corresponds to an element of C. In particular, the equivalnce class for w is indexed by the tuple (q, f) if when M is run on input w, it is in state q the first time that it leaves w to the right, and if it returns to the last symbol of w in state q', it will be in state $f(q')$ the next time that it leaves w to the right. If M accepts before leaving w, then q (resp. $f(q')$) is the accept state. Likewise, if M rejects before leaving w or loops while reading w, then q (resp. $f(q')$) is the reject state. From this construction, if x_1 and x_2 are members of the same equivalence class and y is an arbitrary string, then M accepts x_1y iff it also accepts x_2y. This shows that $L(M)$ is regular.

(b) (**15 points**) Prove that the language $\{M\#w \mid M \text{ accepts } w\}$ is undecidable even if we restrict M to be in $\mathit{NoWriteOverInput}$.

Solution: Let A'_{TM} be the language described above. I'll reduce A_{TM} to A'_{TM}.
First, note that we can't create new tape symbols, i.e. a' for a, b' for b, etc., and make a "primed" copy of the input string in the initially blank portion of the tape. This is because M can't mark symbols of w to keep track of where it is in the copying process.
Here's what we do instead. Given a description of a Turing machine, M' and an input string w', we will create a new TM, M" that does the following:

1. Scans to the right until it encounters a blank.
2. M" overwrites the first blank with a $\vdash$ (i.e. a new tape symbol that will act as a left end marker).

3. M" writes w' onto its tape, following the $\vdash$.
4. M" moves its head back to the first symbol of w'.
5. M" runs M' with the only change that if it every reads a $\vdash$, this means that M' was at the left end of its tape and tried to move left. M" moves the head back to the right, stays in the same state, and keeps the $\vdash$ where it was on the tape.

By this construction, M" recognizes Σ^* if M' accepts w', and M" recognizes $\emptyset$ otherwise. Thus, $w' \in L(M')$ iff M" accepts w" for *any* string w". The construction of M" from M' and w' is clearly Turing computable; thus, I've reduced A_{TM} to A'_{TM} and have therefore shown A'_{TM} to be undecidable.