CS 181 FINAL EXAM **NAME** ________________________________
SPRING 2015 **UCLA ID** ________________________________

You have 3 hours to complete this exam. You may assume without proof any statement proved in class.

(2 pts) **1** Give a high-level description of a Turing machine that computes the function $f\colon 1^* \to 1^*$ given by $f(1^n) = 1^{n^2}$.

Solution. The Turing machine will use two tapes. Initially, the first tape contains the input 1^n, and the second is blank. The Turing machine marks the first cell of the input with a dot and then copies the entire input to the second tape symbol by symbol (using an additional moving mark to implement the copying). The machine then advances the dot to the next input symbol and again copies the entire input string to the second tape, appending it to the second tape's contents. The process repeats until the dot moves beyond the last input symbol. At that point, the second tape contains the string 1^{n^2}, which can then be copied symbol by symbol to the input tape, overwriting the input.

(3 pts) **2** Let M be a Turing machine that never moves its head beyond the 2015th cell on any input. Prove that M recognizes a regular language.

Solution. The key observation is that for any input string w, the symbols of w at indices $2016, 2017, 2018, \ldots$ are never seen by M and thus can be dropped without affecting M's output. To recognize $L(M)$ with a DFA, we simply memorize the input string up to the first 2015 symbols and announce M's output on the memorized string. Formally, the DFA is given by $(\{\varepsilon\} \cup \Sigma \cup \Sigma^2 \cup \cdots \cup \Sigma^{2015}, \Sigma, \delta, \varepsilon, F)$, where $F = \{w : M \text{ accepts } w\}$ and

$$\delta(w, \sigma) = \begin{cases} w\sigma & \text{if } |w| < 2015, \\ w & \text{otherwise.} \end{cases}$$

3 Prove that the following computational problems concerning finite automata are decidable:

(3 pts) **a.** on input a DFA D with alphabet $\{0, 1\}$, determine if D accepts some string with equally many 0s and 1s;

(3 pts) **b.** on input a DFA D and strings u, v, determine if there is a string w such that D accepts uw and rejects vw.

Solution.

a. Construct a PDA P for the language of binary strings with equally many 0s and 1s, for example by converting the context-free grammar $S \to 0S1S \mid 1S0S \mid \varepsilon$ to a PDA. Now use the cross product construction from class to obtain a PDA that recognizes the intersection of the languages recognized by P and D. Finally, convert that composite PDA to a grammar and check to see if it generates any strings, using the algorithm from class.

b. Let $D = (Q, \Sigma, \delta, q_0, F)$ be the given DFA. Run D on u and record the last state reached, say, q_u. Similarly, run D on v and record the last state reached, q_v. Now use the cross product construction from class to obtain a DFA that recognizes the intersection of the languages recognized by $(Q, \Sigma, \delta, q_u, F)$ and $(Q, \Sigma, \delta, q_v, Q \setminus F)$, and check whether that composite DFA accepts at least one string, using the reachability algorithm from class.

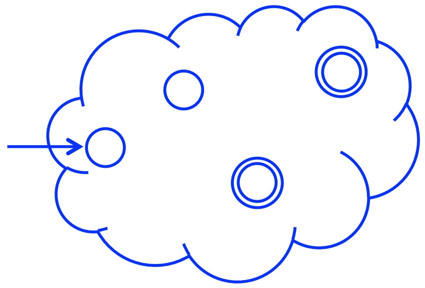
4 Prove that the following computational problems about context-free languages are decidable:

(3 pts) **a.** on input a PDA P, determine if P can ever enter an accept state with its stack empty;

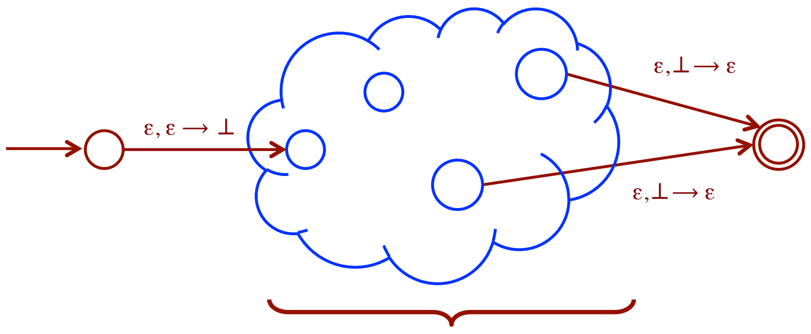
(3 pts) **b.** on input a context-free grammar G, determine if $L(G)$ is finite.

Solution.

a. Suppose that the original PDA is as shown schematically below:



The solution is to create a new PDA that accepts exactly those strings that can take the original PDA from the start state to an accept state, with the stack empty at the beginning and at the end. Then it suffices to check whether the new PDA accepts any strings, by converting it to a context-free grammar and using the algorithm from class to check whether the grammar generates any strings. The new PDA can be obtained by editing the original PDA as follows, where $\perp$ is a symbol that is not in the original stack alphabet.



make each state rejecting

b. Let $G = (V, \Sigma, R, S)$ be a given grammar. Let p be a pumping length for G, such as $p = b^{|V|+1}$ where b stands for the maximum length of the right-hand side of any rule in R. The key insight is that $L(G)$ is infinite if and only if $L(G)$ contains a string of length at least p. Indeed, the "only if" part is clear. The "if" part holds because by the pumping lemma for CFLs, any string of length at least p in $L(G)$ can be pumped, giving an infinite family of strings in $L(G)$.

With this in mind, the algorithm is simply to check whether $L(G) \cap \Sigma^p \Sigma^* \neq \varnothing$. This can be done by converting the grammar to a PDA, constructing a DFA for $\Sigma^p \Sigma^*$, combining these two automata into a PDA for $L(G) \cap \Sigma^p \Sigma^*$ via the cross product construction from class, then converting the resulting PDA to a context-free grammar, and finally using the algorithm from class to check if that grammar generates any strings.

5 For each of the following languages, determine whether it is decidable and prove your answer:

(4 pts) **a.** $L = \{\langle M \rangle$: Turing machine M accepts ε in an even number of moves$\}$

(4 pts) **b.** $L = \{\langle M, w \rangle$: on input w, Turing machine M eventually enters the reject state$\}$

Solution.

a. For the sake of contradiction, assume that L is decidable. Then the following algorithm runs in finite time and decides, on input a Turing machine M, whether M accepts ε:

> "Create a new state q_0', and let M' be the Turing machine that starts in q_0' and then immediately passes control to M. Thus, M' has one more state than M and accepts ε in an even number of moves if and only if M accepts ε in an odd number of moves. Output YES if $\langle M \rangle \in L$ (meaning M accepts ε in an even number of moves) or $\langle M' \rangle \in L$ (meaning M accepts ε in an odd number of moves). Otherwise, output NO."

This contradicts Rice's theorem, which ensures among other things that the problem of determining on input M whether M accepts ε is undecidable. Therefore, L is undecidable.

b. For the sake of contradiction, assume that L is decidable. Then the following algorithm runs in finite time and decides, on input a Turing machine M and a string w, whether M halts on w:

> "Let M' be the Turing machine obtained by interchanging the accept and reject states in M. Output YES if $\langle M, w \rangle \in L$ (meaning M enters the reject state on w) or $\langle M', w \rangle \in L$ (meaning M enters the accept state on w). Otherwise, output NO."

This contradicts the undecidability of the halting problem. Therefore, L is undecidable.