Harvard University
Computer Science 121

Quiz — October 28, 2008

You have 80 minutes to complete the quiz. Points total 80. No notes, books, or other aids are allowed. You may use any result already proved in a lecture or on a problem set.

$\Sigma = \{a, b\}$ unless otherwise specified.

PROBLEM 0 (2+2+2+2 points)

Which of the following strings are generated by the regular expression $(ab\varepsilon)^*(a \cup b \cup \emptyset)ba$? (Answer YES or NO; no explanation needed.)

(A) ε (B) aba (C) $ababba$ (D) $abababa$

(A) NO
(B) YES
(C) NO
(D) YES

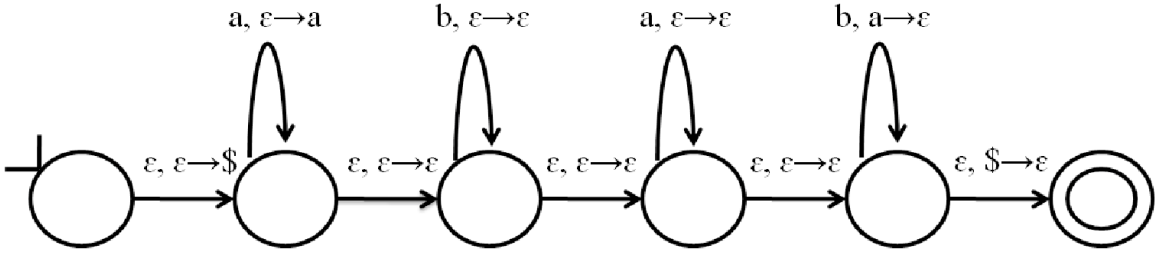
PROBLEM 1 (8 points)

Let G be the context-free grammar given by the rules

$$S \to aSb|Y$$
$$Y \to bY|Ya|\varepsilon$$

Draw the state diagram of a PDA that recognizes $L(G)$.

The given context-free grammar accepts the language $L(G) = \{a^n b^m a^k b^n | n, m, k \in \mathbb{N}\}$. This language can also be recognized by the following Pushdown Automaton:



PROBLEM 2 (4+4+4+4 points)

True or False? Write a sentence or two explaining your answer.
(A) Every language is countable.
(B) If $L_1 \cap L_2$ is regular, then L_1 and L_2 are regular.
(C) If L is non-regular, then so is the complement of L.
(D) If $M_1 = (Q_1, \Sigma, \delta_1, s_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, F_2)$ are DFAs such that $Q_1 = Q_2$, $s_1 = s_2$, and $F_1 = F_2$, then $L(M_1) = L(M_2)$.

(A) True. The strings in any language L can be enumerated by first enumerating the strings of length 0 in L (if any), then the strings in L of length 1, then the strings in L of length 2, etc. (Many people got this problem wrong. One common justification was that "we proved the existence of uncountable sets in lecture." That is true, but not all sets are languages. Languages are only sets of strings. A second common answer was that the set of all (infinite) binary sequences is an example of an uncountable language. This set is indeed uncountable, but it is not a language. Languages only consist of strings, and strings are *finite* sequences of alphabet symbols by definition.)

(B) False. Let $L_1 = \{a^n b^n : n \geq 0\}$ and $L_2 = \emptyset$. Then $L_1 \cap L_2 = \emptyset$ is regular, but L_1 is not. (Note that the converse statement is true and amounts to the closure of regular languages under intersection: if L_1 and L_2 are regular, then so is $L_1 \cap L_2$.)

(C) True. Suppose for contradiction that L is non-regular, but $\overline{L}$ is regular. By closure of regular languages under complement, the complement of $\overline{L}$ is regular. But the complement of $\overline{L}$ is simply L. $\Rightarrow\Leftarrow$

(D) False. By having different transition functions, the languages recognized can be very different (even if everything else is the same). For example, consider the (two-state) DFAs M_1, M_2 with $\Sigma = \{a\}$, $Q_1 = Q_2 = \{q, r\}$, $s_1, s_2 = q$, $F_1 = F_2 = \{r\}$, and define the two transition functions by $\delta_1(q, a) = q, \delta_1(r, a) = r$ and $\delta_2(q, a) = r$, $\delta_2(r, a) = r$. Then $L(M_1) = \emptyset$ and $L(M_2) = \{a^n : n \geq 1\}$.

(TURN OVER!)

PROBLEM 3 (7+7+7 points)

For each of the following languages, determine whether it is regular and whether it is context-free. Justify your answers.

(A) $\{a^n(bc)^n : n \geq 0\}$

(B) $\{a^{2^n} : n \geq 0\}$

(C) $\{a^n a^n a^n : n \geq 0\}$

(A) **soln1:** Recall that both regular languages and context-free languages are closed under homomorphisms. Apply the following homomorphism: $\phi(a) = a, \phi(b) = b, \phi(c) = \varepsilon$. The language becomes $\{a^n b^n : n \geq 0\}$ which we know is non-regular, so the original language is non-regular.
Since we can apply the homomorphism $\phi(a) = a, \phi(b) = bc$ to context-free language $\{a^n b^n : n \geq 0\}$ to obtain $\{a^n(bc)^n : n \geq 0\}$, the latter is also context-free.
soln2: Suppose the language were regular. The language then has some pumping length p. Consider string $w = a^p(bc)^p$. By the pumping lemma, since $|w| = 3p > p$, we can rewrite w as xyz, where $y \neq \varepsilon$ and $|xy| \leq p$, so xy must be a substring of a^p. Let $y = a^k$. Then $xz = a^{p-k}(bc)^p$. Notice that $xz \notin \{a^n(bc)^n : n \geq 0\}$ leading to a contradiction with the pumping lemma! Hence the language must be non-regular.
To show that the language is context-free, consider the following grammar that generates the language:

$$S \rightarrow aSbc|\varepsilon$$

(B) **soln1:** Every regular language is also context-free, so it suffices to show that it is not context-free to prove that it is neither.
Suppose $\{a^{2^n} : n \geq 0\}$ is context-free. It must then have some pumping length p. Consider string $w = a^{2^p}$. By the pumping lemma, since $|w| = 2^p > p$, we can rewrite w as $uvxyz$, where $vy \neq \varepsilon$ and $|vxy| \leq p$. Let $vy = a^k$. Since $|vxy| \leq p$, then $k \leq p$. By the pumping lemma, $uv^2xy^2z \in \{a^{2^n} : n \geq 0\}$. Notice that $2^p < |uv^2xy^2z| = |uvxyz| + |vy| = |w| + |vy| = 2^p + k \leq 2^p + p$. So:

$$2^p < |uv^2xy^2z| \leq 2^p + p < 2^p + 2^p = 2^{p+1}$$

So there does not exist n such that $uv^2xy^2z = a^{2^n}$, so $uv^2xy^2z \notin \{a^{2^n} : n \geq 0\}$, a contradiction with the pumping lemma! It thus follows that the language is neither context-free nor regular.
soln2: It was shown in lecture that $\{a^{2^n} : n \geq 0\}$ is not regular. From the challenge problem in Problem Set 4, every context-free language with a unary alphabet is also regular. Hence the language cannot be context-free.

(C) Notice that the language $\{a^n a^n a^n : n \geq 0\} = \{a^{3n} : n \geq 0\}$ is regular, since it can be generated by the regular expression $R = (aaa)^*$. Every regular language is also context-free, so the language is both regular and context-free.

PROBLEM 4 (9+9 points)

(A) Outline a general procedure for converting a regular expression R into a regular expression R' such that $L(R')$ is the complement of $L(R)$. Be sure to state the type (DFA, RE, etc.) of each intermediate object constructed in the process, and explain informally (in a sentence or two) how each object is obtained from the previous one.

(B) Apply your algorithm (showing all steps) to the regular expression $R = \varepsilon$.

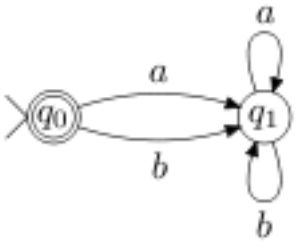
(A)

1. First convert the regular expression to an NFA using the construction in Sipser by gluing together the various smaller regular expressions using epsilon transitions to do concatenation, union, and star. (Several people suggested doing the "reverse" of the GNFA process, which was neither described in Sipser or lecture, and thus required a more detailed explanation. Several people simply said that if we have a regular expression, we have a regular language, and that we can find the DFA for such a language. We were asking for an algorithm to construct the D/NFA here.)
2. Next convert the NFA to a DFA using the subset construction.
3. Then swap the final and non-final states of the DFA to make a DFA accepting the complement of $L(R)$. Note that this only works for DFAs, so you can't do this directly on the NFA.
4. Next turn the DFA into a GNFA and perform the GNFA to regular expression construction in Sipser by "ripping" out states and fixing up the transitions.

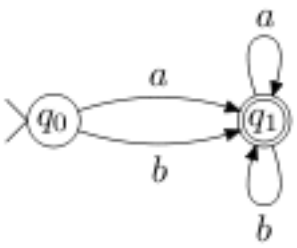
(B)

1. Create an NFA for ε.

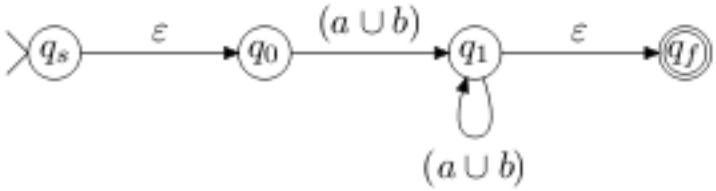
2. Turn the NFA into a DFA.



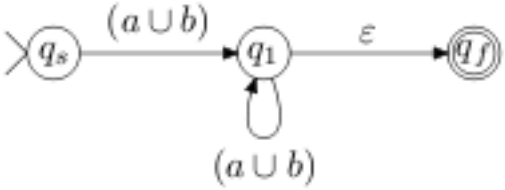
3. Swap final and non-final states.



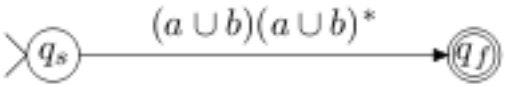
4. Turn the DFA into a GNFA.



5. Rip out state q_0.



6. Rip out state q_1.



7. Get the final regular expression of $(a \cup b)(a \cup b)^*$.

PROBLEM 5 (9 points)

For any language $L \subseteq \Sigma^*$, define $SUFFIX(L) = \{v \in \Sigma^* | uv \in L \text{ for some string } u \in \Sigma^*\}$. Show that if L is regular, then so is $SUFFIX(L)$. (Hint: construct an NFA for $SUFFIX(L)$.)

Given a DFA M for L, we convert it to an NFA for $SUFFIX(L)$ by adding a new start state, and adding ϵ transitions from that start state to every state in M that is *reachable from the original start state*. This will allow the NFA to *guess* where along a computation of the original DFA to start, and then to accept the rest of the string.

Only adding transitions to reachable states means that we won't accidentally introduce new strings that weren't suffixes of L, and jumping to all reachable states means that given any suffix s of a string w in L, we can jump to the point in the computation of M on w where it was about to start reading s, and finish the computation from there, thus recognizing the suffix.

PROBLEM 6 (Extra Credit: 2 points)

A queue is similar to a stack, except that pushing and popping happen at *opposite* ends. That is, symbols are pushed onto the top of the queue, and symbols are read and popped off the bottom of the queue. A QA is a nondeterministic automaton just like a PDA, but it has a queue instead of a stack. Find a language that is recognized by a QA but not a PDA. (There is no need to provide a formal definition of a QA.)

As we have seen, the language $L = \{w\#w : w \in \{a, b\}^*\}$ is not context-free (and hence not recognized by any PDA). L is, however, recognized by the following QA Q: as Q reads input s, it pushes the symbols of s onto the top of the queue until it encounters # (Q pushes # onto the queue as well). As Q reads the remainder of s, it pops off the the symbol on the bottom of the queue iff it matches the symbol of s currently read. Q accepts if the stack contains # after all of s has been read. It turns out that QAs are equivalent in power to Turing Machines.