Midterm Exam 2 Solutions, CS154

April 19, 2010

1. (Pumping Lemma) Please use the pumping lemma to prove that the following language is not regular: $L =$ the set of strings over $\{a, b\}$ in which, if there are $k$ consecutive $a$’s, then that block of $a$’s is followed by at least $k$ consecutive $b$’s.

For example $aabbbaaaabbbb$ is in $L$, because the first 2 $a$’s are followed by more than 2 $b$’s, and the next block of $a$’s, which is of length 4, is followed by (at least) four $b$’s; but $aaab$ is not in $L$ because the block of 3 $a$’s is not followed by enough $b$’s.

Solution. Suppose, for proof by contradiction, that $L$ is not regular. Let $N$ be as in the pumping lemma. Let $w = a^N b^N$. Then $w \in L$. According to the pumping lemma we have $w = uvz$ for some $u$, $v$, and $z$, where $|uv| \leq N$ and $|v| \geq 1$ and $uv^jz \in L$ for $j = 0, 1, 2, \ldots$. Since $|uv| \leq N$, and the first $n$ characters of $w$ are all $a$’s, we have $v = a^p$ for some $p$, and since $|v| \geq 1$ we have $p \geq 1$. Let $m$ be the length of the next block of $b$’s after $v$ in $w$ (there may be additional $a$’s after $v$ before this block of $b$’s begins; and also the case $m = 0$ when there are no more $b$’s is possible). Take $j = m + 2$. Then $v^j$ contains more $a$’s than in the next block of $b$’s, so $uv^jz$ is not in $L$. But according to the pumping lemma it is in $L$. This contradiction shows that $L$ is not regular.
2. (Grammars) Please exhibit a context-free grammar that generates the language $L$ in Problem 1.

$$
S \rightarrow \ bS \mid B \ \epsilon \\
B \rightarrow \ aTb \quad // \ B \text{ generates a string } a^n b^m \text{ with } n > 0 \text{ and } m \geq n \\
T \rightarrow \ aTb \mid Tb \mid \epsilon \quad // \ T \text{ generates a string } a^n b^m \text{ with } n \geq 0 \text{ and } m \geq n
$$

3. (Regular Grammars) Which of the following grammars are regular grammars? (Circle the letter(s) of the correct choice or choices).

(a) $T \rightarrow \ int \mid char \mid T[]$
(b) $T \rightarrow \ int \mid char \mid (*T) \mid T[]$
(c) $T \rightarrow \ int \mid char \mid \ast T \mid T[]$
(d) $T \rightarrow \ int \mid char \mid \ast T$

Please pick one of the grammars you labeled as regular (there is at least one), and exhibit an equivalent finite automaton.

**Solution.** Grammars (a) and (d) are regular. Grammar (b) isn’t regular because the rule isn’t even linear. Grammar (c) is not regular because, even though each of its rules is linear, one of them is right linear and the other is left linear.

An equivalent FA for grammar (c) needs a start state $T$, with a loop on $T$ labeled $\ast$, and an accepting state $F$, and (if $char$ is regarded as four symbols rather than one), chains of states with arrows to accept $char$ and $int$ leading from $T$ to $F$.

An equivalent grammar for (a) is trickier since the algorithm works for rules with terminals on the other side of the nonterminal. The algorithm calls for first reversing all the rules, converting to an FA, then reversing all the arrows. The result has a start state $S$, chains of states to read $int$ and $char$ that lead to state $T$, another state $R$, and transitions $\delta(T,[]) = R$ and $\delta(R,]) = T$. The only accepting state is $T$.

On your exam you could draw pictures of your FA, but on the solution sheet I am just giving these verbal descriptions, which should be enough for you to tell if you got the problem right.
4. (PDA) Please exhibit a PDA that accepts the language in Problem 1. (I recommend that you do this directly rather than basing your work on your answer to problem 2, in case you might have made a mistake in problem 2. This is a separate problem and does not depend on problems 1 or 2, it just mentions the same language.)

Solution. First I give it in English. We need two states, state $B$ and state $A$. In state $A$, while reading $a$, push $a$. While reading $b$ in state $A$, change to state $B$, pop $a$ if possible, or if the stack is empty, just read the $b$. In state $B$ reading $b$, pop $a$ if there is $a$ at the top of the stack, or just read the $b$ if the stack is empty. In state $B$ reading $a$ with an empty stack, push $a$ and change to state $A$. There is no rule for reading $a$ in state $B$ with a nonempty stack—that is when the string is rejected.

To convert this idea into a PDA, we need to be able to test for “stack empty”. You can’t officially do that, so we will first push a special “stack-top” symbol $Z$ on the stack, and then instead of testing for empty stack, we will test for $Z$ on the stack. We only want to do that once, so we make the start state $S$ different from $A$ and $B$.

$$S, \epsilon, \epsilon \rightarrow A, Z // push Z$$

$$A, a, \epsilon \rightarrow A, a // push a$$

$$A, b, a \rightarrow B, \epsilon // pop a and switch states$$

$$B, b, a \rightarrow B, \epsilon // pop a if possible$$

$$B, b, Z \rightarrow B, Z // just read the B$$

$$B, a, Z \rightarrow A, aZ // push the a$$

$$B, \epsilon, Z \rightarrow B, \epsilon // empty the stack at the end$$

5. (Grammar to PDA) Please use the algorithm you practiced in homework to convert the following grammar to an equivalent PDA. The start symbol of the grammar is $S$.

$$S \rightarrow aTb \mid T$$

$$T \rightarrow bSa \mid S \mid \epsilon$$

Solution. The PDA has two states, $P$ and $F$. The start state is $P$ and the only accepting state is $F$. The rules are

$$P, \epsilon, \epsilon \rightarrow F, S$$

$$F, a, S \rightarrow F, Tb$$

$$F, \epsilon, S \rightarrow F, T$$

$$F, b, T \rightarrow F, Sa$$

$$F, \epsilon, T \rightarrow S$$

$$F, \epsilon, T \rightarrow \epsilon$$
6. (Closure Properties) Please use the closure properties of regular languages (not the pumping lemma) to prove that the language generated by the grammar in Problem 5 is not regular.

Solution. The intersection of $L$ with the regular language $L(a^*b^*)$ is the non-regular language $M = \{a^n b^n : n \geq 0\}$, since the rule $T \rightarrow bSa$ can never be used in generating a string with all the $a$'s preceding all the $b$'s, so whenever $T$ occurs in a generated string, it has to be immediately either erased or replaced with $S$; and the rule $S \rightarrow aTb$ always generates equal numbers of $a$'s and $b$'s. Now assume, for proof by contradiction, that $L$ is regular. Then, by the closure of regular languages under intersection, $M$ is regular; but $M$ is known not to be regular. This contradiction proves that $L$ is not regular.