CS 448/2405
Automata and Formal Languages
ASSIGNMENT # 2
DUE DATE: Friday, October 31

Note: You must turn in the assignment by October 31, but you can turn in the extra credit problem anytime until Nov 11.

1. Sipser, Exercise 2.6.

   (a) The set of strings over a, b with twice as many a's as b's.

   (b) The complement of the language $\{a^n b^n \mid n \geq 0\}$.

$$S \to DbaD \mid AC \mid CB$$

$$D \to \epsilon \mid aD \mid bD$$
$$A \to aA \mid a$$
$$B \to bB \mid b$$
$$C \to \epsilon \mid aCb$$

   (c) C is strings of the form $w\#x$ such that w^R is a substring of x.

$$S \to CD$$

$$C \to 0C0 \mid 1C1 \mid A$$
$$A \to \#D$$
$$D \to 0D \mid 1D \mid \epsilon$$

   (d) D is strings of the form $x_1\#x_2\#\ldots\#x_k$ such that $k \geq 1$ and $x_i = x_j^R$ for some i and j.

$$S \to LCR$$
$$C \to aCA \mid bCB \mid \#L \mid \epsilon$$
$$L \to A\#L \mid \epsilon$$
$$R \to \#AR \mid \epsilon$$
$$A \to 0A \mid 1A \mid \epsilon$$

2. Give informal English descriptions of PDA's for the first two languages from Exercise 2.6.

(a) First push the \$ onto the stack so that we know when we are at the empty stack. Now if the stack is empty or there is an 'a' at the top of the stack, and we read an 'a', then push an 'a' onto the stack for each a that is read on the input string. If the stack is empty, or there is a 'b' at the stop of the stack and we read a 'b', then push two 'b's onto the stack for each 'b' that is read. If there are two 'a''s at the top of the stack, and we read a 'b', then pop the two 'a''s as we read the 'b'. Otherwise if there is only one 'a' on the top of the stack, pop that 'a' and push one 'b' onto the stack. If there is a 'b' at the top of the stack and we read an 'a', the pop the 'b' for each 'a' that is read. Finally, if the top of the stack is the dollar sign, then nondeterminstically branch (without reading any new input) to the accept state.

(b)

3. Sipser, Exercise 2.14.

(i) First add a new start symbol S_0 and the transition $S_0 \to A$. This gives:

$$S_0 \to A$$

$$A \to BAB \mid B \mid \epsilon$$

$$B \to 00 \mid \epsilon$$

(ii) Remove ϵ rules. This gives:

$$S_0 \to A$$

$$A \to BAB \mid AB \mid BA \mid B \mid A \mid BB$$

$$B \to 00$$

(iii) Remove unit rules. This gives:

$$S_0 \to BAB \mid BB \mid AB \mid BA \mid 00 \mid \epsilon$$

$$A \to BAB \mid BB \mid AB \mid BA \mid 00$$

$$B \to 00$$

(iv) FInally convert all remaining rules to the proper form. This gives:

$$S_0 \to BA_1 \mid BB \mid AB \mid BA \mid ZZ \mid \epsilon$$

$$A \to BA_1 \mid BB \mid AB \mid BA \mid ZZ$$

$$B \to ZZ$$

$$A_1 \to AB$$

$$Z \to 0$$

4. Sipser, Problem 2.18.

(a) $L = \{0^n 1^n 0^n 1^n \mid n \geq 0\}$. Let p be as in the pumping lemma, and consider the string $s = 0^p 1^p 0^p 1^p$. By the pumping lemma, s can be written as $uvxyz$, where $|vxy| \leq p$, and $|vy| > 0$. There are four cases to consider. (i) If $vxy = 0^i$ for some $i \leq p$, then the string uv^2xy^2z is not of the proper form, since the two strings of 0's are not of the same size. (ii) Likewise, if $vxy = 1^i$ for some $i \leq p$, then again uv^2xy^2z is not of the proper form since the two strings of 1's are not of the same size. (iii) $vxy = 0^i 1^j$ or $vxy = 1^i 0^j$ where $i + j \leq p$, $i + j > 0$. The string uv^2xy^2z is not of the proper form. This string will have either an unequal number of 0's in the two different zero-strings, or it will have an unequal number of 1's in the two different one-strings. There are no more cases to consider since $|vxy| \leq p$.

(b) $L = \{0^n \# 0^{2n} \# 0^{3n} \mid n \geq 0\}$. Let p be as in the pumping lemma, and consider the string $s = 0^p \# 0^{2p} \# 2^{3p}$. We will write $s = a\#b\#c$ where a, b, c are the 3 different sequences of zeroes. Again there are several cases. (i) if vxy is a substring of a, then the string uv^0xy^0z will not be in L; (ii) if vxy is a substring of b, then the string uv^2xy^2z will not be in L since the new middle sequence of zeroes, b' will have too many zeroes. (iii) if vxy is a substring of c, then again the string uv^2xy^2z will not be in L. (iv) vxy is a substring of $a\#b$. Then the string uv^0xy^0z is not of the proper form. (v) vxy is a substring of of $b\#c$. Then similar to the reasoning in (iv), the string uv^0xy^0z is not in the proper form.

(c) $L = \{w\#x \mid w \text{ is a substring of } x\}$. Here, both w and x are over the language $\{a, b\}$. Let p be as in the pumping lemma, and consider $s = 0^p 1^p \# 0^p 1^p$. The argument is similar to (a). (i) If v or y contains the pound symbol, then we pump up, and the string cannot be in the proper form since it will contain more than one pound symbol. (ii) Let $x = 0^{p-i}$, $vxy = 0^i 1^j$, $z = 1^{p-j} \# 0^p 1^p$. Then pumping s upwards ($i = 2$) will yields a string that is not of the proper form since the length of the string to the left of the pound will be greater than the string to the right of the pound. (iii) Let $x = 0^p 1^p \# 0^{p-i}$, $vxy = 0^i 1^j$, $z = 1^{p-j}$. By the same reasoning as (ii), the string obtained by pumping s upwards ($i = 2$) is not of the proper form. (iv) $vxy = 1^i \# 0^j$, where x contains the pound symbol. Pumping s upwards ($i = 2$) will yield a string s' that is not of the proper form since there won't be the same number of 0's and 1's on the left as on the right.

(d) $L = \{x_1 \# x_2 \# \ldots \# x_k \mid k \geq 2, \text{ and} x_i = x_j \text{ for some } i \neq j\}$. Let $s = 0^p 1^p \# 0^p 1^p$. This is very similar to case (c) above.

5. Sipser, Problem 2.26. We want to show that the language $L = \{x\#y \mid x \neq y\}$ is a CFL. (Here I will only give a high level sketch of the idea.) To do this, we will construct a PDA, P accepting exactly L. P will guess a position in which the string x differs from y, and verify this guess. In order to do this, as it reads the string $w = x\#y$, it will push a symbol onto the stack for each symbol of x that is processed. At some point i after processing $x' = x_1 \ldots x_i$ (the first i

symbols of x), P will go into a state which will remember x_i, and then advance the input head forward until it reaches the # sign. During this second phase of the computation (as the input head scans from x_{i+1} to #, it will no longer push symbols onto the stack. After reading the #, P will then process $y' = y_1, \ldots y_i$, the first i symbols if y, popping a symbol off of the stack for each symbol of y' read. Upon reaching y_i, P will check whether $y_i = x_i$. (This is possible since we remembered the value of x_i by the finite state control.) As long as they are not equal, P accepts.

6. Construct a PDA equivalent to the following grammar.

$$S \to aAA$$

$$A \to aS \mid bS \mid a$$

7. (Extra Credit) Prove that the set of primes in binary is not a CFL.