

## Homework 2

Due: Oct 2nd, 2008

YanWei Wu

**Problem 1:** Sipser problem 1.45 (page 90). Let  $A/B = \{w \mid wx \in A \text{ for some } x \in B\}$ . Show that if  $A$  is regular and  $B$  is regular, then  $A/B$  is regular.

**Proof:** We can prove by construction.

Construct a DFA  $D_A$  for regular language  $A$  and a DFA  $D_B$  for regular language  $B$ . For each state  $q$  in  $D_A$ , if the string produced from this state  $q$  to any final state  $q_f$  in  $D_A$  is accepted by DFA  $D_B$ . We simply put  $q$  to final state set  $F$ . Thus we construct the DFA  $D_{A/B}$ ,<sup>1</sup> which is same to  $D_A$  except that the final states are all states in set  $F$ .

We can conclude that  $A/B$  is regular because there is a DFA  $D_{A/B}$  recognizing it.

This finishes the proof.

■

**Problem 2:** Sipser exercise 2.4 and 2.6 (page 128). Given context free grammars that generate the following languages. In all parts, the alphabet  $\Sigma$  is  $\{0, 1\}$ .

- $\{w \mid w \text{ starts and ends with the same symbol}\}$

**Solution 2:**[1]  $S \rightarrow 1A1 \mid 0A0$   
 $A \rightarrow 0A \mid 1A \mid \varepsilon$

- $\{w \mid \text{the length of } w \text{ is odd}\}$

**Solution 2:**[2]  $S \rightarrow 0S0 \mid 1S1 \mid 1S0 \mid 0S1 \mid 0 \mid 1$

- $\{w \mid w = w^R\}$ . Here  $w^R$  is the reverse of  $w$ .

**Solution 2:**[3]  $S \rightarrow 0S0 \mid 1S1 \mid 1 \mid 0 \mid \varepsilon$

- The complement of the language  $\{0^n 1^n \mid n \geq 0\}$ , i.e.,  $\{w \mid w \text{ not in format } 0^n 1^n\}$ .

**Solution 2:**[4] There are only three cases:

- Starting with 1
- Starting with 0, but it has "10" as its substring
- It is in the format of  $0^m 1^n$ , but  $m \neq n$ ;

So based on the cases above, we can get the context free grammar of the aiming language as following:  $S = 1D|0A|B$

$A = D10D$

$B = EC|CF$

$C = \epsilon|0C1$  ( $C$  denotes  $0^n 1^n$ )

$D = \epsilon|0D|1D$  ( $D$  denotes  $\Sigma^*$ )

$E = 0|0E$  ( $E$  denotes  $0^+$ )

$F = 1|1F$  ( $F$  denotes  $1^+$ )

<sup>1</sup>For more completed proof, we should also prove this DFA really recognize  $A/B$ .

5.  $\{w \mid w \text{ has equal number of 0s and 1s}\}$ .

**Solution 2:**[5]  $S \rightarrow 0S1 \mid 1S0 \mid SS \mid \varepsilon$

6.  $\{0^m 1^n \mid m \neq n, \text{ and } 2m \neq n\}$

**Solution 2:**[6] There are also three cases:

- $m > n$   
 $A = 0A_1$   
 $A_1 = \epsilon|0A_1|0A_11;$
- $2m < n$   
 $B = B_11$   
 $B_1 = \epsilon|B_11|0B_111;$
- $n/2 < m < n$   
 $C = 0C_11$   
 $C_1 = 0C_11|0C_211 \quad C_2 = \epsilon|0C_211.$

So  $S = A|B|C$ .

**Problem 3:** Sipser problem 2.20 (page 130). Let  $A/B = \{w \mid wx \in A \text{ for some } x \in B\}$ . Show that if  $A$  is context-free and  $B$  is regular language, then  $A/B$  is context-free.

**Proof:** We prove by construction.

Construct PDA  $P_A$  for language  $A$  and DFA  $D_B$  for language  $B$ . For each state  $q$  in  $P_A$ , check (1) if the stack is empty; (2) if the string produced from this state  $q$  to any final state  $q_f$  in  $P_A$  is accepted by DFA  $D_B$ . If we get both positive answers, put state  $q$  to the final state set  $F$ .

Then the new PDA is  $P_A$  but the final states are the stated in set  $F$ , which can recognize language  $A/B$ .<sup>2</sup>

Because we can construct PDA for language  $A/B$ ,  $A/B$  is context free language.

This finished the proof.

■

**Problem 4:** Given pushdown automatas that generate the following languages. You have to explicitly explain the meanings of the states used in your automata.

1.  $\{w \mid w \text{ has equal number of 0s and 1s}\}$ . Here the alphabet  $\Sigma = \{0, 1\}$ .

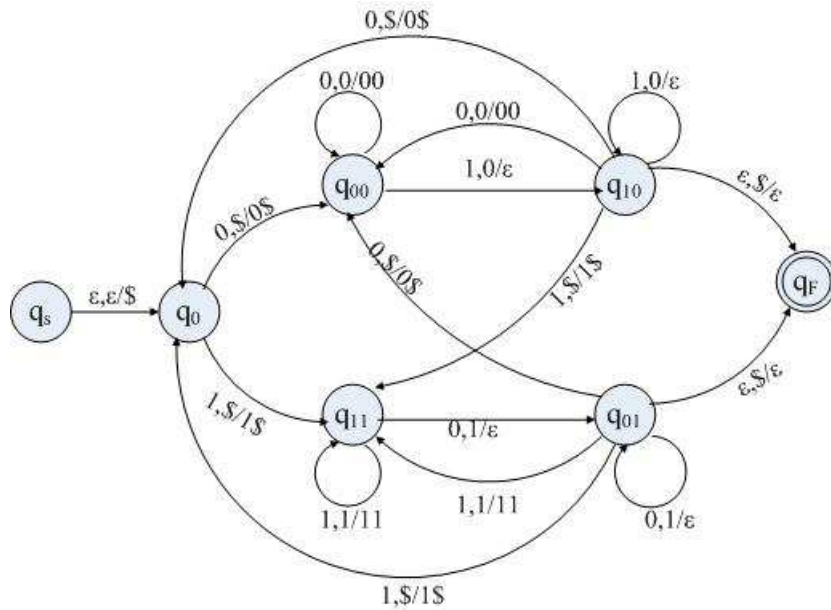
**Solution 4:**[1]

There are 2 cases:

1. If we first read '0's, we should push '0' into the stack, go to state  $q_00$ , and then if we start to read '1's, pop '0's out of the stack, go to state  $q_10$ , at this state, we have 3 cases: (1) then if we read '0', go back to state  $q_00$ ; or if read '1', the stack is empty, we should push this '1' into the stack, and go to state  $q_11$ ; (3) or if the string is empty, and stack is empty, we accept this input, and go to the final state.
2. If we first read '1's, we should push '1' into the stack, go to state  $q_11$ , and then if we start to read '0's, pop '1's out of the stack, go to state  $q_01$ , at this state, we have 3 cases: (1) then if we read '1', go back to state  $q_11$ ; or if read '0', the stack is empty, we should push this '0' into the stack, and go to state  $q_00$ ; (3) or if the string is empty, and stack is empty, we accept this input, and go to the final state.

---

<sup>2</sup>For more completed proof, we should also prove this PDA really recognize  $A/B$ .



$q_s$ :  
 $\delta(q_s, 0, \$) \rightarrow (q_{00}, 0\$)$  if input is 0, the top of stack is \$, then push 0 into stack, and go to  $q_{00}$   
 $\delta(q_s, 1, \$) \rightarrow (q_{11}, 1\$)$  if input is 1, the top of stack is \$, then push 1 into stack, and go to  $q_{11}$   
 $q_{00}$ :  
 $\delta(q_{00}, 0, 0) \rightarrow (q_{00}, 00)$  if input is 0, the top of stack is 0, then push 0 into stack, and stay at  $q_{00}$   
 $\delta(q_{00}, 1, 0) \rightarrow (q_{10}, \epsilon)$  if input is 1, the top of stack is 0, then pop 0 out of stack, and go to  $q_{10}$   
 $q_{11}$ :  
 $\delta(q_{11}, 1, 1) \rightarrow (q_{11}, 11)$  if input is 1, the top of stack is 1, then push 1 into stack, and stay at  $q_{11}$   
 $\delta(q_{11}, 0, 1) \rightarrow (q_{01}, \epsilon)$  if input is 0, the top of stack is 1, then pop 1 out of stack, and go to  $q_{01}$   
 $q_{10}$ :  
 $\delta(q_{10}, 1, 0) \rightarrow (q_{10}, \epsilon)$  if input is 1, the top of stack is 0, then pop 0 out of stack, and stay at  $q_{10}$   
 $\delta(q_{10}, 0, 0) \rightarrow (q_{00}, 00)$  if input is 0, the top of stack is 0, then push 0 into stack, and go to  $q_{00}$   
 $\delta(q_{10}, 1, \$) \rightarrow (q_{11}, 1\$)$  if input is 1, the top of stack is \$, then push 1 into stack, and go to  $q_{11}$   
 $\delta(q_{10}, \epsilon, \$) \rightarrow \delta(q_F, \epsilon)$  if input is  $\epsilon$ , the top of stack is \$, then go to the final state  $q_F$   
 $q_{01}$ :  
 $\delta(q_{01}, 0, 1) \rightarrow (q_{01}, \epsilon)$  if input is 0, the top of stack is 1, then pop 1 out of stack, and stay at  $q_{01}$   
 $\delta(q_{01}, 1, 1) \rightarrow (q_{11}, 11)$  if input is 1, the top of stack is 1, then push 1 into stack, and go to  $q_{11}$   
 $\delta(q_{01}, 0, \$) \rightarrow (q_{00}, 0\$)$  if input is 0, the top of stack is \$, then push 0 into stack, and go to  $q_{00}$   
 $\delta(q_{01}, \epsilon, \$) \rightarrow (q_F, \epsilon)$  if input is  $\epsilon$ , the top of stack is \$, then go to the final state  $q_F$

2.  $\{a^i b^j c^k \mid j = i, \text{ or } j = k, i, j, k \geq 0\}$ . Here the alphabet  $\Sigma = \{a, b, c\}$ .

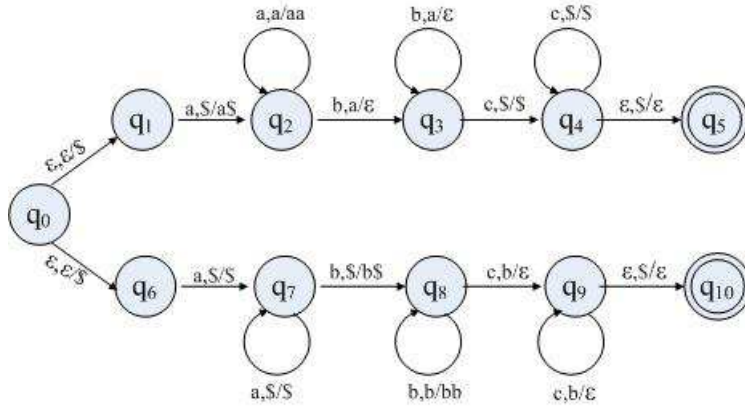
**Solution 4:**[2]

There are 2 cases:

(1)  $a^i b^i c^k$ : when read 'a', we should push 'a' into the stack, after we have read all 'a's, when we start to read 'b', pop the 'a's out of the stack, after we have read all 'b's, the stack should

be empty, there are equal number of 'a's and 'b's. when we read 'c', we do not have to do any operation to the stack, just omit the 'c's, then if we read  $\epsilon$ , and the stack is empty, we go to the final state.

(2)  $a^i b^k c^k$ : when read 'a', we should omit this input and do not have to do any operation on the stack, then we start to read b, push 'b' into the stack, after we finished to read 'b', when start to read 'c', pop 'b's out of the stack. If we finished reading the string, and stack is empty, we can conclude there are equal number of 'b's and 'c's, and go to the final state.



if  $j=i$ ,  $L=a^i b^i c^k$

start at  $q_0$ , automatically go to  $q_1$ , and put  $\$$  into stack.

$q_1$ :

$\delta(q_1, a, \$) \rightarrow (q_2, a\$)$

if input is a, the top of stack is  $\$$ , then push a into stack, and go to  $q_2$

$q_2$ :

$\delta(q_2, a, a) \rightarrow (q_2, aa)$  if input is a, the top of stack is a, then push a into stack, and stay at  $q_2$

$\delta(q_2, b, a) \rightarrow (q_3, \epsilon)$  if input is b, the top of stack is a, then pop a out of stack, and go to  $q_3$

$q_3$ :

$\delta(q_3, b, a) \rightarrow (q_3, \epsilon)$  if input is b, the top of stack is a, then pop a out of stack, and stay at  $q_3$

$\delta(q_3, c, \$) \rightarrow (q_4, \$)$  if input is c, the top of stack is  $\$$ , then omit this input, and go to  $q_4$

$q_4$ :

$\delta(q_4, c, \$) \rightarrow (q_4, \$)$  if input is c, the top of stack is  $\$$ , then omit this input, and stay at  $q_4$

$\delta(q_4, \epsilon, \$) \rightarrow \delta(q_5, \epsilon)$  if input is  $\epsilon$ , the top of stack is  $\$$ , then go to final state  $q_5$

if  $j=k$ ,  $L=a^i b^k c^k$

start at  $q_0$ , automatically go to  $q_6$ , and put  $\$$  into stack.

$q_6$ :

$\delta(q_6, a, \$) \rightarrow (q_7, \$)$  if input is a, the top of stack is  $\$$ , then omit this input, and then go to  $q_7$

$q_7$ :

$\delta(q_7, a, \$) \rightarrow (q_7, \$)$  if input is a, the top of stack is  $\$$ , then omit this input, and stay at  $q_7$

$\delta(q_7, b, \$) \rightarrow (q_8, b\$)$  if input is b, the top of stack is  $\$$ , then push b into stack, and go to  $q_8$

$q_8$ :

$\delta(q_8, b, b) \rightarrow (q_8, bb)$  if input is b, the top of stack is b, then push b into stack, and stay at  $q_8$

$\delta(q_8, c, b) \rightarrow (q_9, \epsilon)$  if input is c, the top of stack is b, then pop b out of stack, and go to  $q_9$

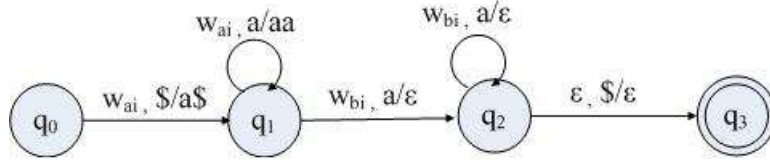
$q_9$ :

$\delta(q_9, c, b) \rightarrow (q_9, \epsilon)$  if input is c, the top of stack is b, then pop b out of stack, and stay at  $q_9$

$\delta(q_9, \epsilon, \$) \rightarrow (q_{10}, \epsilon)$  if input is  $\epsilon$ , the top of stack is  $\$$ , then go to final state  $q_{10}$

**Problem 5:** Assume that you are given the DFA automata  $M_A = (Q_A, \Sigma_A, q_a, F_A)$  for a regular language  $A$  and the DFA automata  $M_B = (Q_B, \Sigma_B, q_b, F_B)$  for a regular language  $B$ . Construct a pushdown automata for the following language using  $M_A$  and  $M_B$ .  $A \diamond B = \{xy \mid x \in A \text{ and } y \in B \text{ and } |x| = |y|\}$ . Here  $|x|$  is the length of the string  $x$ .

**Proof:** We can construct a pushdown automata which first recognize  $w \in A$ , and push 'a' into the stack, and then recognize  $w \in B$ , and pop 'a' out of the stack. If after reading all strings, the stack is empty, the PDA can pop all the strings that had been read before, this means the length of  $x$  is equal to the length of  $y$ , so it can accept  $A \diamond B = \{xy \mid x \in A \text{ and } y \in B \text{ and } |x| = |y|\}$ .



■

**Problem 6:** Prove that the following languages are *not context-free*.

1.  $L_1 = \{w \mid N(w, a) = N(w, b) = N(w, c)\}$ . Here the alphabet  $\Sigma = \{a, b, c\}$  and  $N(w, a)$  denotes the number of  $a$ 's in the string  $w$ .

**Proof:** We prove by contradiction.

$\forall N, \exists w = a^n b^n c^n$ , assume it is context-free, we can write  $w = uvxyz$

(1) if  $vxy \in a^*$ , let  $|vy| = t$  for  $v^i xy^i = a^{n+(i-1)t} b^n c^n$ ,

when  $i \geq 2$ , there would be more 'a' than 'b' nor 'c'.

(2) if  $vxy \in b^*$ , let  $|vy| = t$  for  $v^i xy^i = a^n b^{n+(i-1)t} c^n$ ,

when  $i \geq 2$ , there would be more 'b' than 'a' nor 'c'.

(3) if  $vxy \in c^*$ , let  $|vy| = t$  for  $v^i xy^i = a^n b^n c^{n+(i-1)t}$ ,

when  $i \geq 2$ , there would be more 'c' than 'a' nor 'b'.

(4) assume  $w = uv^i xy^i z = a^n b^n c^n$ , let  $v \in a^i$ , and  $y \in b^j, |vy| > 0, i + j \leq n$   
for  $w = uv^{i+t} xy^{i+t} z = a^{n+t} b^{n+t} c^n$ , there would be more 'a' and 'b' than 'c'.

(5) assume  $w = uv^i xy^i z = a^n b^n c^n$ , let  $v \in b^i$ , and  $y \in c^j, |vy| > 0, i + j \leq n$   
for  $w = uv^{i+t} xy^{i+t} z = a^n b^{n+t} c^{n+t}$ , there would be more 'b' and 'c' than 'a'.

Therefore,  $\exists w = uv^i xy^i z \notin L_1$ , so this language is not context-free. ■

2.  $L_2 = \{w \mid N(w, a) \text{ is a prime number}\}$ . Here the alphabet  $\Sigma = \{a, b\}$ .

**Proof:** We prove by contradiction.

$\forall N, \exists w = a^p$ , assume it is context-free, we can write  $w = uvxyz$

(Let  $|vy| = t > 0, uv^i xy^i z = a^{p+(i-1)t}$ ,

$\exists i = p + 1, a^{p+(i-1)t} = a^{p \times (1+t)}$ , obviously, this is not a prime number.

Therefore,  $\exists w = uv^i xy^i z \notin L_2$ , so this language is not context-free.

■