COMP 335: Assignement #3 Solution

Winter, 2013

Tuesdays & Thursdays 13:15-14:05

Instructor: Professor Grahne

Tutor: F. Boroomand

Problem # 1

The DFA describing $L\big((ab+ba)^*\big)$ is:

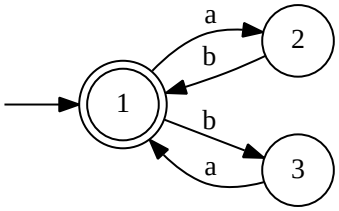


Figure 1: G_1

$L\big((abb+aab+baa+bba)^*\big)$ can be represented with the following DFA:

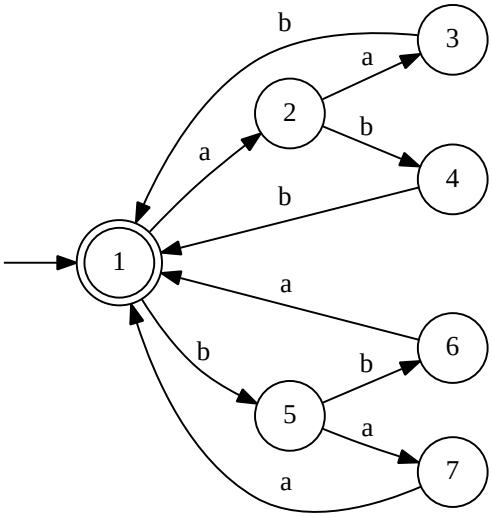


Figure 2: G_2

Finding the product of the two languages gives us the following DFA which describes the intersection of two languages.

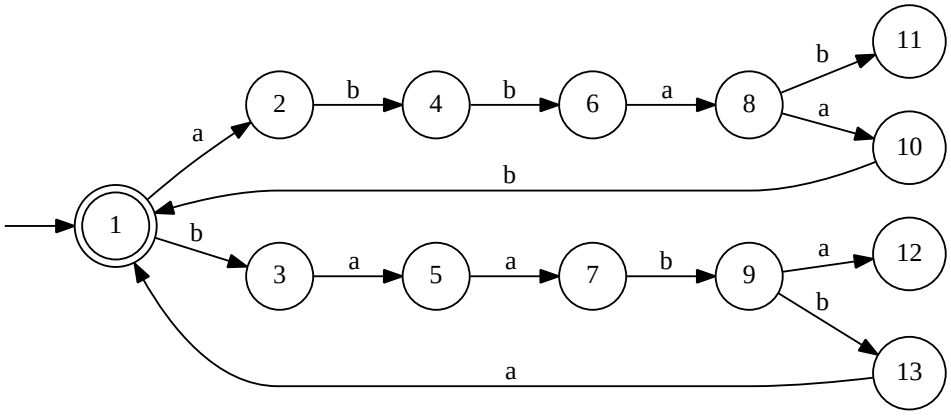


Figure 3: G=product(G_1, G_2)

The following table shows the state equivalencies.

state of G	$(p,q) : p \in Q_{G_1}, q \in Q_{G_2}$
1	(1,1)
2	(2,2)
3	(3,5)
4	(1,4)
5	(1,7)
6	(3,1)
7	(2,1)
8	(1,2)
9	(1,5)
10	(2,3)
11	(3,4)
12	(2,7)
13	(3,6)

Problem # 2

If L is regular, it is accepted by some DFA, say $A = (Q, \Sigma, \delta, s_0, F)$. We will construct a ϵ-NFA B, such that $L(B) = third(L(A))$. Here you need four copies of A. Formally,

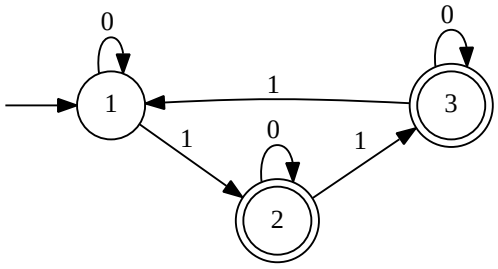
$$B = (Q \times \{1,2,3,4\}, \Sigma, \rho, F \times \{4\}),$$

where

$$\begin{aligned} \rho \quad = \quad & \{(\langle p,1\rangle, \epsilon, \langle q,2\rangle) : (p,a,q) \in \delta, \text{ for some } a \in \Sigma\} \cup \\ & \{(\langle p,2\rangle, \epsilon, \langle q,3\rangle) : (p,a,q) \in \delta, \text{ for some } a \in \Sigma\} \cup \\ & \{(\langle p,3\rangle, a, \langle q,4\rangle) : (p,a,q) \in \delta\} \cup \\ & \{(\langle p,4\rangle, \epsilon, \langle p,1\rangle) : p \in Q\}. \end{aligned}$$

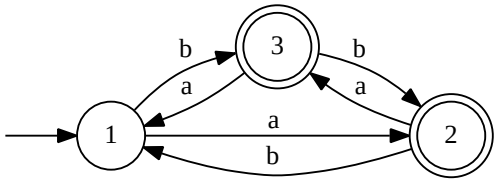
Problem # 3

The following DFA describes the Language L.



The DFA for $h^{-1}(L)$ is followed.

B	×							
C	×	×						
D	◯	×	×					
E	×	◯	×	×				
F	×	×	◯	×	×			
G	◯	×	×	◯	×	×		
H	×	◯	×	×	◯	×	×	
I	×	×	◯	×	×	◯	×	×
	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*



Problem # 4

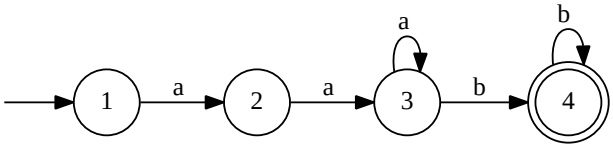
The following table shows the state equivalencies where × shows a pair of distinguishable states and ◯ shows equivalency.

Thus, the states $\{A, D, G\}$, $\{B, E, H\}$ and $\{C, F, I\}$ are equivalent. The minimized DFA is shown below.

DFA_{min}	0	1
$\rightarrow ADG$	BEH	BEH
BEH	CFI	CFI
$*CFI$	ADG	BEH

Problem # 5

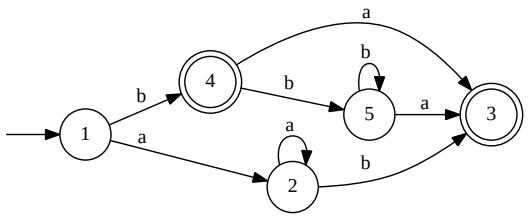
a)



Applying TF algorithm we see that all the states are distinguishable. This means that the DFA is minimal.

2	×		
3	×	×	
4	×	×	×
	1	2	3

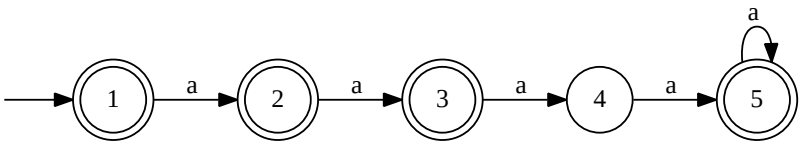
b)



Applying TF algorithm we see that all the states are distinguishable. This means that the DFA is minimal.

2	×			
3	×	×		
4	×	×	×	
5	×	×	×	×
	1	2	3	4

c)



Applying TF algorithm we see that all the states are distinguishable. This means that the DFA is minimal.

2	×			
3	×	×		
4	×	×	×	
5	×	×	×	×
	1	2	3	4

Problem # 6

a) The designed grammar generates at most two b's for generating one a.
$S \rightarrow aSb \mid aSbb \mid \epsilon$

b) Consider the language in two different cases:

- $L_1 = \{a^i b^j a^k : i = j, k \geq 0\}$:

$$\begin{aligned} S_1 &\rightarrow aS_1bA \mid \epsilon \\ A &\rightarrow Aa \mid \epsilon \end{aligned}$$

- $L_2 = \{a^i b^j a^k : i \geq 0, j > k\}$:

$$\begin{aligned} S_2 &\rightarrow AB \\ A &\rightarrow aA \mid \epsilon \\ B &\rightarrow bBa \mid bB \mid \epsilon \end{aligned}$$

Finally, the following production is required to produce the language L:

$$S \rightarrow S_1 \mid S_2$$

Problem # 7

To prove that a language L is the language generated by the grammar G, i.e. L = L(G) you have to prove two things:

- Every string w generated by G is in L. Typically, this is a proof by induction on the number of steps in the derivation of w $\big(L(G) \subseteq L\big)$.
- Every string w in L can be generated by G. Typically, this is a proof by induction on the length, or some other parameter, of w. $\big(L \subseteq L(G)\big)$

We only prove the first inclusion $(L(G) \subseteq L)$ here. The other inclusion can be proved with the similar approach. We prove it by induction on the number steps required for derivation of a string.

a) The second production rule generates $L_1 = \{w : w = w^R, w \in \{a,b\}^*\}$. We claim that the language generated by the CFG is $L = \{a^n w b^n : w = w^R \wedge n \geq 1\} \cup \{b^n w a^n : w = w^R \wedge n \geq 1\}$.

- Base: The minimum number of steps for derivation of a string is 2. Derivation with two steps gives us the strings $ab, ba \in L$, so the base step holds.
- Hypothesis: assume that for every derivation $S \Rightarrow^* w$ with $n \geq 2$ steps, w is in L.
- Induction step: prove that every derivation with $n+1$ steps generates a string in L. Let $S \Rightarrow^* w$ be a derivation with $n+1$ steps. The first derivation can be either $S \rightarrow aAb$ or $S \rightarrow bAa$. In the firs case, the string s will be in form of as_1b where s_1 is generated by the production rules from variable A in n steps. Thus, $s_1 = s_1^R$ and $as_1b \in L$. We have the same story for the other case.

b) We claim that the language generated by the CFG is all the string ending with a $L = \{wa : w \in \{a,b\}^*\}$.

- Base: Derivation with one step gives us the strings $a \in L$, so the base step holds.
- Hypothesis: assume that for every derivation $S \Rightarrow^* w$ with $n \geq 1$ steps, w is in L.
- Induction step: prove that every derivation with $n+1$ steps generates a string in L. Let $S \Rightarrow^* w$ be a derivation with $n+1$ steps. The first derivation can be either $S \rightarrow aS$ or $S \rightarrow bS$. In the firs case, the string s will be in form of as_1 where s_1 is generated in n steps . Consequently, $s_1 \in L$ and it ends with a, so as_1 also ends with a and $as_1 \in L$. We have the same story for the other case.

c) The language generated by the CFG is similar to the part (b). We claim that the language is $L = \{w : w \in (b^*a)^+\}$.

- Base: Derivation with one step gives us the strings $a \in L$, so the base step holds.

- Hypothesis: assume that for every derivation $S \Rightarrow^* w$ with $n \geq 1$ steps, w is in L.
- Induction step: prove that every derivation with $n+1$ steps generates a string in L. Let $S \Rightarrow^* w$ be a derivation with $n+1$ steps. The first derivation can be either $S \to SS$ or $S \to bS$. In the firs case, the string s will be in form of $s_1 s_2$ where s_1, s_2 are generated in n steps . Consequently, $s_1, s_2 \in L$ and their concatenation also belongs to L. We have the same story for the other case.

d) The language generated by the CFG is $L = \{w : w \in \{a, b\}^*\}$

- Base: Derivation with one step is $S \to \epsilon$ which gives us the strings $\epsilon \in L$, so the base step holds.
- Hypothesis: assume that for every derivation $S \Rightarrow^* w$ with $n \geq 1$ steps, w is in L.
- Induction step: prove that every derivation with $n+1$ steps generates a string in L. Let $S \Rightarrow^* w$ be a derivation with $n+1$ steps. The first derivation can be1: $S \to s_1 a s_2 | b s_1 | a s_1 | b s_1$, where s_1, s_2 are generated in n steps . Consequently, $s_1, s_2 \in L$ and all the combinations produced in the first derivation also belong to L.

Problem # 8

a) As an example the string aaa has two different derivations.

$$S \Rightarrow SS \Rightarrow SSS \Rightarrow aSS \Rightarrow aaS \Rightarrow aaa$$
$$S \Rightarrow SS \Rightarrow aS \Rightarrow aaS \Rightarrow aaa$$

An unambiguous grammar is:

$$\begin{aligned} S &\to AS \mid A \\ A &\to ab \mid a \end{aligned}$$

b) The string a can be derived in two ways:

$$S \Rightarrow ABA \Rightarrow BA \Rightarrow A \Rightarrow aA \Rightarrow a$$
$$S \Rightarrow ABA \Rightarrow aABA \Rightarrow aBA \Rightarrow aA \Rightarrow a$$

An unambiguous grammar is:

$$\begin{aligned} S &\to ABA \mid AB \mid BA \mid A \mid B \\ A &\to aA \mid a \\ B &\to aB \mid b \end{aligned}$$

c) The string $aaabb$ can be derived in two ways:

$$S \Rightarrow aSb \Rightarrow aaaSbb \Rightarrow aaabb$$
$$S \Rightarrow aaSb \Rightarrow aaaSbb \Rightarrow aaabb$$

An unambiguous grammar is:

$$\begin{aligned} S &\rightarrow aSb \mid aaAb \mid \epsilon \\ A &\rightarrow aaAb \mid \epsilon \end{aligned}$$

Problem # 9

a) The idea is to push a *"0"* into the stack for every observation of 0 and then pop it upon observation of 1. If we observe a 1 while the stack is empty then we go the trap state, since there exists a prefix with more 1's than 0's. The PDA $M = (\{q_0, q_t, q_f\}, \{0,1\}, \{0, Z_0\}, \delta, q_0, Z_0, q_f)$ where δ is:

state	input	stack	new state	new stack
q_0	0	Z_0	q_0	$0Z_0$
q_0	0	0	q_0	00
q_0	1	0	q_0	ϵ
q_0	1	Z_0	q_t	Z_0
q_0	0	0	q_f	0
q_0	ϵ	Z_0	q_f	Z_0

b) The PDA $M = (\{q_1, q_2\}, \{0,1\}, \{Z_0, X, Y\}, \delta, q_1, Z_0, \varnothing)$ (accepting by empty stack) where δ is:

state	input	stack	new state	new stack
q_1	0	Z_0	q_1	XZ_0
q_1	1	Z_0	q_2	Z_0
q_1	1	Y	q_2	Y
q_1	0	Y	q_1	ϵ
q_1	1	X	q_2	X
q_1	0	X	q_1	XX
q_1	ϵ	Z_0	q_1	ϵ

c) The PDA $M = (\{q_1, q_2, q_3, q_4, q_5, q_6\}, \{a, b\}, \{Z_0, X, Y\}, \delta, q_1, Z_0, q_6)$ (accepting by empty stack) where δ is presented in Table 1.

Another idea is to start with a CFG and then convert it to a PDA with a single state. A CFG describing the language is:

$$\begin{aligned} S &\rightarrow AB \mid BA \mid A \mid B \\ A &\rightarrow aAa \mid aAb \mid bAa \mid bAb \mid a \\ B &\rightarrow aBa \mid aBb \mid bBa \mid bBb \mid b \end{aligned}$$

state	input	stack	new state	new stack
q_1	b	Z_0	q_1	XZ_0
q_1	b	X	q_1	XX
q_1	a	Z_0	q_1	XZ_0
q_1	a	X	q_1	XX
q_1	a	Z_0	q_2	Z_0
q_1	a	X	q_2	X
q_1	b	Z_0	q_4	Z_0
q_1	b	X	q_4	X
q_2	a	X	q_2	ϵ
q_2	a	Z_0	q_2	YZ_0
q_2	a	Y	q_2	YY
q_2	b	X	q_2	ϵ
q_2	b	Z_0	q_2	YZ_0
q_2	b	Y	q_2	YY
q_2	b	Z_0	q_3	Z_0
q_2	b	Y	q_3	Y
q_4	a	X	q_4	ϵ
q_4	a	Z_0	q_4	YZ_0
q_4	a	Y	q_4	YY
q_4	b	X	q_4	ϵ
q_4	b	Z_0	q_4	YZ_0
q_4	b	Y	q_4	YY
q_4	a	Z_0	q_5	Z_0
q_4	a	Y	q_5	Y
q_3	a	Y	q_3	ϵ
q_3	b	Y	q_3	ϵ
q_5	a	Y	q_5	ϵ
q_5	b	Y	q_5	ϵ
q_3	ϵ	Z_0	q_6	Z_0
q_5	ϵ	Z_0	q_6	Z_0

Table 1: Problem 9, part (c): δ function