

(1.)

- 2.) a.)
- i. 1
 - ii. 2
 - iii. 3
 - iv. 4
 - v. 5

b.)

$$[x_1] = \{ x \in \{a,b\}^* \mid x \text{ does not end in } b, |x| \text{ even} \}$$

$$[x_2] = \{ x \in \{a,b\}^* \mid |x| \text{ odd} \}$$

$$[x_3] = [x_2]$$

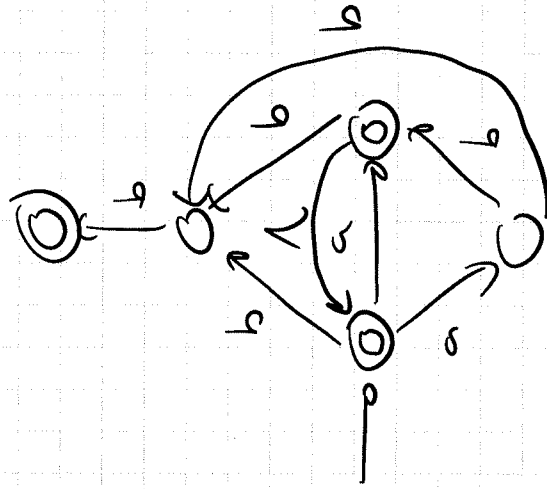
$$[x_4] = [x_1]$$

$$[x_5] = L$$

c.) No, because there are multiple states corresponding to the same equivalence class;
 or there are multiple states with the same future set;
 or an FA with fewer states is provided;
 or the minimization algorithm is performed.

- 3.) a.)
 - i. yes
 - ii. no
 - iii. yes
 - iv. yes

b.)



4.) Suppose L is accepted by an automaton having n states, $n \geq 1$.

Note $x = a^n b^n \in L$, as $N_a(x) = n = N_b(x)$ and

$$n \geq n \geq n - 2.$$

Moreover, $|x| = 2n \geq n$.

Let $x = uvw$ with $|uv| \leq n$ and $|v| \geq 1$ be arbitrary.

Then $u = a^i$, $v = a^j$, $i+j \leq n$, $j \geq 1$.

However, $uv^2w = a^{n+2j}b^n \notin L$,

because $n+2j-2 \geq n+1 \neq n$.

Hence, L is not regular.

21.11

5) a) Yes, $L(G)$ is regular. You can easily simulate all productions by regular productions.

21.13

b) Yes $L(G)$ is regular. The language can be described by the regular expression $a^*(a+b)b^*$

21.14

c) No $L(G)$ is not regular. The language is $\{a^n \{a,b\} b^n \mid n \geq 0\}$

21.19

d) Yes, $L(G)$ is regular. G is a regular grammar.

21.20

6) a) The first six elements in the canonical (shortlex) order of L :
 $a, aa, aaa, aab, aba, baa$

21.22 The next elements are: $aaaa, aaab, aaba, abaa, baaa, aaaaa$

b) G has startvariable S and the following transitions

$S \rightarrow S_1 \mid S_2 \mid S_3$ S_1 for $j=0 \rightarrow$ a non-empty string of a 's
 S_2 for $1 \leq j < i$
 S_3 for $j \geq i$, so $i \leq j < i+k$

$S_1 \rightarrow aS_1 \mid a$

$S_2 \rightarrow XA$

X for $a^i b^j$ with $1 \leq j < i$

A for a^k with $k \geq 0$

$X \rightarrow aXb \mid aAAb$

for every b also an a in a^i ,
 and at least one a more

$A \rightarrow aA \mid \Lambda$

$S_3 \rightarrow YZ$

$Y \rightarrow aYb \mid \Lambda$

for every a in a^i also a b .

$Z \rightarrow bZa \mid aA$

for every extra b an a in a^k ,
 and at least one a more.

21.40

A simple ambiguous grammar would be:

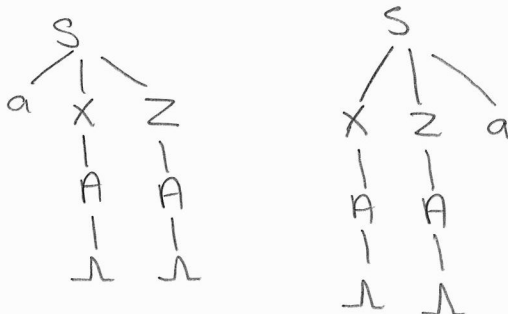
$S \rightarrow aXZ \mid XZa$

X for b 's that are matched by a 's in a^i
 Z for b 's that are matched by a 's in a^h
 at least one a more

$X \rightarrow aXb \mid A$

$Z \rightarrow bZa \mid A$

$A \rightarrow aA \mid \Lambda$



Two derivation trees
 for $x=a$

21.45

7)

21.47 a) Set of nullable variables: $\{B\}$

b) Add productions where occurrences (or an occurrence) of B are left out. G_2 has start variable S and the following productions

$$\begin{aligned} S &\rightarrow XBBb \mid XB \mid XBb \mid Xb \mid X \\ X &\rightarrow aX \mid ab \\ B &\rightarrow bB \mid b \end{aligned}$$

21.50

c) There is one unit production: $S \rightarrow X$
 G_3 has start variable S and the following productions:

$$\begin{aligned} S &\rightarrow XBBb \mid XB \mid XBb \mid Xb \mid aX \mid ab \\ X &\rightarrow aX \mid ab \\ B &\rightarrow bB \mid b \end{aligned}$$

21.53

d) G_4 has start variable S and the following productions:

$$\begin{aligned} S &\rightarrow XBBZ_b \mid XB \mid XBZ_b \mid XZ_b \mid Z_aX \mid Z_aZ_b \\ X &\rightarrow Z_aX \mid Z_aZ_b \\ B &\rightarrow Z_bB \mid b \\ Z_a &\rightarrow a \\ Z_b &\rightarrow b \end{aligned}$$

(we use Z_a and Z_b , rather than X_a and X_b to avoid confusion with variable X)

21.56

e) Two productions are too long: $S \rightarrow XBBZ_b \mid XBZ_b$

G_5 has start variable S and the following productions:

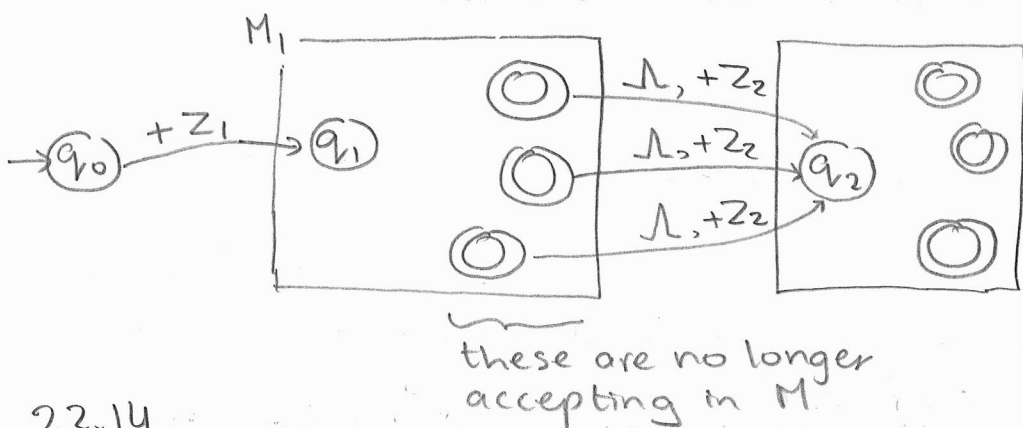
$$\begin{aligned} S &\rightarrow XY_1 \mid XB \mid XY_3 \mid XZ_b \mid Z_aX \mid Z_aZ_b \\ Y_1 &\rightarrow BY_2 \\ Y_2 &\rightarrow BZ_b \\ Y_3 &\rightarrow BZ_b \\ X &\rightarrow Z_aX \mid Z_aZ_b \\ B &\rightarrow Z_bB \mid b \\ Z_a &\rightarrow a \\ Z_b &\rightarrow b \end{aligned}$$

It is possible to combine Y_2 and Y_3 in a single variable.

22.01

8/ M has ^{new} start state q_0 and ^{new} initial stack symbol Z_0 ,
 M starts by pushing Z_1 onto the stack, while moving from q_0 to q_1 , the start state of M_1 .
 M then performs a computation in M_1 .
 If M arrives in an accepting state of M_1 , it may take a Λ -transition (for any stack symbol in $\Gamma, \cup \{Z_0\}$) to q_2 , the start state of M_2 , while pushing Z_2 onto the stack.
 M then performs a computation in M_2 .
 The (only) accepting states of M are the ones of M_2 .

In pictures:



22.14

9a) Occurrence 1: state 1, input a , stack symbol Z_0
 (the notation $a, +A$ includes $a, Z_0 / AZ_0$)

Occurrence 2: state 2, input b , or Λ , stack symbol Z_0
 (the notation $b, +B$ includes b, Z_0, BZ_0)

22.19

b) $L_e(M_1) = \{ x \in \{a,b\}^* \mid n_a(x) = n_b(x) + 1 \text{ and } x \text{ ends with } a \}$

M keeps track of the difference in the numbers of a's and b's read so far. In state 1, M counts an excess in a's by symbols A on the stack. In state 2, M counts an excess in b's by symbols B on the stack.

A's are pushed in state 1 on reading a's, and are popped in that state on reading b's. Similarly, B's are pushed in state 2 on reading b's, and are popped in that state on reading a's.

When the top stack symbol is Z_0 , this is the only symbol on the stack, and the number of a's read so far is equal to the number of b's. If we are in state 2, then we can follow the Λ -transition to state 1. If we are in state 1 with top stack symbol Z_0 , then we can pop Z_0 on reading an extra a, and accept the string, as the stack is completely empty.