1. Convert the following grammar to Chomsky normal form:

$S \rightarrow AB|BCS$
$A \rightarrow aA|C$
$B \rightarrow bbB|b$
$C \rightarrow cC|\lambda$

Using the following rules and notation:

START: Eliminate the start symbol from the right-hand side
TERM: Eliminate rules with non-solitary terminals
BIN: Eliminate right-hand sides with more than 2 nonterminals
DEL: Eliminate lambda-rules
UNIT: Eliminate unit rules

START:

$S_0 \rightarrow S$ (START)
$S \rightarrow AB|BCS$
$A \rightarrow aA|C$
$B \rightarrow bbB|b$
$C \rightarrow cC|\lambda$

TERM:

$S_0 \rightarrow S$
$S \rightarrow AB|BCS$
$A \rightarrow X_1A|C$ (TERM)
$B \rightarrow X_2X_2B|b$ (TERM)
$C \rightarrow X_3C|\lambda$ (TERM)
$X_1 \rightarrow a$
$X_2 \rightarrow b$
$X_3 \rightarrow c$

BIN:

$S_0 \rightarrow S$
$S \rightarrow AB|X_4S$ (BIN)
$A \rightarrow X_1A|C$
$B \rightarrow X_5B|b$ (BIN)
$C \rightarrow X_3C|\lambda$
$X_1 \rightarrow a$
$X_2 \rightarrow b$
$X_3 \rightarrow c$
$X_4 \rightarrow BC$
$X_5 \rightarrow X_2X_2$

DEL:

$S_0 \rightarrow S$
$S \rightarrow AB|X_4S$
$A \rightarrow X_1A|C|\lambda$ (DEL)
$B \rightarrow X_5B|b$
$C \rightarrow X_3C$ (DEL)
$X_1 \rightarrow a$
$X_2 \rightarrow b$
$X_3 \rightarrow c$
$X_4 \rightarrow BC$
$X_5 \rightarrow X_2X_2$

$S_0 \rightarrow S$
$S \rightarrow AB|X_4S|B$ (DEL)
$A \rightarrow X_1A|C$ (DEL)
$B \rightarrow X_5B|b$
$C \rightarrow X_3C$
$X_1 \rightarrow a$
$X_2 \rightarrow b$
$X_3 \rightarrow c$
$X_4 \rightarrow BC$
$X_5 \rightarrow X_2X_2$

UNIT:

$S_0 \rightarrow S$
$S \rightarrow AB|X_4S|X_5X_6|b$ (UNIT)
$A \rightarrow X_1A|C$
$\cancel{B \rightarrow X_5B|b}$ (UNIT)
$C \rightarrow X_3C$
$X_1 \rightarrow a$
$X_2 \rightarrow b$
$X_3 \rightarrow c$
$X_4 \rightarrow BC$
$X_5 \rightarrow X_2X_2$
$X_6 \rightarrow X_5X_6|b$ (UNIT)

The final produced grammar is

$S_0 \rightarrow S$
$S \rightarrow AB|X_4S|X_5X_6|b$
$A \rightarrow X_1A|C$
$C \rightarrow X_3C$
$X_1 \rightarrow a$
$X_2 \rightarrow b$
$X_3 \rightarrow c$
$X_4 \rightarrow BC$
$X_5 \rightarrow X_2X_2$
$X_6 \rightarrow X_5X_6|b$

2. Show that all the symbols of the following grammar are useful

S → A|CB
A → C|D
B → bB|b
C → cC|c
D → dD|d

Construct an equivalent grammar G_c by removing the chain rules from the grammar. Show that G_c contains useless symbols

S → A and A → C, so by transitivity S → C
S → A and A → D, so by transitivity S → D

therefore

S → cC|c|dD|d|CB
A → C|D
B → bB|b
C → cC|c
D → dD|d

A is now useless since no other rules use it, so we can remove it

S → cC|c|dD|d|CB
B → bB|b
C → cC|c
D → dD|d

Every rule can derive a terminal and every rule is reachable by S

3. Give the upper diagonal matrix produced by the CYK algorithm when run with the Chomsky normal form grammar:

S → AT|AB
T → XB
X → AT|AB
A → a
B → b

4. Construct a grammar G' that contains no left-recursive rules and is equivalent to

S → A|C
A → AaB|AaC|B|a
B → Bb|Cb
C → cC|c

Left Recursion Substitution Rule:

A -> β_1A^1|β_2A^1 … |β_nA^1
A^1 -> α_1A^1|α_2A^1 … |α_mA^1

S → A|C
A → AaB|AaC|B|a (Left Recursive)
B → Bb|Cb
C → cC|c

Put in a more clear form for substitution:

A → AaB|AaC|a
A → B

β_1 = a
α_1 = aB
α_2 = aC

A -> β_1A^1
A^1 -> α_1A^1|α_2A^1|λ

Substitute alphas and beta:

A -> aA^1
A^1 -> aBA^1|aCA^1|λ

New grammar:

$S \rightarrow A|C$
$A \rightarrow aA^1|B$
$B \rightarrow Bb|Cb$ (Left Recursive)
$C \rightarrow cC|c$
$A^1 \text{ -> } aBA^1|aCA^1|\lambda$

Substitute C in the B rule

$B \rightarrow Bb|cCb|cb$

Left Recursion Substitution Rule

β_1 = cCb
β_2 = cb
α = b

$B \text{ -> } \beta_1 B^1|\beta_2 B^1$
$B^1 \text{ -> } \alpha B^1|\lambda$

$B \text{ -> } cCbB^1|cbB^1$
$B^1 \text{ -> } bB^1|\lambda$

New grammar:

$S \rightarrow A|C$
$A \rightarrow aA^1|B$
$B \rightarrow cCbB^1|cbB^1$
$C \rightarrow cC|c$
$A^1 \text{ -> } aBA^1|aCA^1|\lambda$
$B^1 \text{ -> } bB^1|\lambda$

5. Convert the Chomsky normal form grammar:

$S \rightarrow AB$
$A \rightarrow BB|CC$
$B \rightarrow AD|CA$
$C \rightarrow a$
$D \rightarrow b$

to Greibach normal form. Process the variables according to the order S, A, B, C, D.

Using the following rules and notation:

START: Eliminate the start symbol from the right-hand side
DEL: Eliminate lambda-rules
UNIT: Eliminate unit rules
LR: Eliminate all direct and indirect left-recursion
SUB: Perform substitutions to convert to GNF

START:

There are no occurrences of S on the right hand side of any of the rules, so skip.

DEL:

There are no occurrences of lambda on the right hand side of any of the rules, so skip.

UNIT:

There are no occurrences of unit productions for now, so skip.

LR:

$S \rightarrow AB$
$A \rightarrow BB|CC$
$B \rightarrow AD|CA$ (Indirect LR)
$C \rightarrow a$
$D \rightarrow b$

$B \to CCDB^1|CAB^1$
$B^1 \to BDB^1|\lambda$

New Grammar:

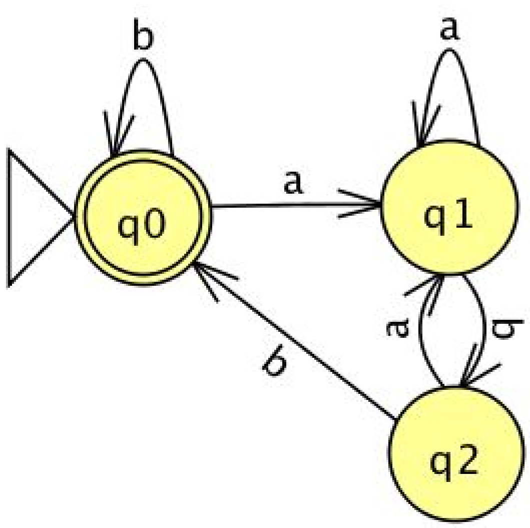
$S \rightarrow AB$
$A \rightarrow BB|CC$
B -> $CCDB^1|CAB^1$
$C \rightarrow a$
$D \rightarrow b$
B^1 -> $BDB^1|\lambda$

6. Let M be the deterministic finite automaton:

```
Q = {q0,q1,q2}
Σ = {a,b}
F = {q0}

δ  | a  | b
---------------
q0 | q1 | q0
q1 | q1 | q2
q2 | q1 | q0
```

(a) Give the state diagram of M



(b) Trace the computation of M that processes *babaab*

```
[q0, babaab]
|- [q0, abaab]
|- [q1, baab]
|- [q2, aab]
|- [q1, ab]
|- [q1, b]
|- [q, λ]
```

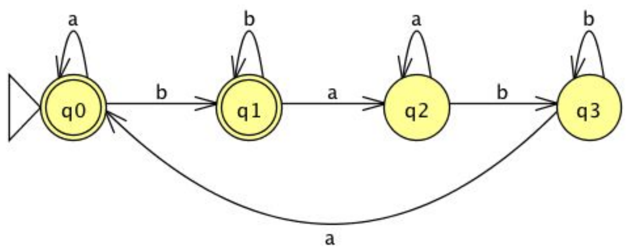
(c) Give a regular expression for L(M)

```
b*aa*b(aa*b)*b)*
```

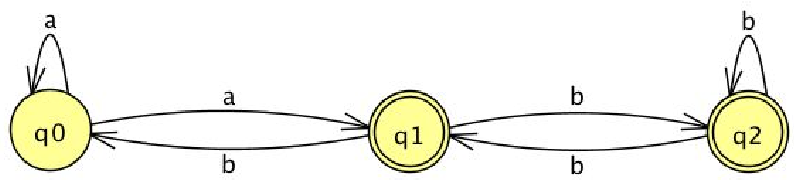
(d) Give a regular expression for the language accepted if both q0 and q1 are accepting states

```
(b*aa*b(aa*b)*b)*) ∪ (a*b(aa*b)*b)*)
```

7. Build a DFA that accepts the language: The set of strings over {a,b} that contain an even number of substrings *ba*



8. Let M be the nondeterministic finite automaton:



(a) Construct the transition table of M

δ	a	b
q0	{q0,q1}	λ
q1	λ	{q0,q2}
q2	λ	{q1,q2}

(b) Trace all computations of the string *aabb* in M

```
   [q0, aabb]
|- [q0, abb]
|- [q1, bb]
|- [q2, b]
|- [q2, λ]

   [q0, aabb]
|- [q0, abb]
|- [q1, bb]
|- [q2, b]
|- [q1, λ]
```

(c) Is *aabb* in L(M)

Yes

(d) Give a regular expression for L(M)

aab*(aab*)*

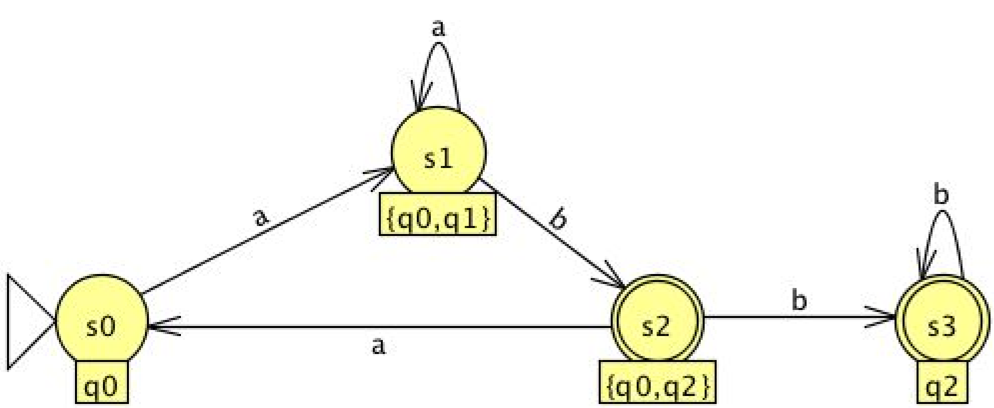
(e) Construct a DFA that accepts L(M)

Initial NFA State Table

	q	δ(q,a)	δ(q,b)
start ->	q0	{q0,q1}	∅
	q1	∅	{q0,q2}
end ->	q2	∅	{q1,q2}

DFA State Table

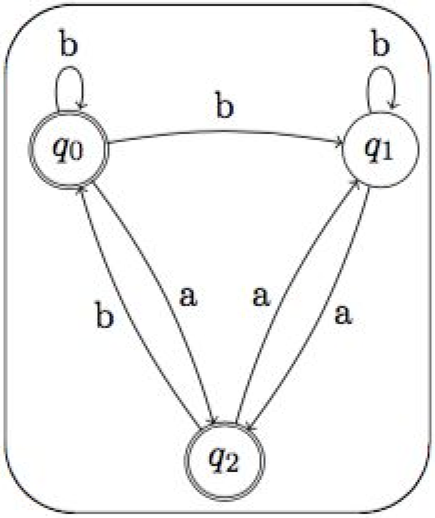
	q	δ(q,a)	δ(q,b)
start ->	q0	{q0,q1}	∅
	{q0,q1}	{q0,q1}	{q0,q2}
end ->	{q0,q2}	q0	q2
end ->	q2	∅	q2



(f) Give a regular expression for the language accepted if q0 and q1 are accepting states

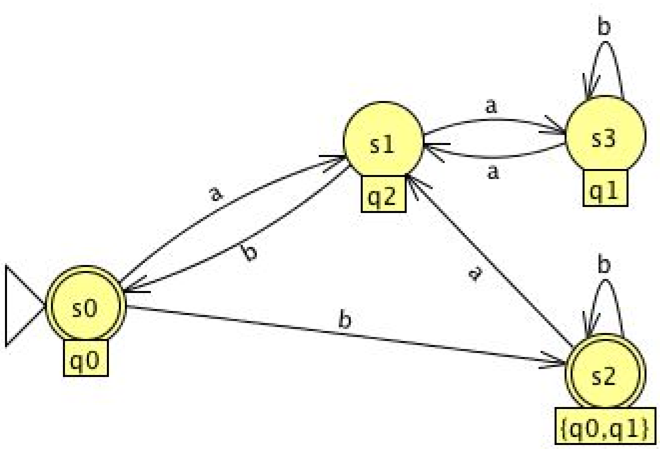
aa*((ba*) ∪ (bb*b) ∪ (bb*bba*))

9. Let M be the NFA:



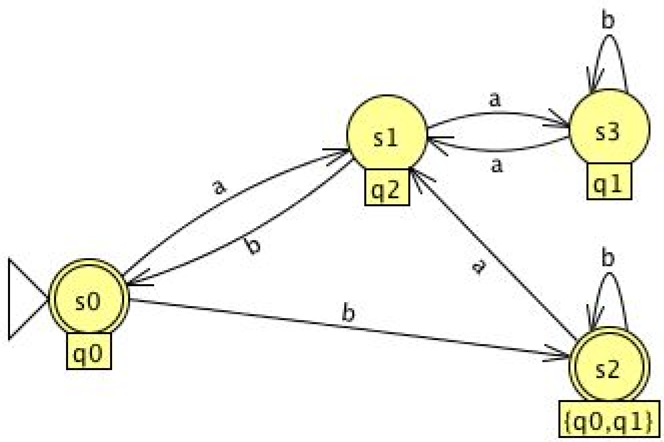
(a) Use algorithm 5.6.3 to construct the state diagram of an equivalent DFA

	q	δ(q,a)	δ(q,b)
start, end ->	q0	q2	{q0,q1}
	q2	q1	q0
end ->	{q0,q1}	q2	{q0,q1}
	q1	q2	q1

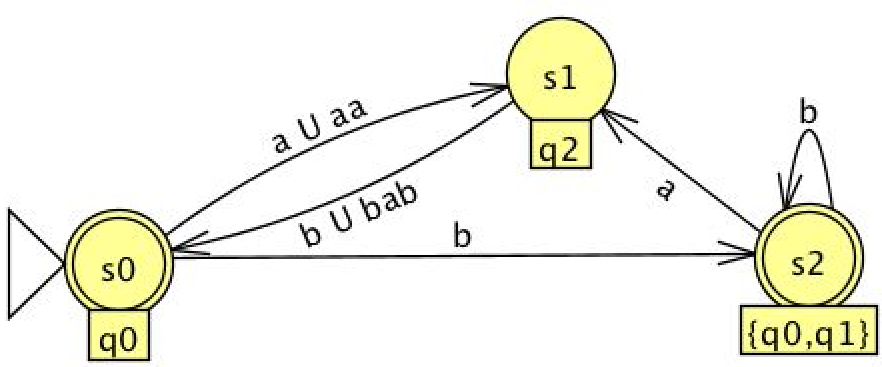


(b) Use algorithm 6.2.2 to construct a regular expression for the language accepted by the automaton

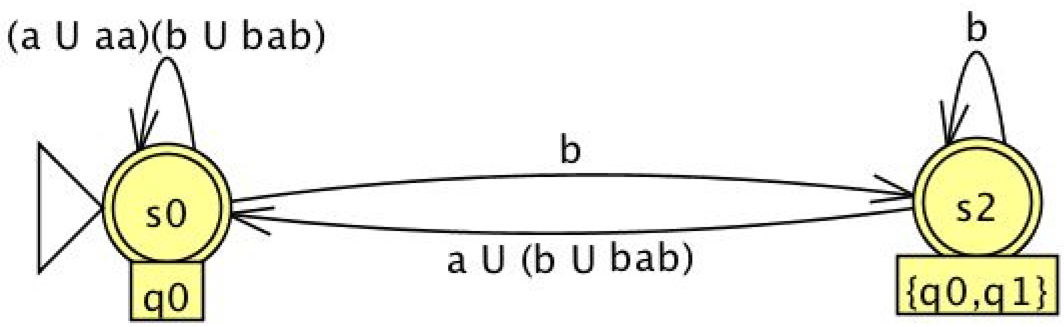
Original



Remove s3



Remove s1



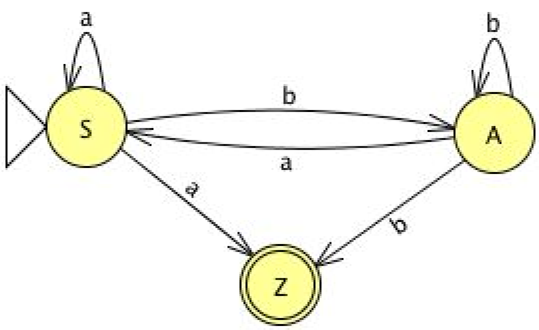
Resulting regular expression:

((a ∪ aa)(b ∪ bab))*b(b* ∪ (a ∪ (b ∪ bab)))*

10. Let G be the grammar

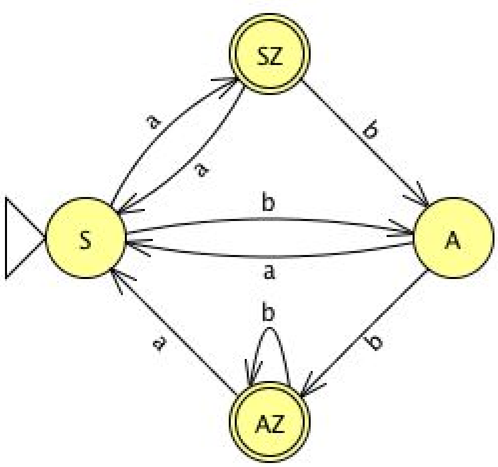
S → aS|bA|a
A → aS|bA|b

(a) Use Theorem 6.3.1 to build an NFA *M* that accepts L(G)



(b) Using the result of part (a), build a DFA M' that generates L(M')

	q	δ(q,a)	δ(q,b)
start ->	S	{S,Z}	A
end ->	{S,Z}	S	A
	A	S	{A,Z}
end ->	{A,Z}	S	{A,Z}



(c) Construct a regular grammar from M that generates L(M)

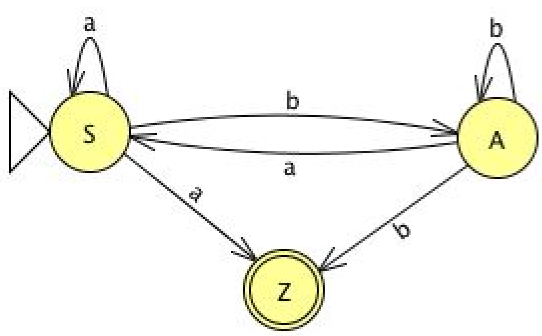
S → aS|bA|aZ
A → aS|bA|bB
Z → λ

(d) Construct a regular grammar from M' that generates L(M')

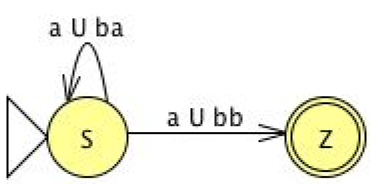
V = {S,S',A,A') // For clarity, change states SZ and AZ to variables S' and A'
S → aS|aS'
S' → bA|aS
A → bA'|aS
A' → bA'|aS| λ

(e) Give the regular expression for M using algorithm 6.2.2

Original



Remove State A



Regular expression

(a ∪ ba)*(a ∪ bb)