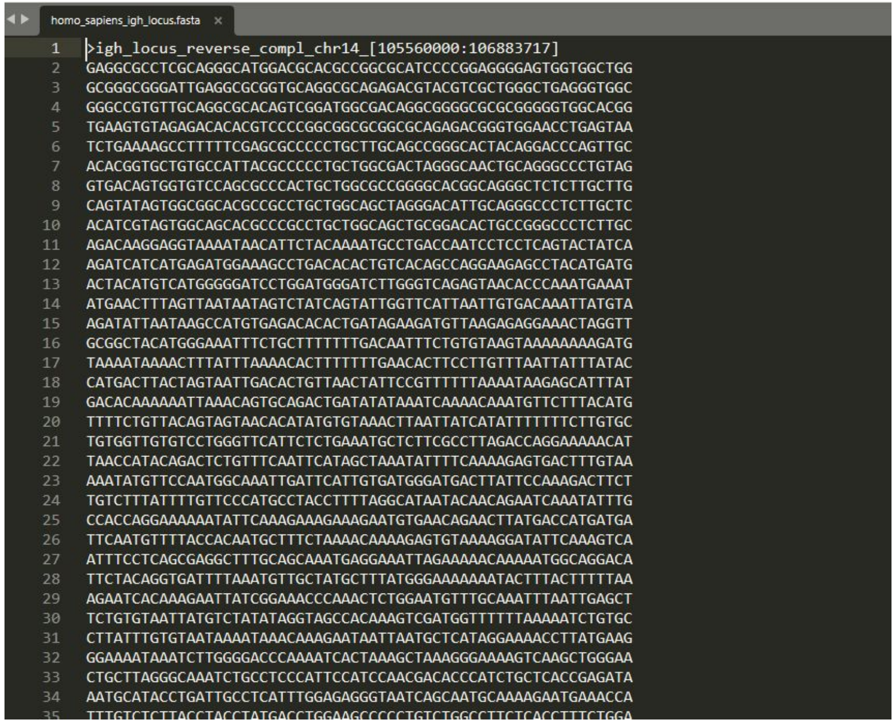
DFAs, NFAs and Regular Expressions

CSE 105 Week 3 Discussion

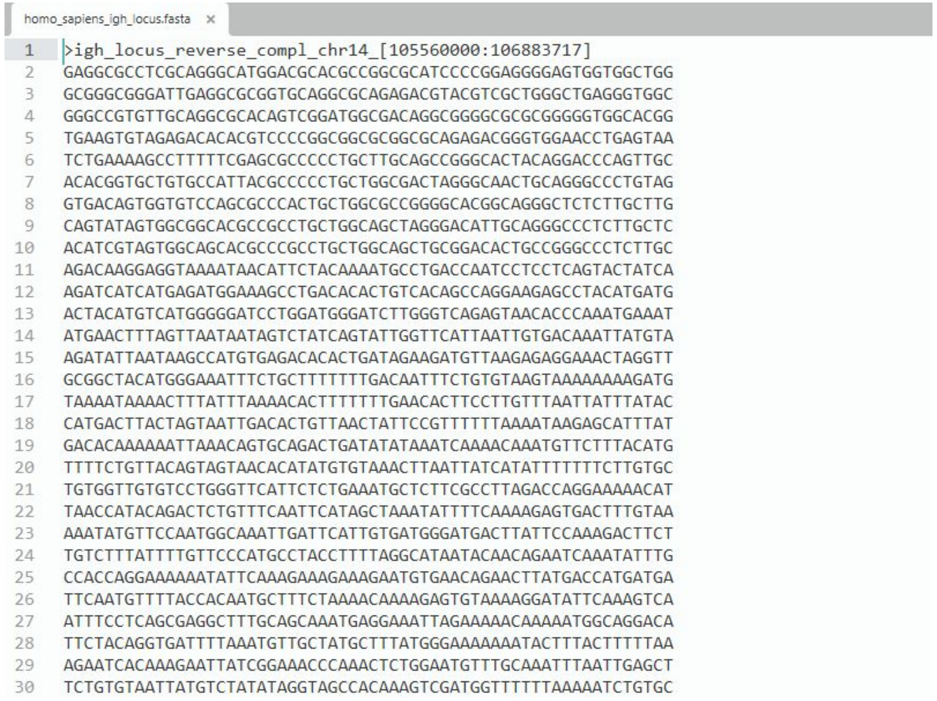
Deadlines and Logistics

- Review your HW 1 grade
- Schedule your tests asap on PrairieTest !
- HW 3 due next week on 22nd (Tuesday) at 5 PM
- As always, slides are not self-contained
- Link for the slides : https://docs.google.com/presentation/d/1vYfZESYxh_BaQ-rno0CJJbf4R40-iHGaWi_nNVcK40I/edit?usp=sharing

Poll : Dark mode fan or light mode enjoyer ?

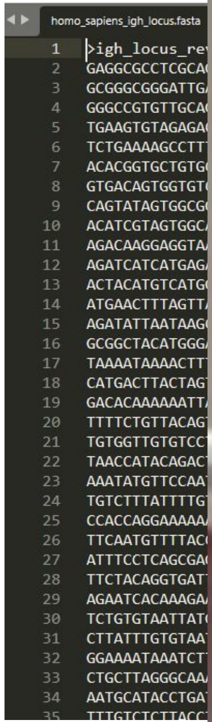


```
>igh_locus_reverse_compl_chr14_[105560000:106883717]
GAGGCGCCTCGCAGGGCATGGACGCACGCCGGCGCATCCCCGGAGGGGAGTGGTGGCTGG
GCGGGCGGGATTGAGGCGCGGTGCAGGCGCAGAGACGTACGTCGCTGGGCTGAGGGTGGC
GGGCCGTGTTGCAGGCGCACAGTCGGATGGCGACAGGCGGGGCGCGCGGGGGTGGCACGG
TGAAGTGTAGAGACACACGTCCCCGGCGGCGCGGCGCAGAGACGGGTGGAACCTGAGTAA
TCTGAAAAGCCTTTTTCGAGCGCCCCCTGCTTGCAGCCGGGCACTACAGGACCCAGTTGC
ACACGGTGCTGTGCCATTACGCCCCCTGCTGGCGACTAGGGCAACTGCAGGGCCCTGTAG
GTGACAGTGGTGTCCAGCGCCCACTGCTGGCGCCGGGGCACGGCAGGGCTCTCTTGCTTG
CAGTATAGTGGCGGCACGCCGCCTGCTGGCAGCTAGGGACATTGCAGGGCCCTCTTGCTC
ACATCGTAGTGGCAGCACGCCCGCCTGCTGGCAGCTGCGGACACTGCCGGGCCCTCTTGC
AGACAAGGAGGTAAAATAACATTCTACAAAATGCCTGACCAATCCTCCTCAGTACTATCA
AGATCATCATGAGATGGAAAGCCTGACACACTGTCACAGCCAGGAAGAGCCTACATGATG
ACTACATGTCATGGGGGATCCTGGATGGGATCTTGGGTCAGAGTAACACCCAAATGAAAT
ATGAACTTTAGTTAATAATAGTCTATCAGTATTGGTTCATTAATTGTGACAAATTATGTA
AGATATTAATAAGCCATGTGAGACACACTGATAGAAGATGTTAAGAGAGGAAACTAGGTT
GCGGCTACATGGGAAATTTCTGCTTTTTTTGACAATTTCTGTGTAAGTAAAAAAAAGATG
TAAAATAAAACTTTATTTAAAACACTTTTTTTGAACACTTCCTTGTTTAATTATTTATAC
CATGACTTACTAGTAATTGACACTGTTAACTATTCCGTTTTTTAAAATAAGAGCATTTAT
GACACAAAAAATTAAACAGTGCAGACTGATATATAAATCAAAACAAATGTTCTTTACATG
TTTTCTGTTACAGTAGTAACACATATGTGTAAACTTAATTATCATATTTTTTTCTTGTGC
TGTGGTTGTGTCCTGGGTTCATTCTCTGAAATGCTCTTCGCCTTAGACCAGGAAAAACAT
TAACCATACAGACTCTGTTTCAATTCATAGCTAAATATTTTCAAAAGAGTGACTTTGTAA
AAATATGTTCCAATGGCAAATTGATTCATTGTGATGGGATGACTTATTCCAAAGACTTCT
TGTCTTTATTTTGTTCCCATGCCTACCTTTTAGGCATAATACAACAGAATCAAATATTTG
CCACCAGGAAAAAATATTCAAAGAAAGAAAGAATGTGAACAGAACTTATGACCATGATGA
TTCAATGTTTTACCACAATGCTTTCTAAAACAAAAGAGTGTAAAAGGATATTCAAAGTCA
ATTTCCTCAGCGAGGCTTTGCAGCAAATGAGGAAATTAGAAAAACAAAAATGGCAGGACA
TTCTACAGGTGATTTTAAATGTTGCTATGCTTTATGGGAAAAAAATACTTTACTTTTTAA
AGAATCACAAAGAATTATCGGAAACCCAAACTCTGGAATGTTTGCAAATTTAATTGAGCT
TCTGTGTAATTATGTCTATATAGGTAGCCACAAAGTCGATGGTTTTTTAAAAATCTGTGC
CTTATTTGTGTAATAAAATAAACAAAGAATAATTAATGCTCATAGGAAAACCTTATGAAG
GGAAAATAAATCTTGGGGACCCAAAATCACTAAAGCTAAAGGGAAAAGTCAAGCTGGGAA
CTGCTTAGGGCAAATCTGCCTCCCATTCCATCCAACGACACCCATCTGCTCACCGAGATA
AATGCATACCTGATTGCCTCATTTGGAGAGGGTAATCAGCAATGCAAAAGAATGAAACCA
TTTGTCTCTTACCTACCTATGACCTGGAAGCCCCCTGTCTGGCCTTCTCACCTTTCTGGA
```



```
>igh_locus_reverse_compl_chr14_[105560000:106883717]
GAGGCGCCTCGCAGGGCATGGACGCACGCCGGCGCATCCCCGGAGGGGAGTGGTGGCTGG
GCGGGCGGGATTGAGGCGCGGTGCAGGCGCAGAGACGTACGTCGCTGGGCTGAGGGTGGC
GGGCCGTGTTGCAGGCGCACAGTCGGATGGCGACAGGCGGGGCGCGCGGGGGTGGCACGG
TGAAGTGTAGAGACACACGTCCCCGGCGGCGCGGCGCAGAGACGGGTGGAACCTGAGTAA
TCTGAAAAGCCTTTTTCGAGCGCCCCCTGCTTGCAGCCGGGCACTACAGGACCCAGTTGC
ACACGGTGCTGTGCCATTACGCCCCCTGCTGGCGACTAGGGCAACTGCAGGGCCCTGTAG
GTGACAGTGGTGTCCAGCGCCCACTGCTGGCGCCGGGGCACGGCAGGGCTCTCTTGCTTG
CAGTATAGTGGCGGCACGCCGCCTGCTGGCAGCTAGGGACATTGCAGGGCCCTCTTGCTC
ACATCGTAGTGGCAGCACGCCCGCCTGCTGGCAGCTGCGGACACTGCCGGGCCCTCTTGC
AGACAAGGAGGTAAAATAACATTCTACAAAATGCCTGACCAATCCTCCTCAGTACTATCA
AGATCATCATGAGATGGAAAGCCTGACACACTGTCACAGCCAGGAAGAGCCTACATGATG
ACTACATGTCATGGGGGATCCTGGATGGGATCTTGGGTCAGAGTAACACCCAAATGAAAT
ATGAACTTTAGTTAATAATAGTCTATCAGTATTGGTTCATTAATTGTGACAAATTATGTA
AGATATTAATAAGCCATGTGAGACACACTGATAGAAGATGTTAAGAGAGGAAACTAGGTT
GCGGCTACATGGGAAATTTCTGCTTTTTTTGACAATTTCTGTGTAAGTAAAAAAAAGATG
TAAAATAAAACTTTATTTAAAACACTTTTTTTGAACACTTCCTTGTTTAATTATTTATAC
CATGACTTACTAGTAATTGACACTGTTAACTATTCCGTTTTTTAAAATAAGAGCATTTAT
GACACAAAAAATTAAACAGTGCAGACTGATATATAAATCAAAACAAATGTTCTTTACATG
TTTTCTGTTACAGTAGTAACACATATGTGTAAACTTAATTATCATATTTTTTTCTTGTGC
TGTGGTTGTGTCCTGGGTTCATTCTCTGAAATGCTCTTCGCCTTAGACCAGGAAAAACAT
TAACCATACAGACTCTGTTTCAATTCATAGCTAAATATTTTCAAAAGAGTGACTTTGTAA
AAATATGTTCCAATGGCAAATTGATTCATTGTGATGGGATGACTTATTCCAAAGACTTCT
TGTCTTTATTTTGTTCCCATGCCTACCTTTTAGGCATAATACAACAGAATCAAATATTTG
CCACCAGGAAAAAATATTCAAAGAAAGAAAGAATGTGAACAGAACTTATGACCATGATGA
TTCAATGTTTTACCACAATGCTTTCTAAAACAAAAGAGTGTAAAAGGATATTCAAAGTCA
ATTTCCTCAGCGAGGCTTTGCAGCAAATGAGGAAATTAGAAAAACAAAAATGGCAGGACA
TTCTACAGGTGATTTTAAATGTTGCTATGCTTTATGGGAAAAAAATACTTTACTTTTTAA
AGAATCACAAAGAATTATCGGAAACCCAAACTCTGGAATGTTTGCAAATTTAATTGAGCT
TCTGTGTAATTATGTCTATATAGGTAGCCACAAAGTCGATGGTTTTTTAAAAATCTGTGC
```

Dark mode fan or light mode enjoyer ?


homo_sapiens_igh_locus.fasta

```
>igh_locus_re
GAGGCGCCTCGCA
GCGGGCGGGATTG
GGGCCGTGTTGCA
TGAAGTGTAGAGA
TCTGAAAAGCCTT
ACACGGTGCTGTG
GTGACAGTGGTGT
CAGTATAGTGGCG
ACATCGTAGTGGC
AGACAAGGAGGTA
AGATCATCATGAG
ACTACATGTCATG
ATGAACTTTAGTT
AGATATTAATAAG
GCGGCTACATGGG
TAAAATAAAACTT
CATGACTTACTAG
GACACAAAAAATT
TTTTCTGTTACAG
TGTGGTTGTGTCC
TAACCATACAGAC
AAATATGTTCCAA
TGTCTTTATTTTG
CCACCAGGAAAAA
TTCAATGTTTTAC
ATTTCCTCAGCGA
TTCTACAGGTGAT
AGAATCACAAAGA
TCTGTGTAATTAT
CTTATTTGTGTAA
GGAAAATAAATCT
CTGCTTAGGGCAA
AATGCATACCTGA
TTTGTCTCTTACC
```

Current progress - Answer Y/N

1. Given a DFA and a string, I can tell tell if the string is accepted or not
2. Given a DFA, I can identify the language that is recognized by it
3. Given a regular expression or a Language, I can define and draw a DFA

and,

4. Given an NFA and a string, I can tell tell if the string is accepted or not
5. Given an NFA, I can identify the language that is recognized by it
6. Given a regular expression or a Language, I can define and draw an NFA

Today's Topics

1. Recap of ε-transitions in an NFA
2. Closure over U, ∩, ^{c},* and ○ operations in NFAs, DFAs
3. Equivalence of DFAs, NFAs
4. Tying it all together : DFAs, NFAs and regular expressions : Regular languages (if time permits)

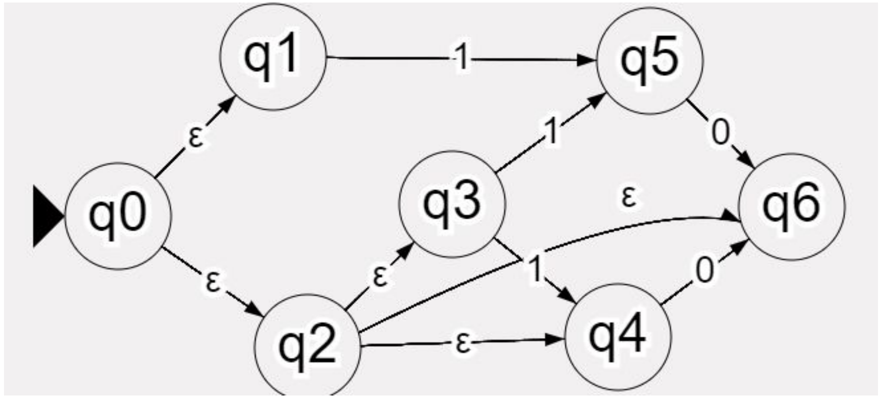
ε-transitions

ε-transitions

ε-transitions are essentially spontaneous moves - you can (and have to) traverse them whenever you encounter them !

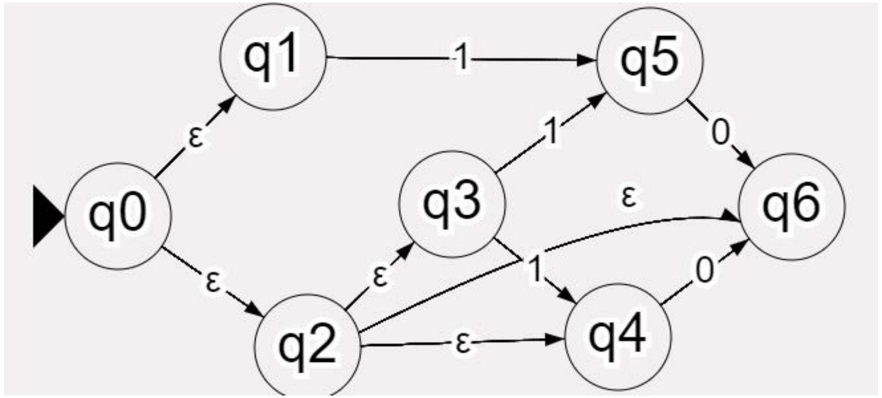
ε-transitions

1. What state(s) do you reach when you read:
   a. 10
   b. 1
   c. 0
   d. ε



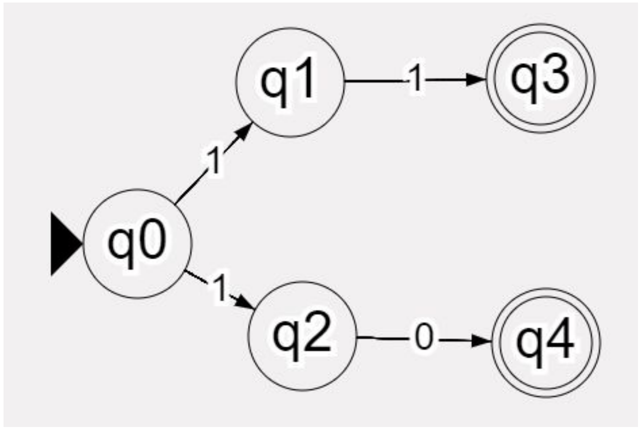
ε-transitions

1. What state(s) do you reach when you read:
   a. 10 : Q6
   b. 1 : Q4, Q5
   c. 0 : Q6
   d. ε: Q0, Q1, Q2, Q3, Q4, Q6



Modify this NFA to...

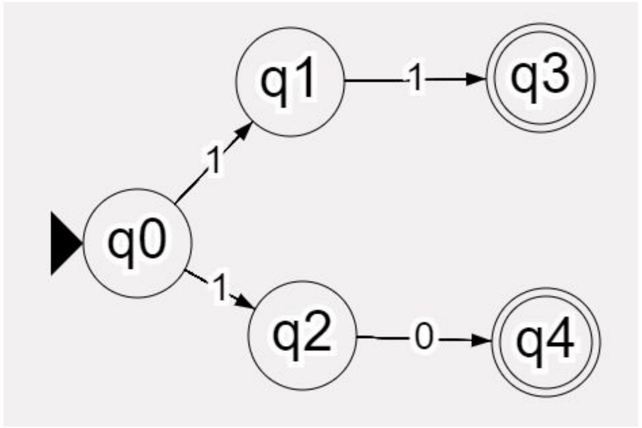
1. Have exactly one accept state
   a. With and without changing Q (the set of states) in the 5-tuple definition
2. Have 5 accept states, $q_3 \notin F$, $q_4 \notin F$

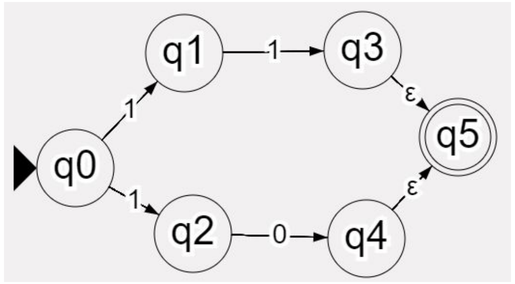


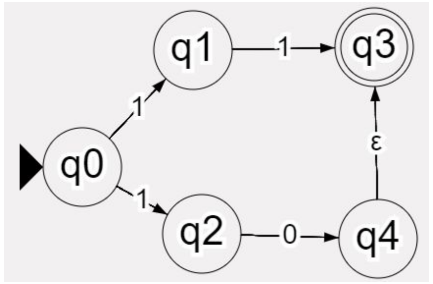
Note - The modified NFA has to recognize the same language !

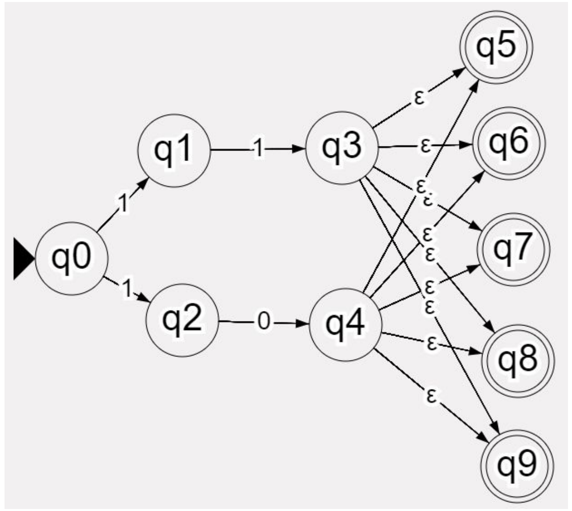
Modify this NFA to...

1. Have exactly one final state
   a. With and without changing Q (the set of states) in the 5-tuple definition
2. Have 5 final states, $q_3 \notin F$, $q_4 \notin F$









DFAs and NFAs closure over U, * and ○

Closure - What we learnt last week

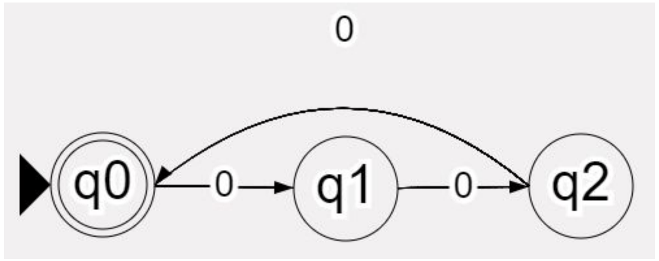
Languages accepted by DFAs are closed under complementation

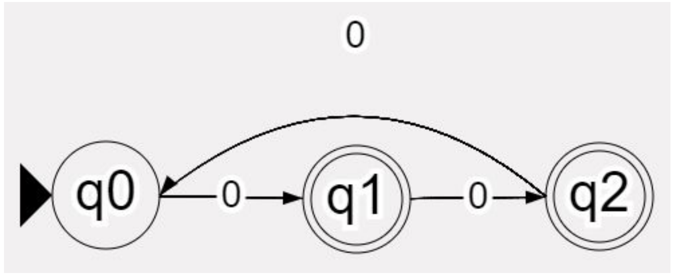
Strategy :

Closure - What we learnt last week

Languages accepted by DFAs are closed under complementation

Strategy : Flip the accept states and non-accept states





Closure - What we learnt last week

Languages accepted by NFAs are closed under union

Strategy:

Closure - What we learnt

Languages accepted by NFAs are closed under union

Strategy:

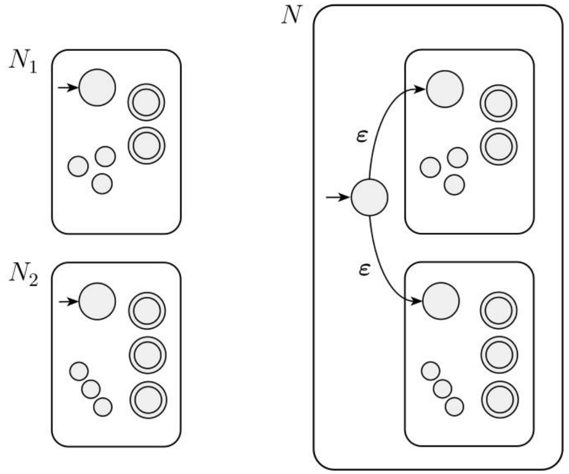


FIGURE 1.46
Construction of an NFA N to recognize $A_1 \cup A_2$

Closure - New stuff

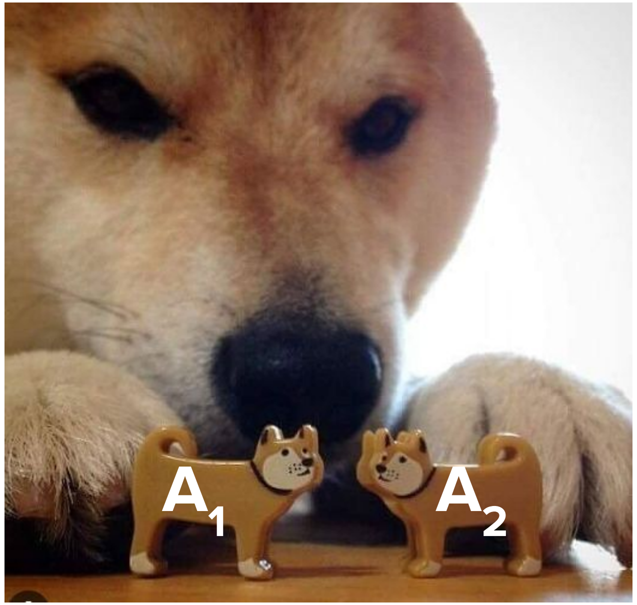
1. **Languages accepted by DFAs are closed under union**
2. **Languages accepted by DFAs are closed under intersection**

Strategy:

Closure - New stuff

1. **Languages accepted by DFAs are closed under union**
2. **Languages accepted by DFAs are closed under intersection**

Strategy: Parallel Computation



Motivating example : $\Sigma = \{0,1\}$

$L(A_1)$: Set of all strings over Σ containing even number of 0's

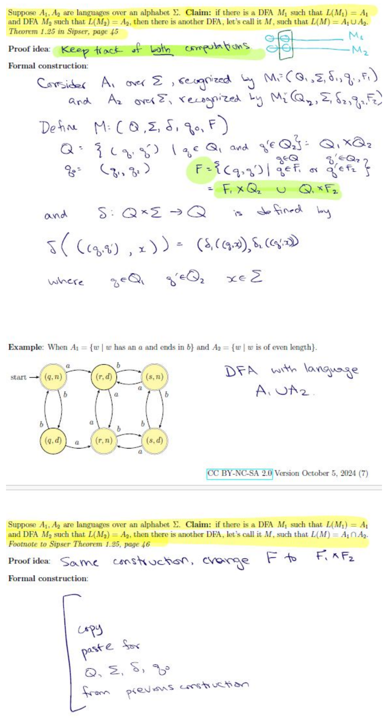
$L(A_2)$: Set of all strings containing non negative integer repeats of 10

A_1 and A_2 are DFAs

Create a DFA A such that :

1. $L(A) = A_1 \cup A_2$
2. $L(A) = A_1 \cap A_2$

Last Friday's lecture formalized this process !



Suppose A_1, A_2 are languages over an alphabet Σ. **Claim:** if there is a DFA M_1 such that $L(M_1) = A_1$ and DFA M_2 such that $L(M_2) = A_2$, then there is another DFA, let's call it M, such that $L(M) = A_1 \cup A_2$.
Theorem 1.25 in Sipser, page 45

Proof idea: Keep track of both computations

Formal construction:

Consider A_1 over Σ, recognized by $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
and A_2 over Σ, recognized by $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

Define $M = (Q, \Sigma, \delta, q_0, F)$

$Q = \{(q, q') \mid q \in Q_1 \text{ and } q' \in Q_2\} = Q_1 \times Q_2$

$q_0 = (q_1, q_2)$

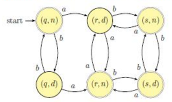
$F = \{(q, q') \mid \overset{q \in Q}{q \in F_1} \text{ or } \overset{q' \in Q_2}{q' \in F_2}\}$
$= F_1 \times Q_2 \cup Q_1 \times F_2$

and $\delta : Q \times \Sigma \to Q$ is defined by

$\delta(((q, q'), x)) = (\delta_1((q, x)), \delta_2((q', x)))$

where $q \in Q_1$ $q' \in Q_2$ $x \in \Sigma$

Example: When $A_1 = \{w \mid w \text{ has an } a \text{ and ends in } b\}$ and $A_2 = \{w \mid w \text{ is of even length}\}$.



$$\begin{tikzcd}
\text{start} \arrow[r] & (q,n) \arrow[r, "a"] \arrow[d, bend left, "b"] & (r,d) \arrow[r, bend left, "b"] \arrow[d, bend left, "a"] & (s,n) \arrow[l, bend left, "a"] \arrow[d, bend left, "b"] \\
& (q,d) \arrow[u, bend left, "b"] \arrow[r, "a"] & (r,n) \arrow[u, bend left, "a"] \arrow[r, bend left, "b"] & (s,d) \arrow[l, bend left, "a"] \arrow[u, bend left, "b"]
\end{tikzcd}$$

DFA with language
$A_1 \cup A_2$.

Suppose A_1, A_2 are languages over an alphabet Σ. **Claim:** if there is a DFA M_1 such that $L(M_1) = A_1$ and DFA M_2 such that $L(M_2) = A_2$, then there is another DFA, let's call it M, such that $L(M) = A_1 \cap A_2$.
Footnote to Sipser Theorem 1.25, page 46

Proof idea: Same construction, change F to $F_1 \wedge F_2$

Formal construction:

copy
paste for
Q, Σ, δ, q_0
from previous construction

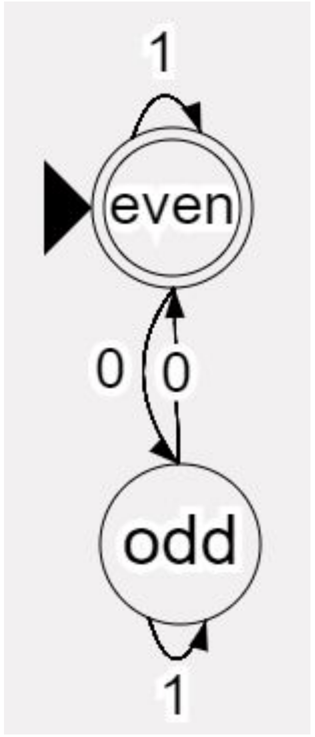
Week 2, Page 7

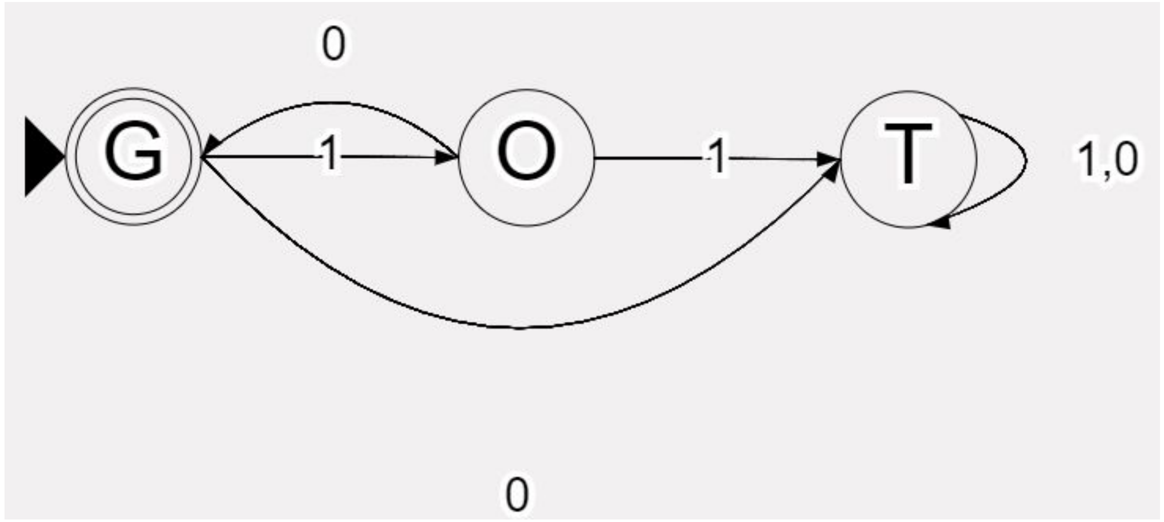
Let us develop some informal intuition !

$L(A_1)$: Set of all strings over Σ containing even number of 0’s

$L(A_2)$: Set of all strings containing non negative integer repeats of 10

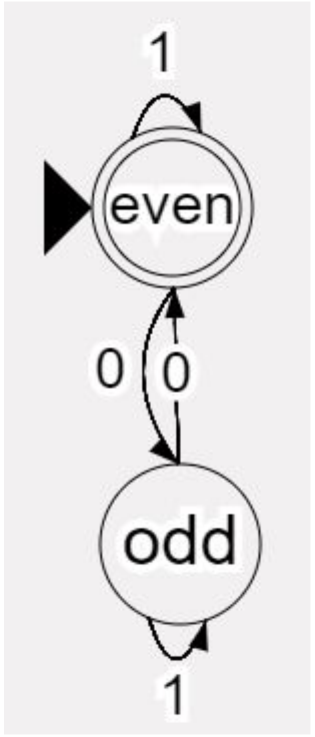
A_1 (L) and A_2 (R)



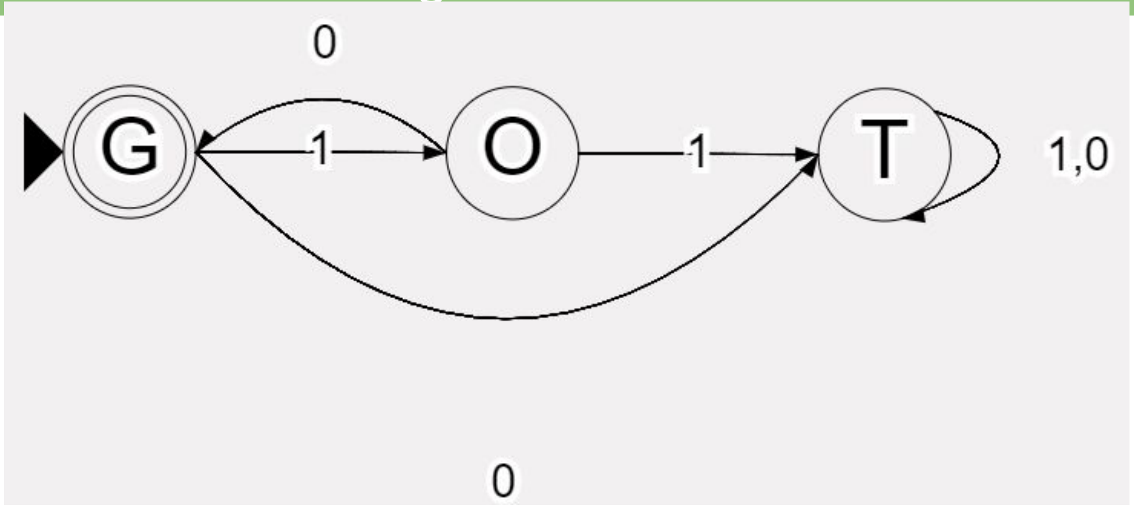


G - (G)ood to go !
O - I read a (O)ne !
T - (T)rapped - no returns !

A_1 (L) and A_2 (R)

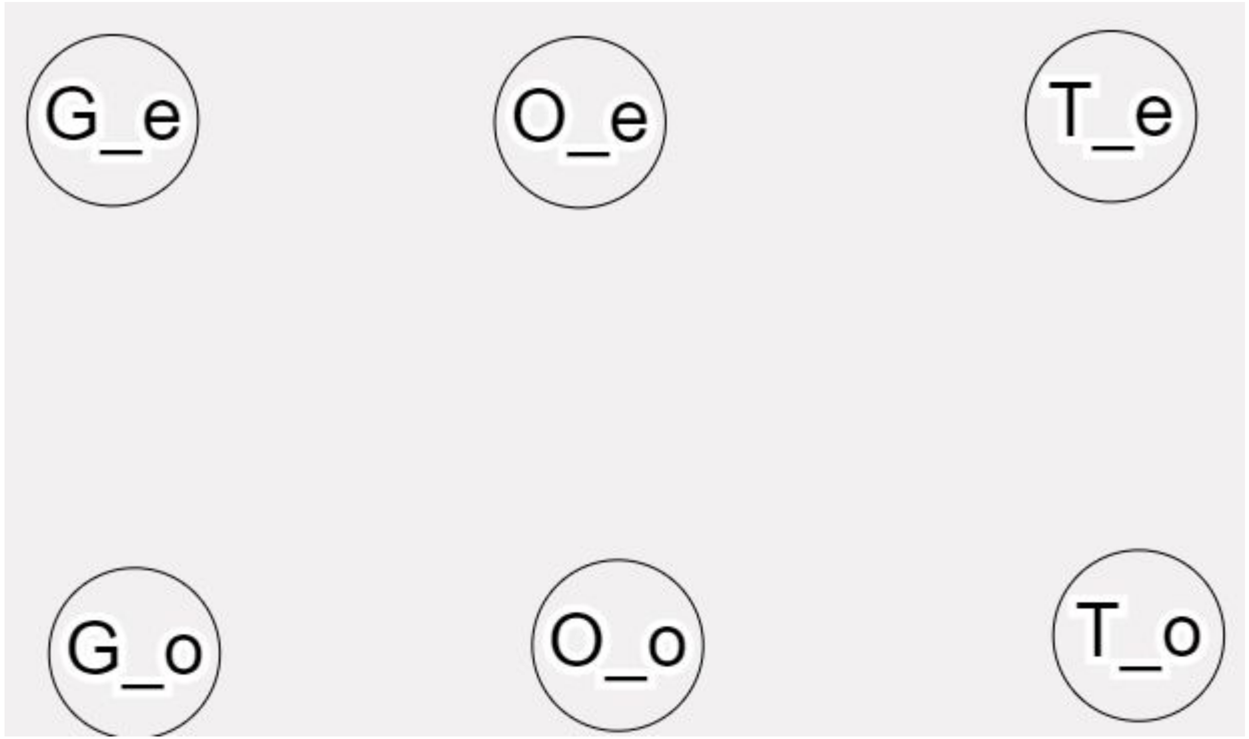


You dont have to actually label your states like this, but it is good to have an idea what each state indicates, especially when you are drawing out smaller state diagrams like these !

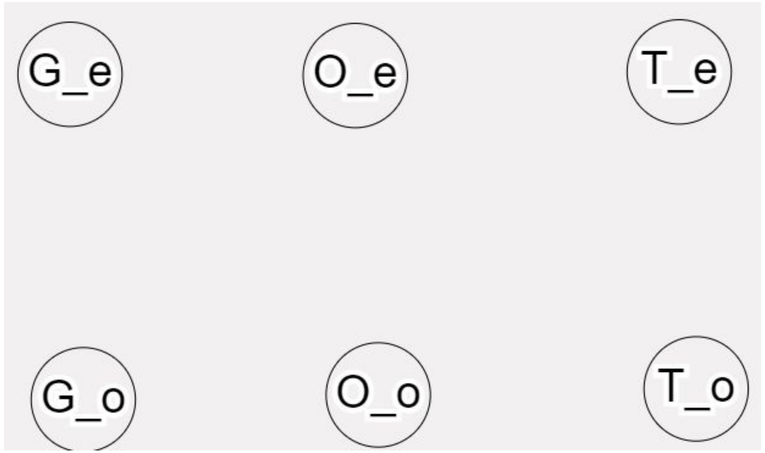


G - (G)ood to go !
O - I read a (O)ne from G !
T - (T)rapped - no returns !

1: Identify states (Q)



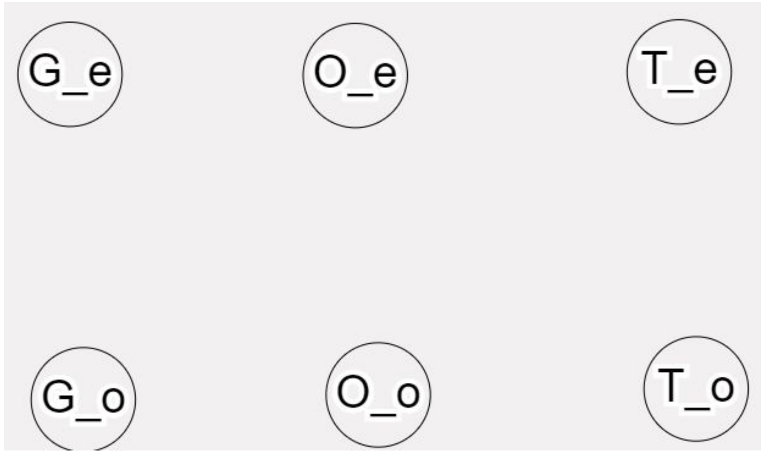
1: Identify states (Q)



Think and answer :

- What does G_e represent ?
- What about T_o ?
- What strings will end at state G_o?
- What strings will end at state O_0
- What about O_e ?

1: Identify states (Q)



Think and answer :

- What strings will end at state G_e
- What strings will end at state T_o ?
- What strings will end at state G_o?
- What strings will end at state O_0
- What strings will end at state O_e ?

1: Identify states (Q)

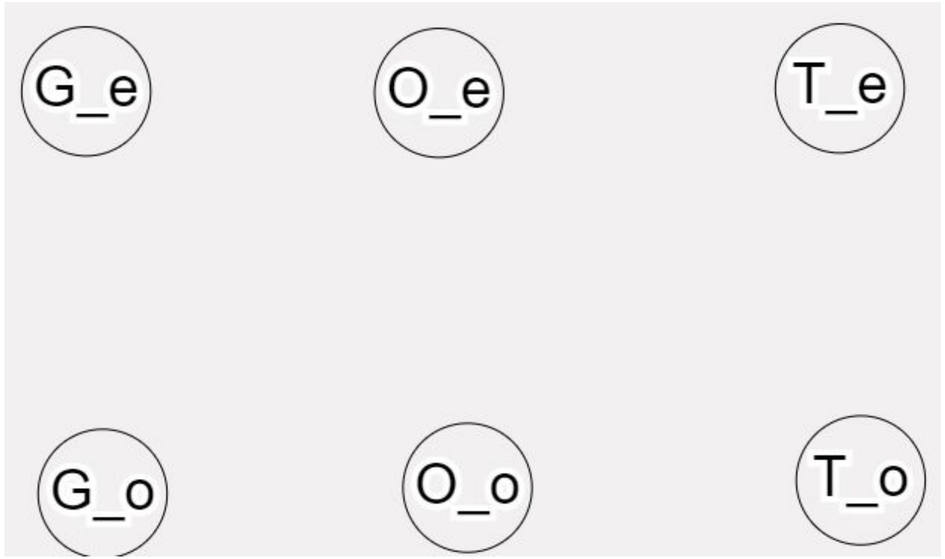
Think and answer :

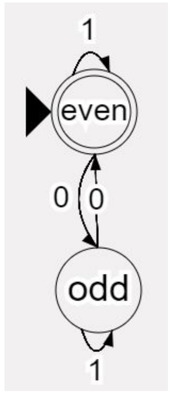
- What strings will end at state G_e
- What strings will end at state T_o ?
- What strings will end at state G_o?
- What strings will end at state O_0
- What strings will end at state O_e ?

Non exhaustive examples:

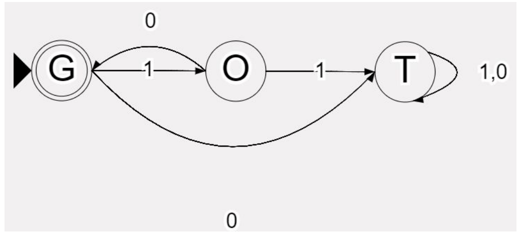
- 10101010, 1010, ε
- 000, 0, 010011
- 10, 101010
- 101, 1010101
- 1, 10101

2: Identify q_0



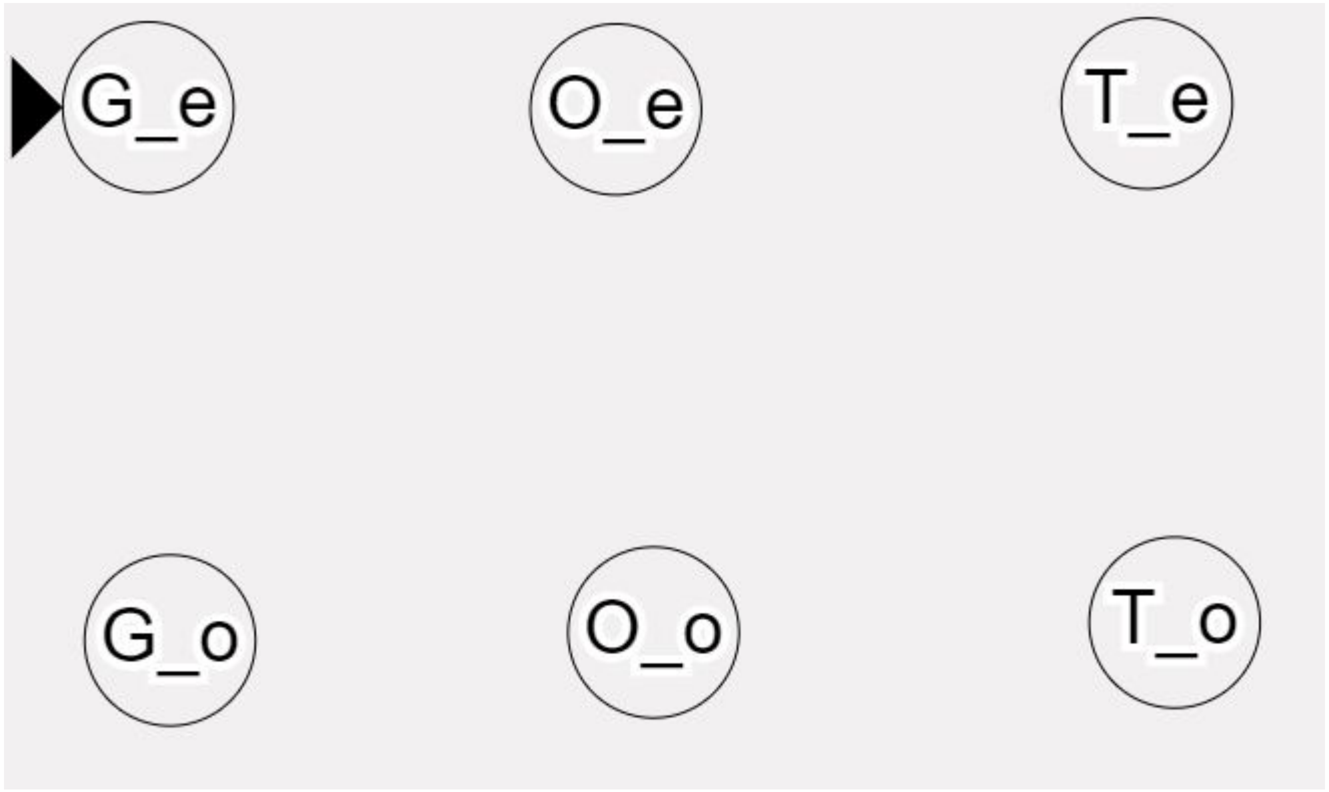


A_1

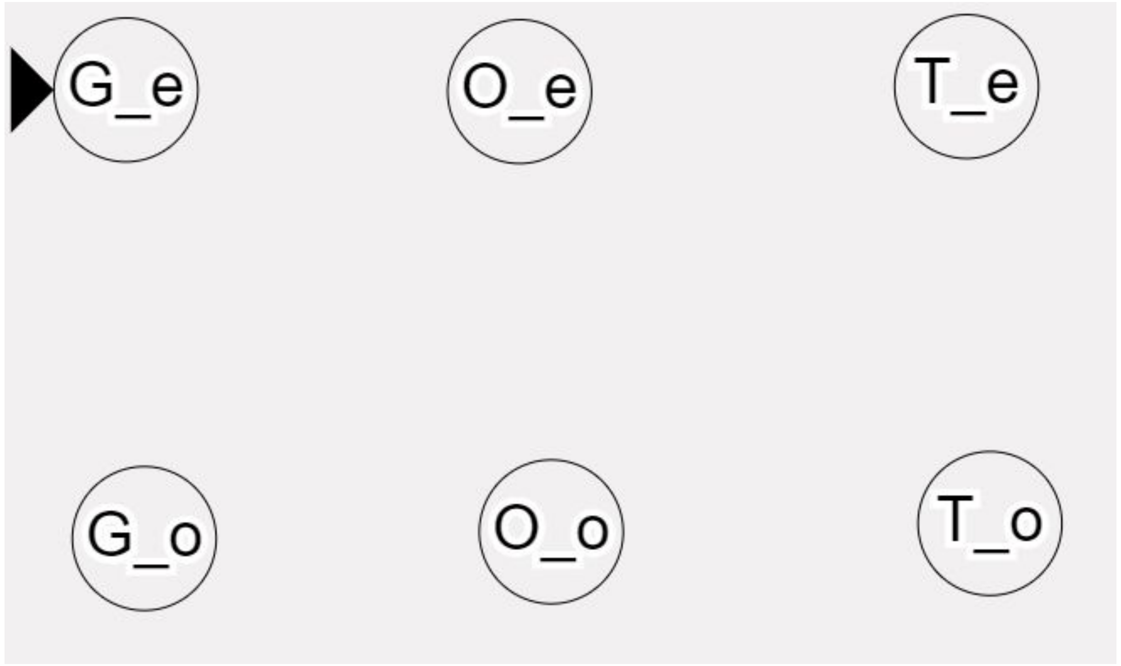


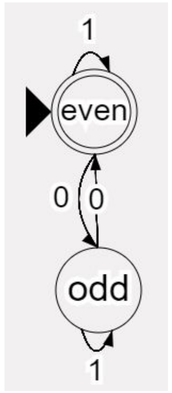
A_2

2: Identify q_0

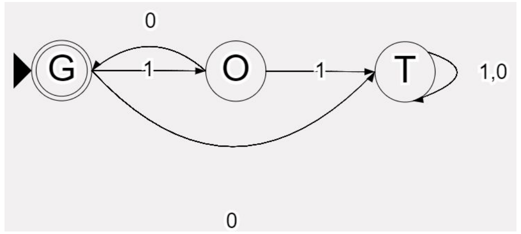


3: Identify δ



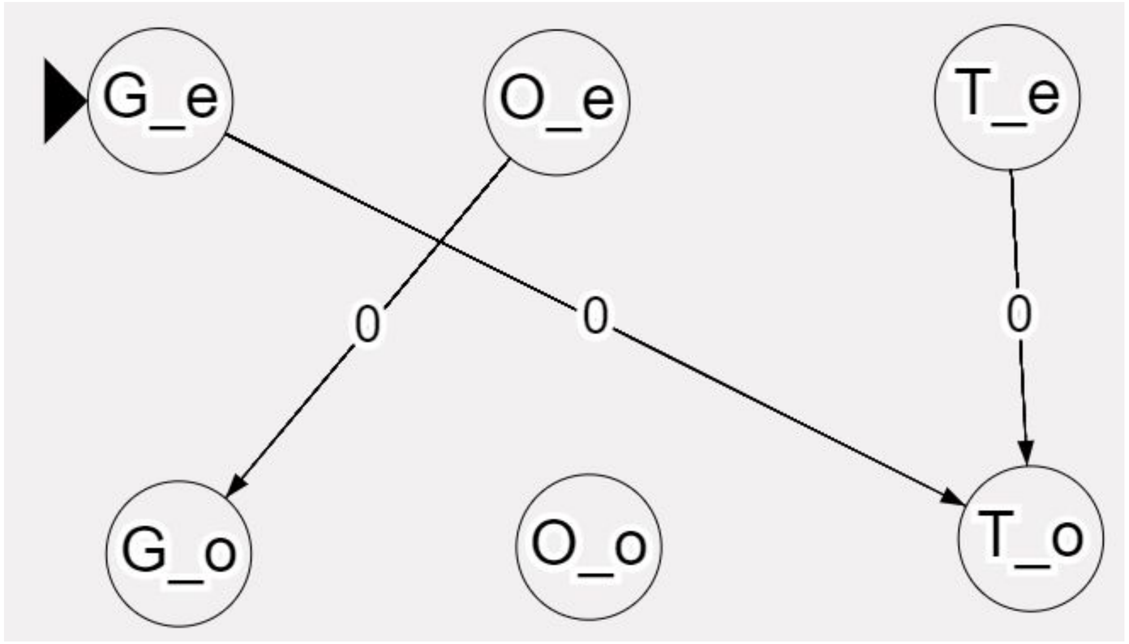


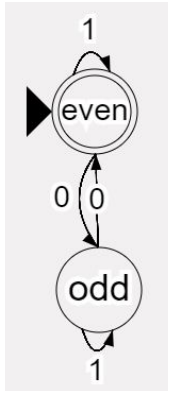
A_1



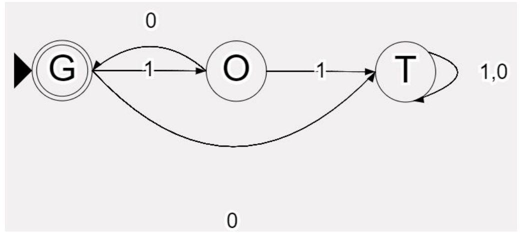
A_2

3: Identify δ



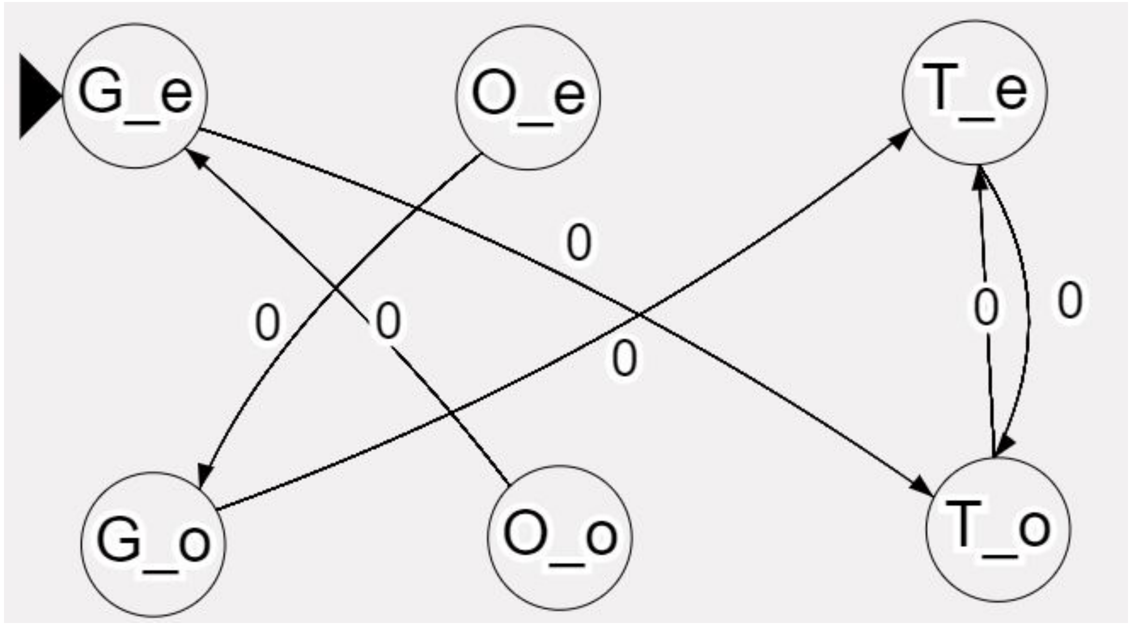


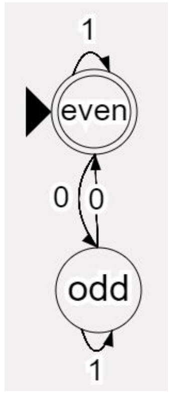
A_1



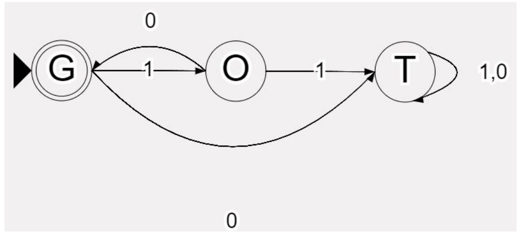
A_2

3: Identify δ



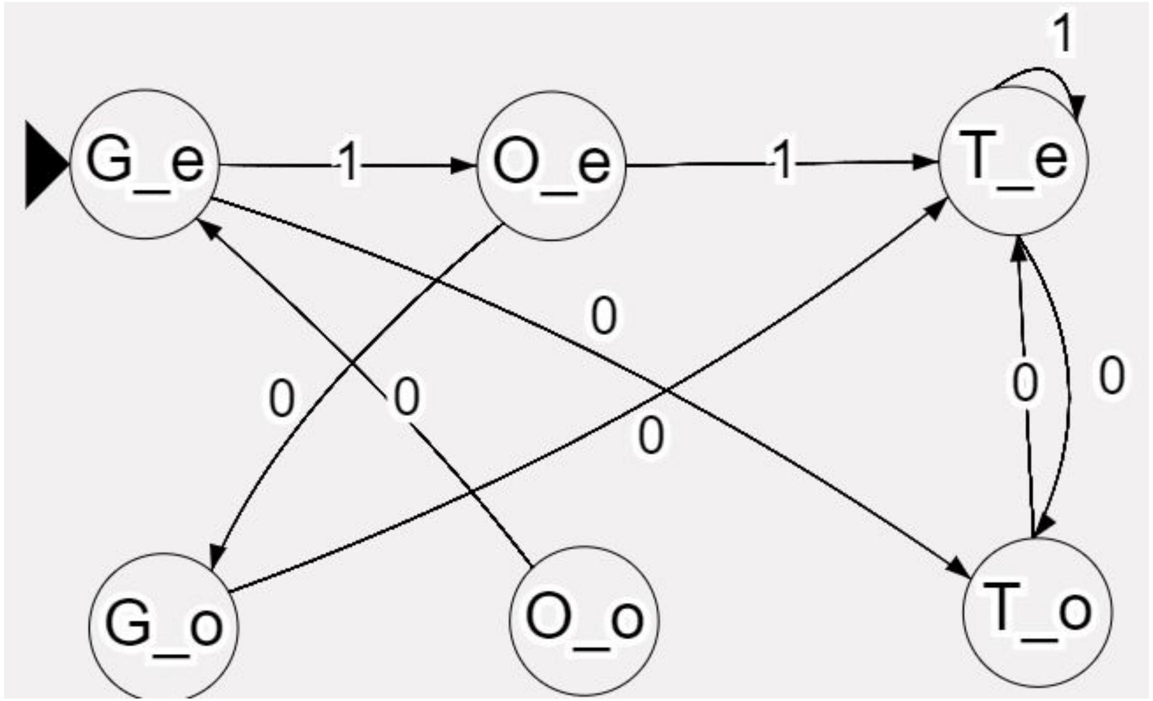


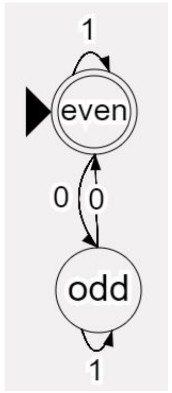
A$_1$



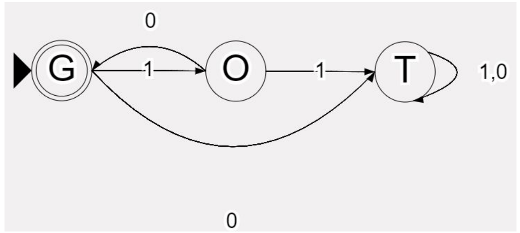
A$_2$

3: Identify δ



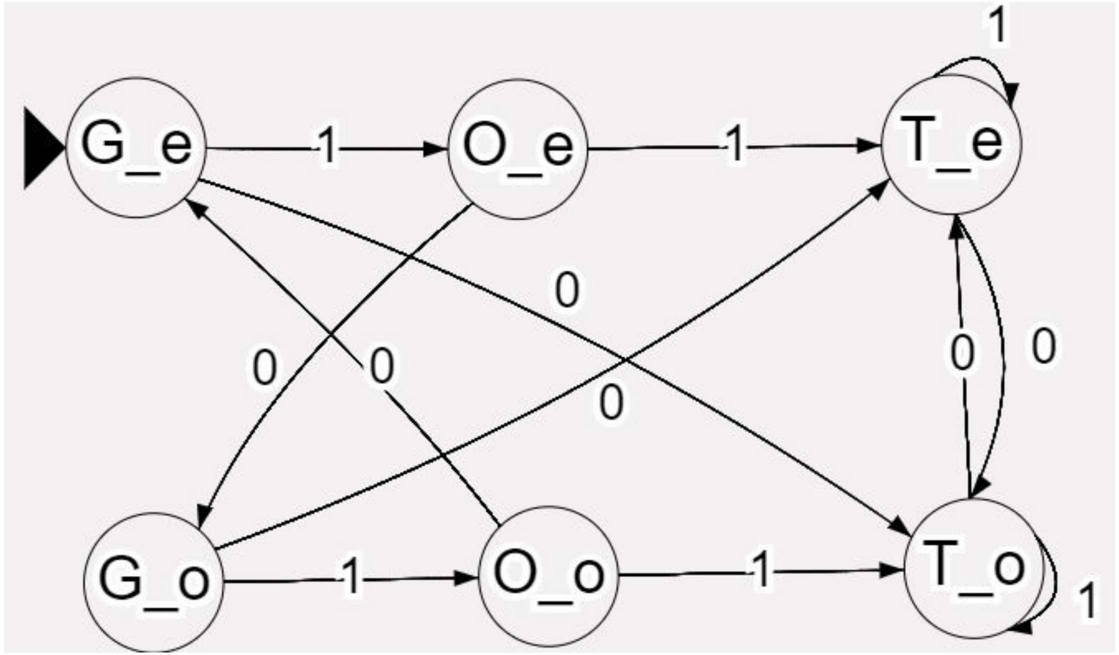


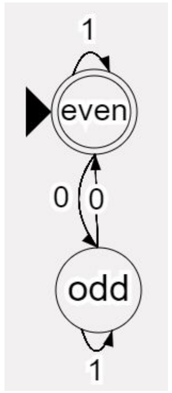
A$_1$



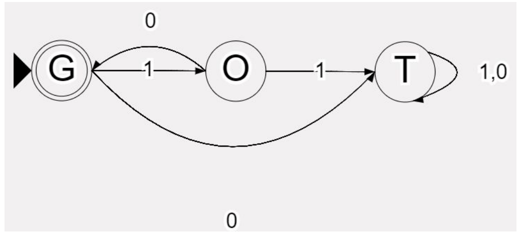
A$_2$

3: Identify δ



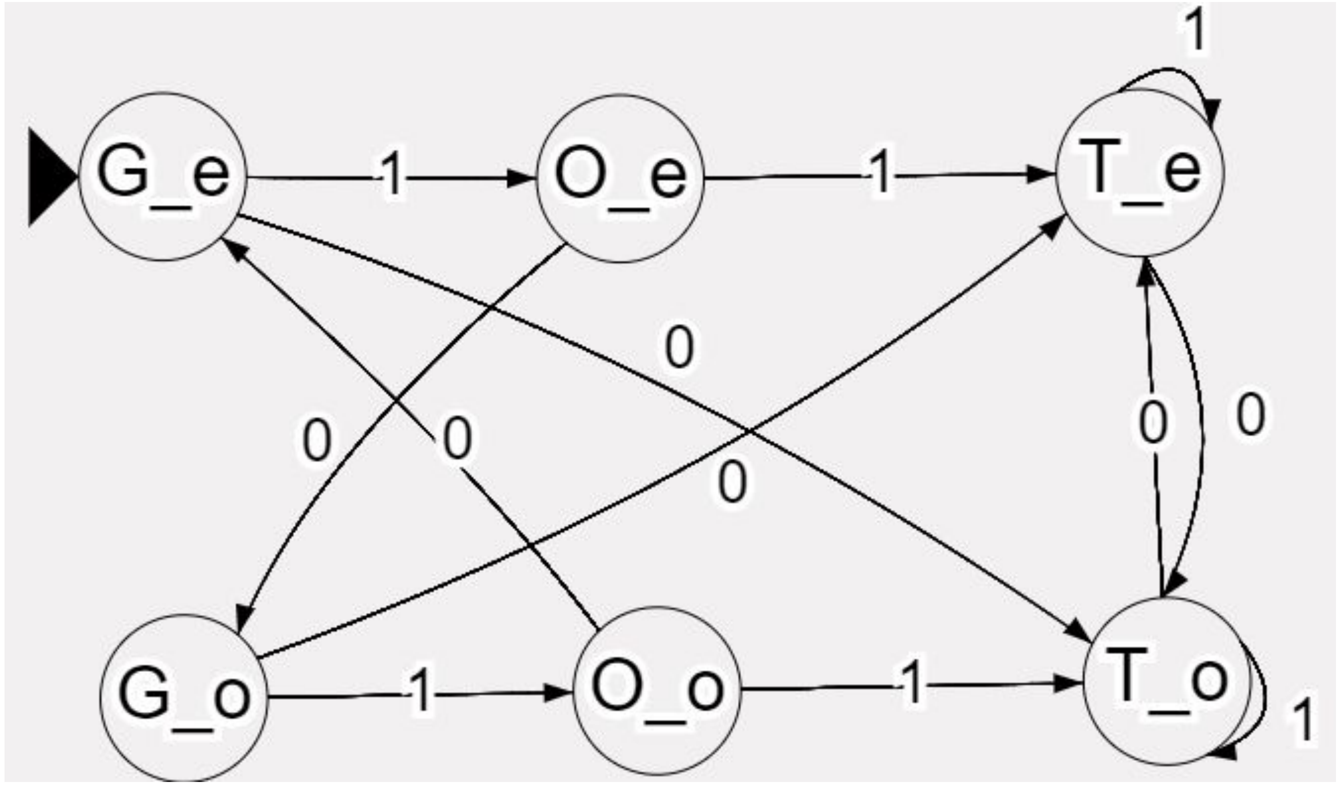


A$_1$

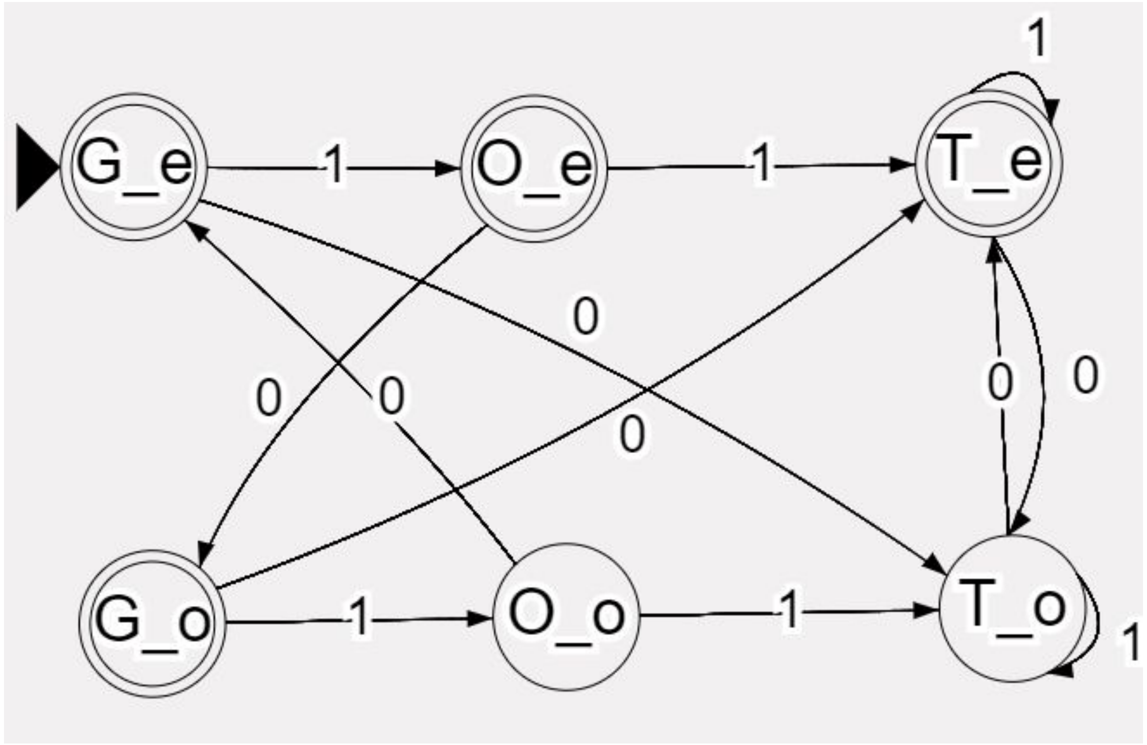


A$_2$

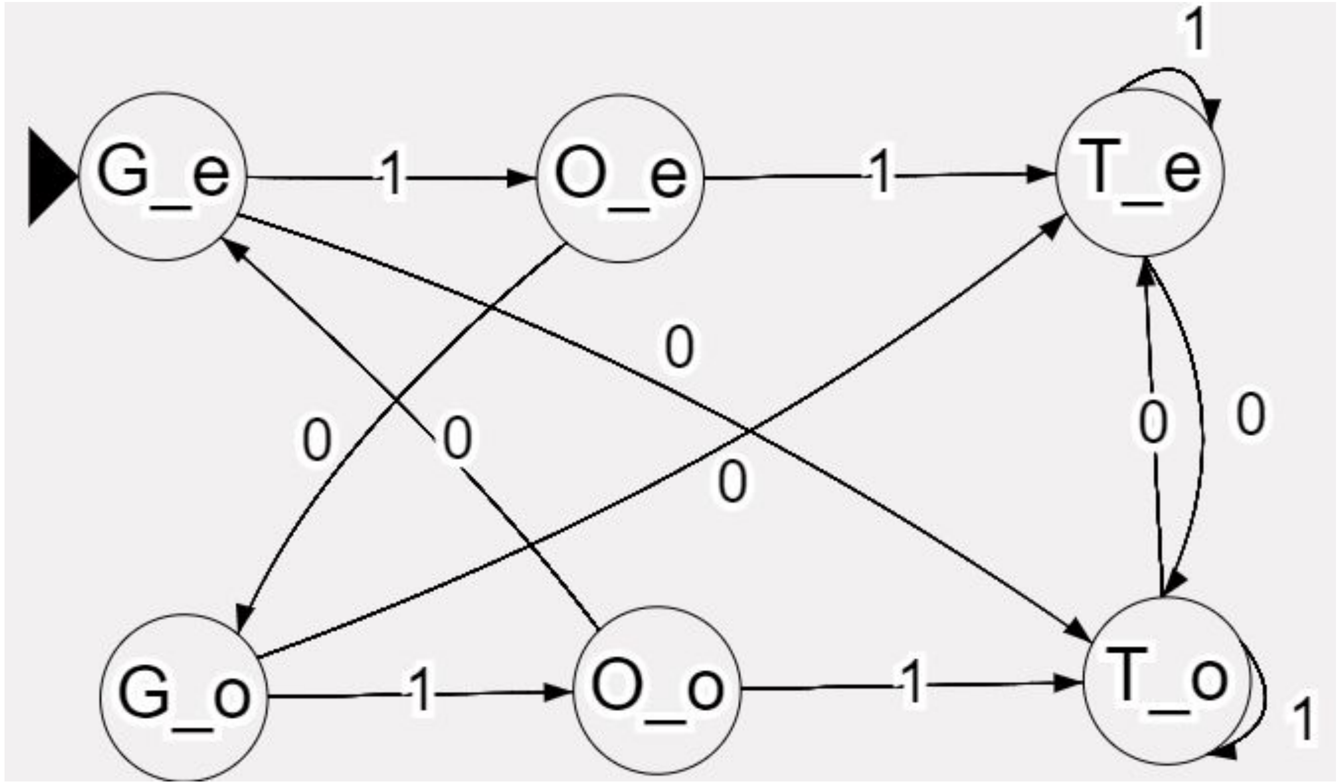
4: Identify F : A_1UA_2



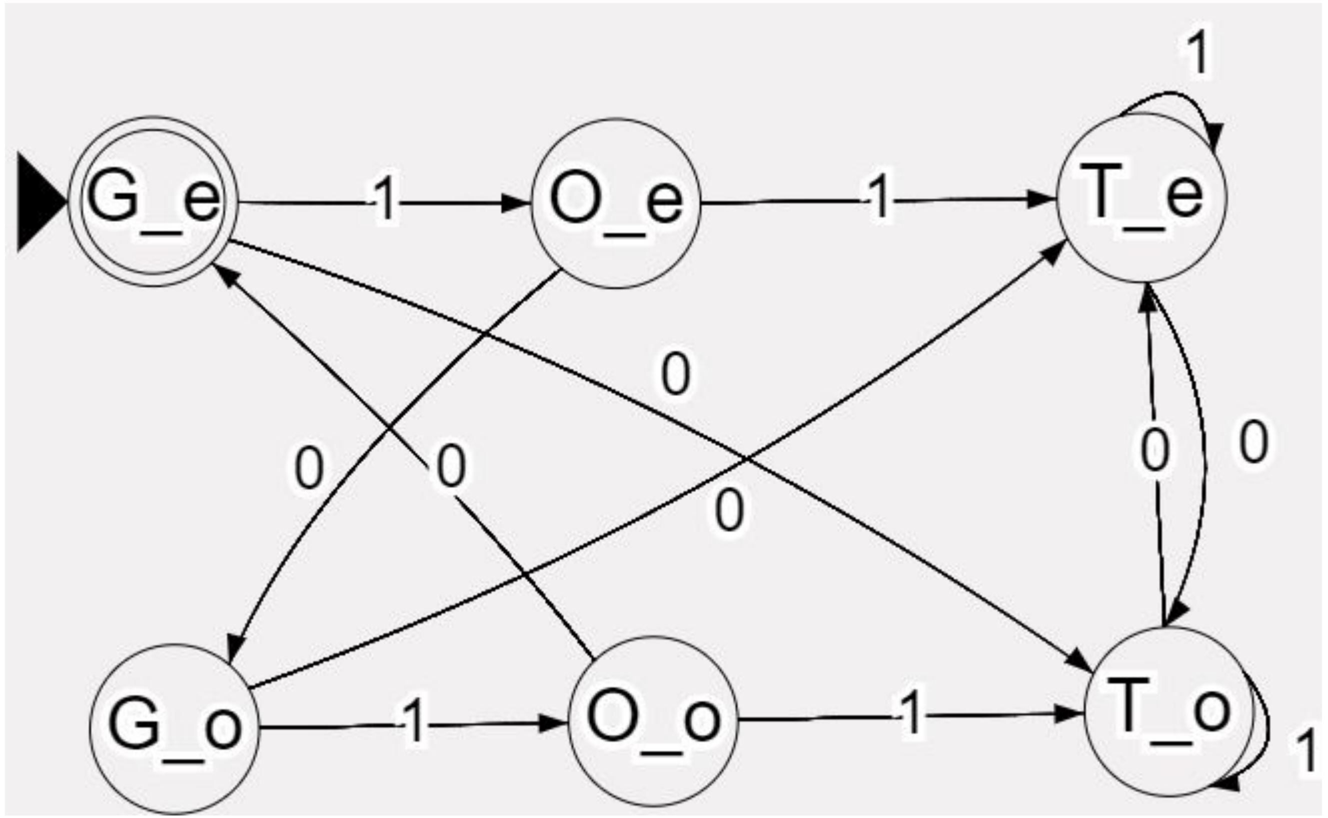
4: Identify F : $A_1 U A_2$



4: Identify F : $A_1 \cap A_2$



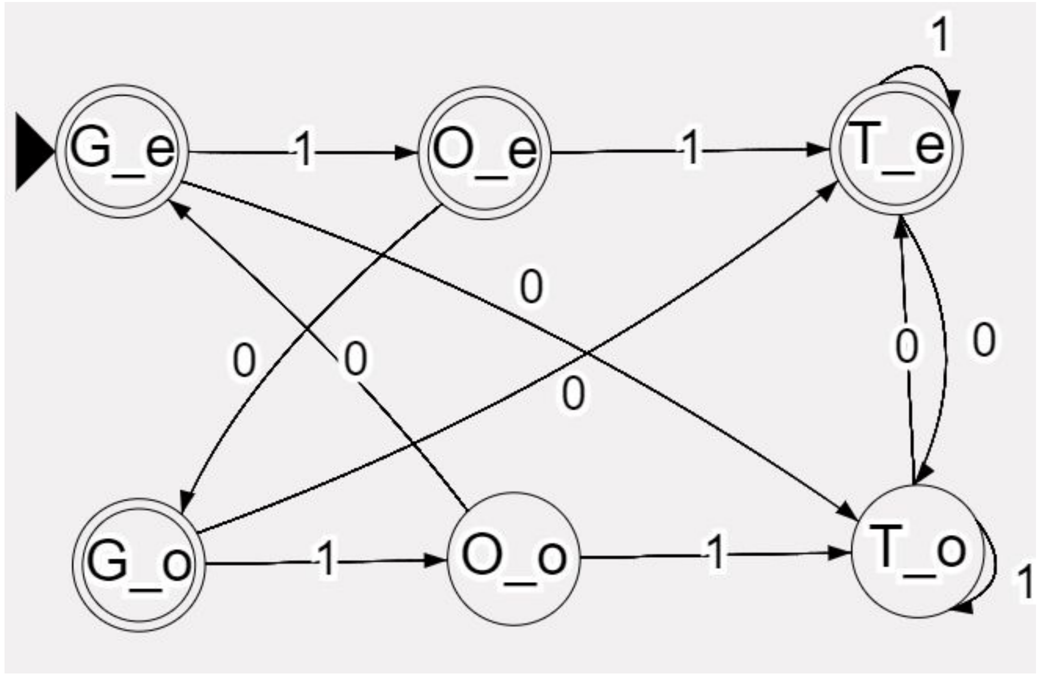
4: Identify F : $A_1 \cap A_2$



Reading strings over this automaton

Think and answer :

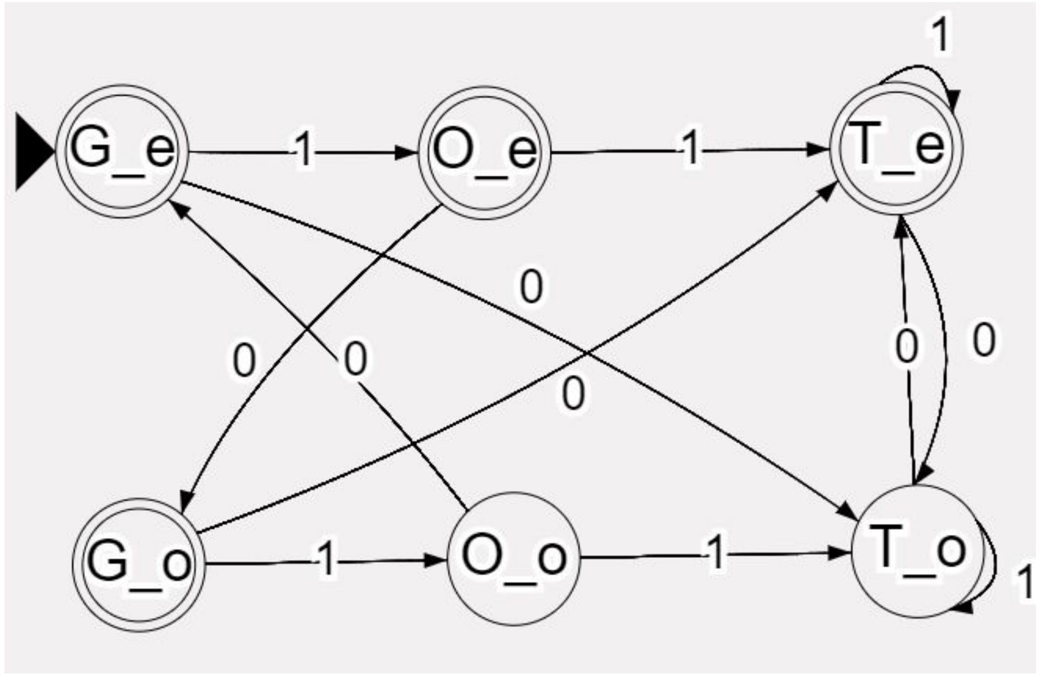
- What strings will end at state G_e
- What strings will end at state T_o ?
- What strings will end at state G_o?
- What strings will end at state O_0
- What strings will end at state O_e ?



Reading strings over this automaton : Trace and verify !

Non exhaustive examples:

- 10101010, 1010, ε
- 000, 0, 010011
- 10, 101010
- 101, 1010101
- 1, 10101



Set operations over L(NFAs)

Languages accepted by NFAs are closed under concatenation

Strategy :

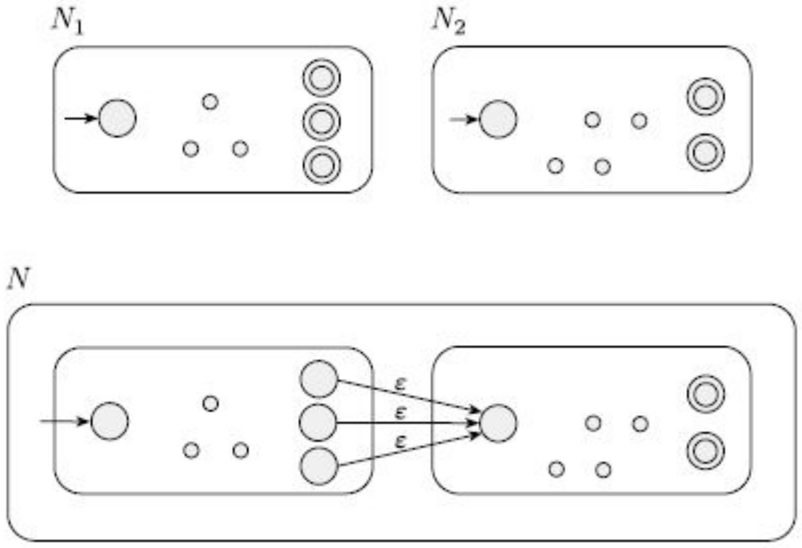


FIGURE 1.48
Construction of N to recognize $A_1 \circ A_2$

Set operations over L(NFAs)

Languages accepted by NFAs are closed under Kleene *

Strategy :

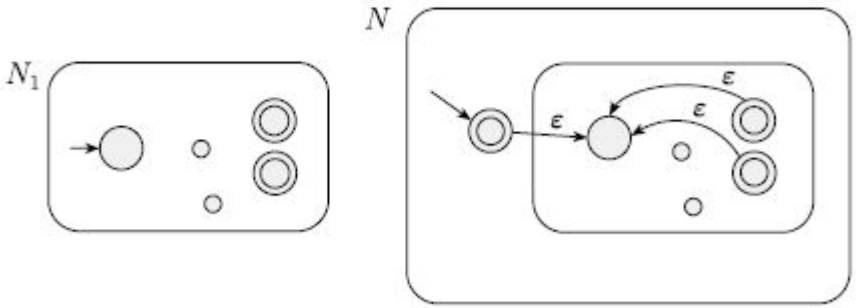


FIGURE 1.50
Construction of N to recognize A^*

DFAs, NFAs and Regular Expressions are equally expressive

Let us start with DFA and NFA equivalence

Alice : “To find an NFA which is equivalent to a given DFA is easy ! All DFAs are NFAs by default”

True or False ?

Let us start with DFA and NFA equivalence

Alice : “To find an NFA which is equivalent to a given DFA is easy ! All DFAs are NFAs by default”

False ! Remember that the 5-tuple formal definition for DFAs and NFAs is *slightly* different. Recall what changes need to be made to quickly “convert” a DFA to an equivalent NFA

Let us start with DFA and NFA equivalence

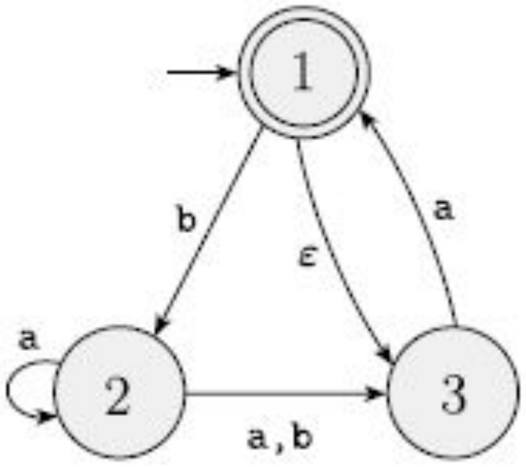
Bob : “To find a DFA which is equivalent to a NFA is slightly harder. I should have paid attention during lecture today and I possibly need to revise the material from Sipser pg 54-58”

True or False ?

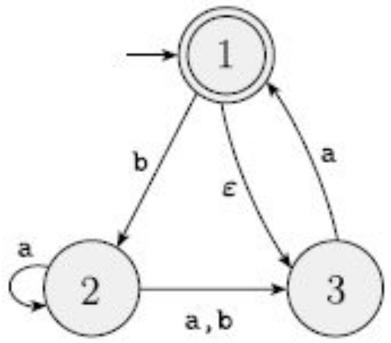
Let us start with DFA and NFA equivalence

General Idea - Create “Macro States” for the DFA that keeps track of combinations of states of a given NFA

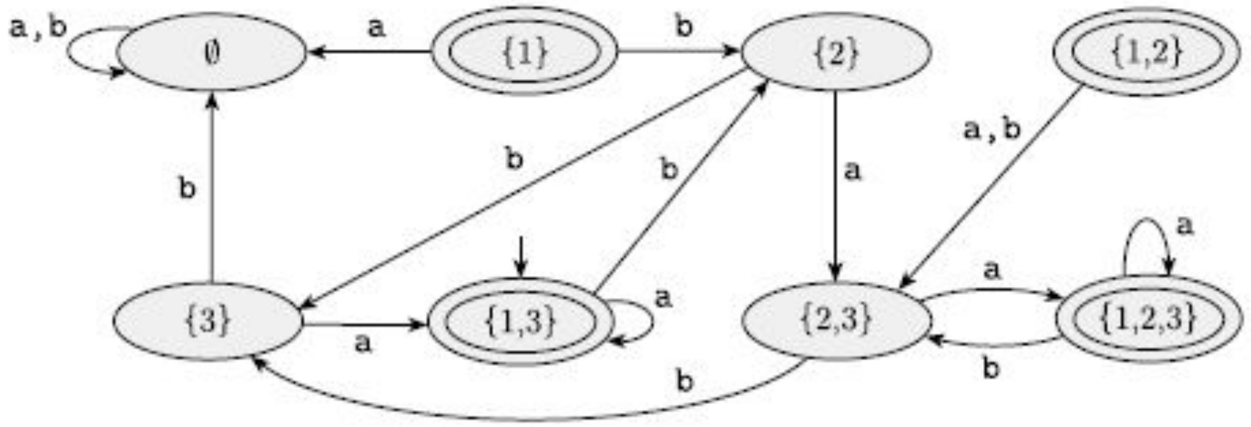
Sipser pg 57



Sipser pg 57



NFA N



DFA D recognizing L(N)

Now, NFA and RegEx equivalence

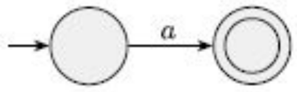
Regex to NFA

Recall :

1. $R = a$, where $a \in \Sigma$
2. $R = \varepsilon$
3. $R = \emptyset$
4. $R = (R_1 \cup R_2)$, where R_1, R_2 are themselves regular expressions
5. $R = (R_1 \circ R_2)$, where R_1, R_2 are themselves regular expressions
6. (R_1^*), where R_1 is a regular expression.

Regex to NFA

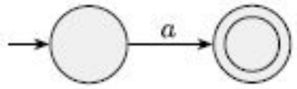
Recall :

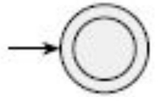
1. $R = a$, where $a \in \Sigma$ 

2. $R = \varepsilon$
3. $R = \emptyset$
4. $R = (R_1 \cup R_2)$, where R_1, R_2 are themselves regular expressions
5. $R = (R_1 \circ R_2)$, where R_1, R_2 are themselves regular expressions
6. (R_1^*), where R_1 is a regular expression.

Regex to NFA

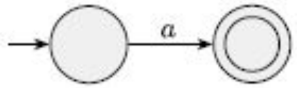
Recall :

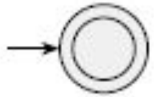
1. $R = a$, where $a \in \Sigma$ 

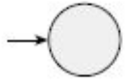
2. $R = \varepsilon$ 
3. $R = \emptyset$
4. $R = (R_1 \cup R_2)$, where R_1, R_2 are themselves regular expressions
5. $R = (R_1 \circ R_2)$, where R_1, R_2 are themselves regular expressions
6. (R_1^*), where R_1 is a regular expression.

Regex to NFA

Recall :

1. $R = a$, where $a \in \Sigma$ 

2. $R = \varepsilon$ 

3. $R = \emptyset$ 

4. $R = (R_1 \cup R_2)$, where R_1, R_2 are themselves regular expressions

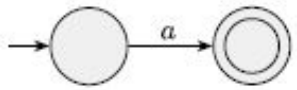
5. $R = (R_1 \circ R_2)$, where R_1, R_2 are themselves regular expressions

6. (R_1^*), where R_1 is a regular expression.

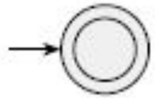
Regex to NFA

Recall :

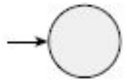
1. $R = a$, where $a \in \Sigma$

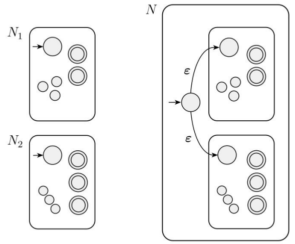


2. $R = \varepsilon$



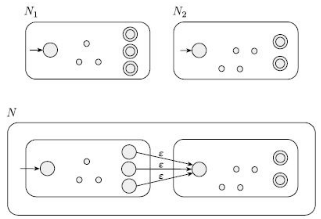
3. $R = \emptyset$



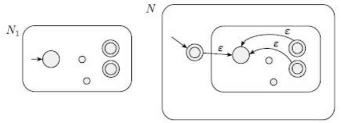


4. $R = (R_1 \cup R_2)$, where R_1, R_2 are themselves regular expressions

5. $R = (R_1 \circ R_2)$, where R_1, R_2 are themselves regular expressions



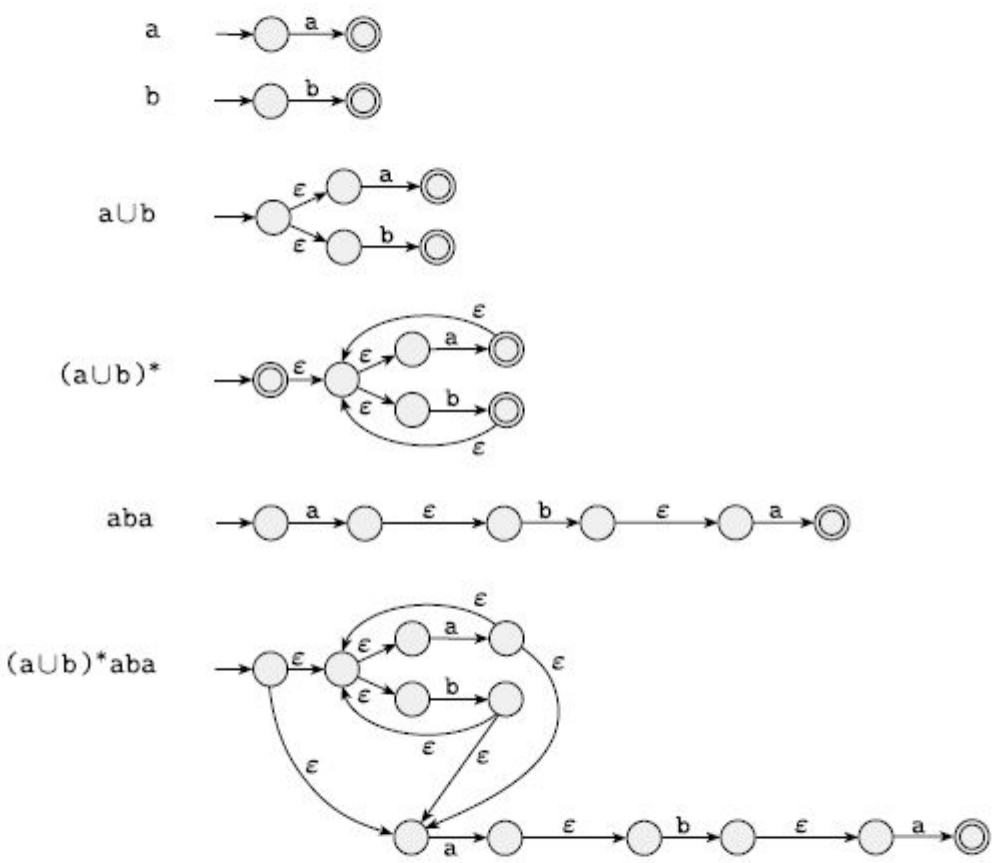
6. (R_1^*), where R_1 is a regular expression.



Practice : (aUb)*aba

Example 1.58 Sipser, pg 69

Practice : (aUb)*aba

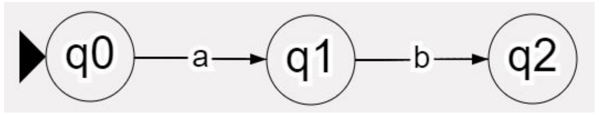


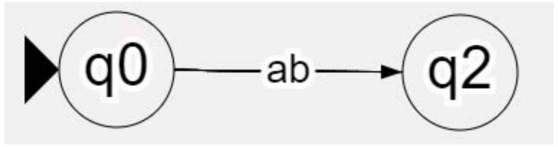
Example 1.58 Sipser, pg 69

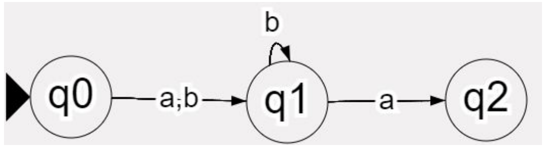
NFA/DFA to Regex

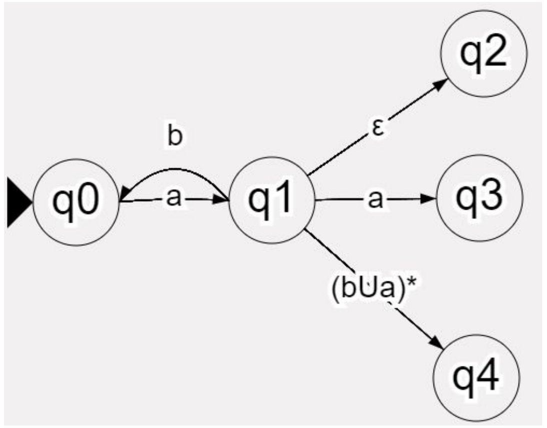
1. Add one extra start and end state respectively, and make requisite connections
2. Prune away states one by one making sure to re-make edge connections such that the state diagram is equivalent to itself prior to pruning. Remade edges can be labelled with regular expressions.
3. Rinse and repeat till you have a single edge between the added start and end state.

Examples of removing states (q1)

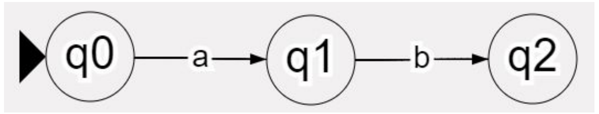


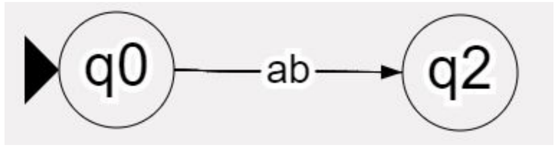


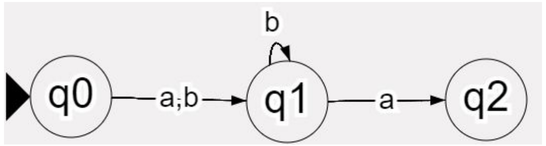


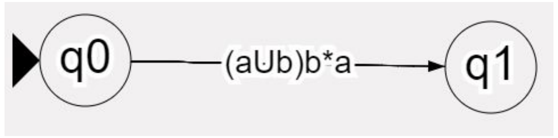


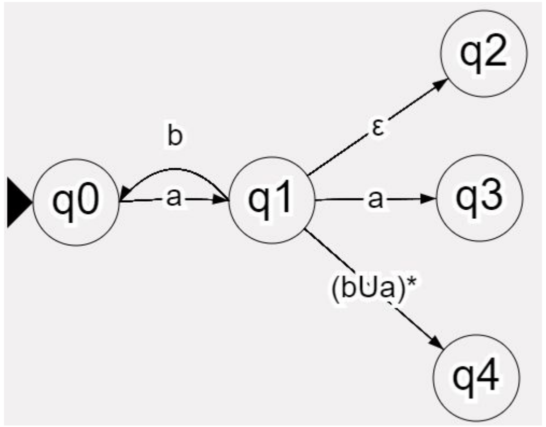
Examples of removing states (q1)



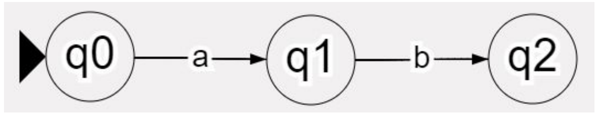


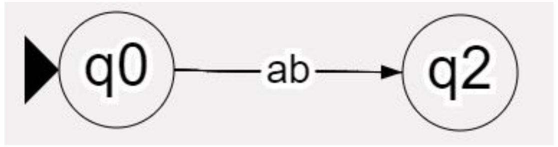


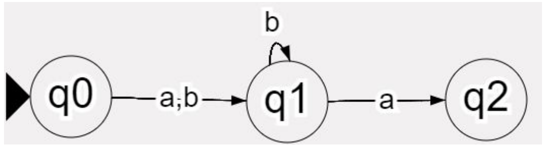


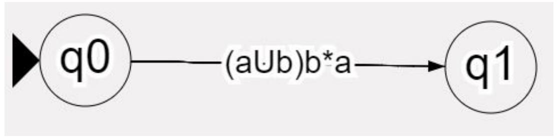


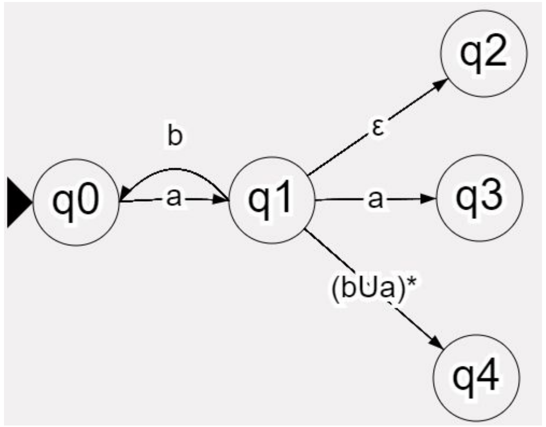
Examples of removing states (q1)

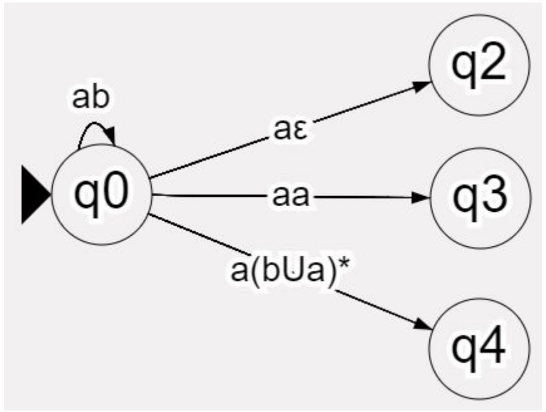





b
q0
a;b
q1
a
q2




q2
b
ε
q0
a
q1
a
q3
(b∪a)*
q4


q2
ab
aε
q0
aa
q3
a(b∪a)*
q4