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 2 due next week on 30th (Thursday) at 5 PM

Current progress - Answer Y/N

1. Given a DFA and a string, I can 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 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

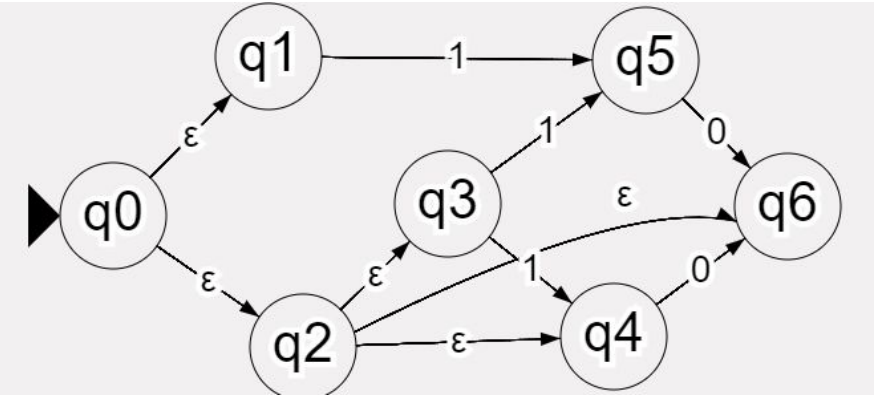
Today's Topics

1. Recap of ε-transitions in an NFA
2. Closure over U, ∩, ̄, * 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

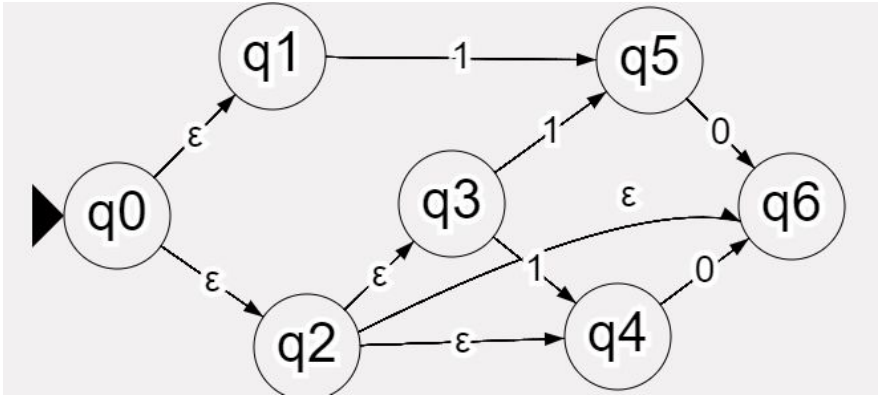
ε-transitions

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

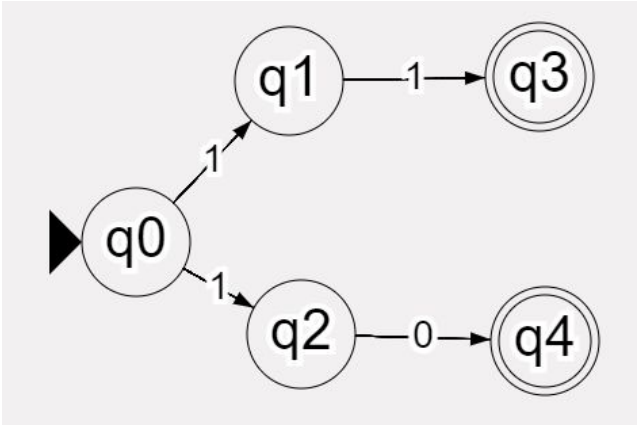


ε-transitions

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...

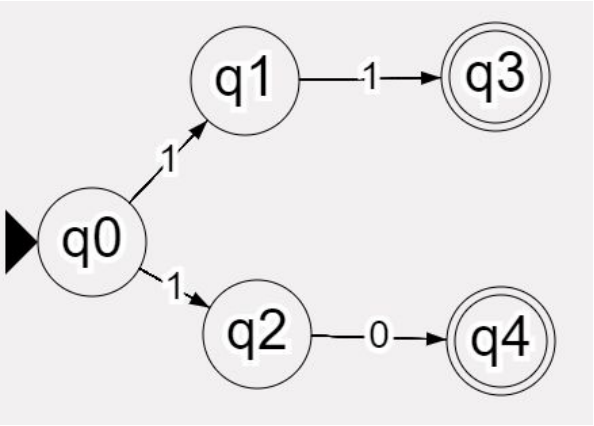


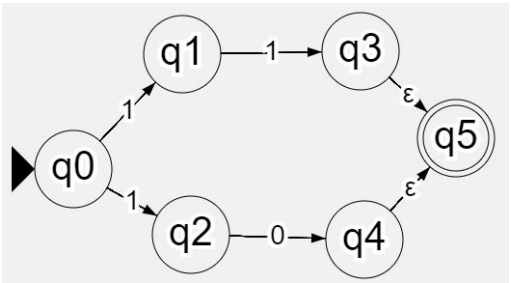
1. Have exactly one accept state
    a. With and without changing Q (the set of states) in the 5-tuple definition
2. Have 5 accept states, $q_3 \notin F$, $q_4 \notin F$

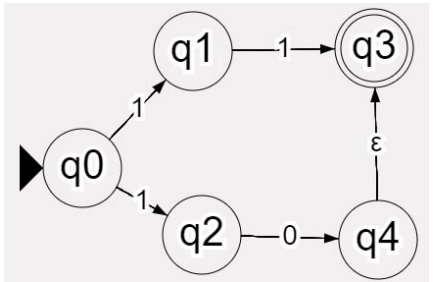
Note - The modified NFA has to recognize the same language !

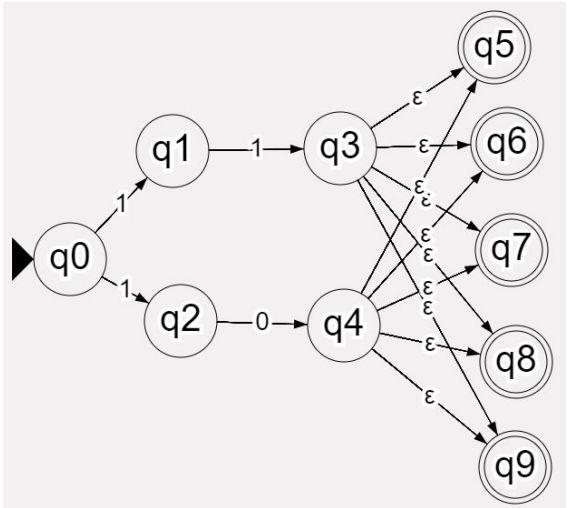
Modify this NFA to...

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









DFAs and NFAs closure over U, * and ○

Closure - What we learnt last week

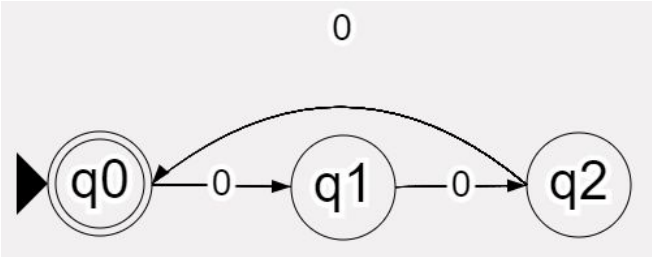
Languages accepted by DFAs are closed under complementation

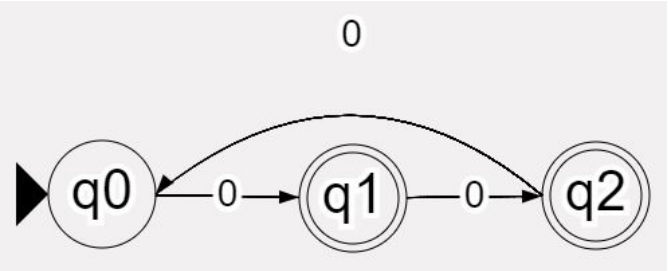
Strategy :

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:

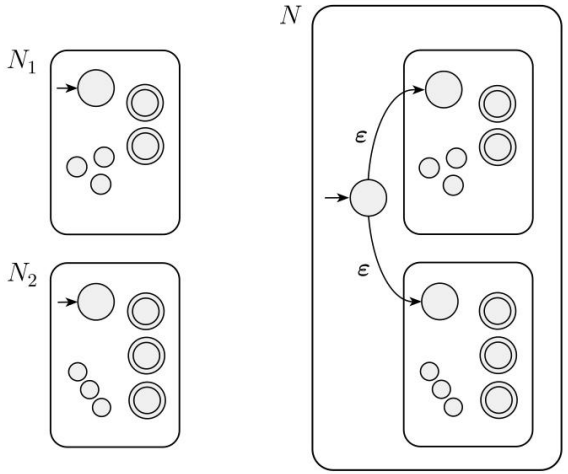


FIGURE **1.46**
Construction of an NFA N to recognize $A_1 \cup A_2$

Closure

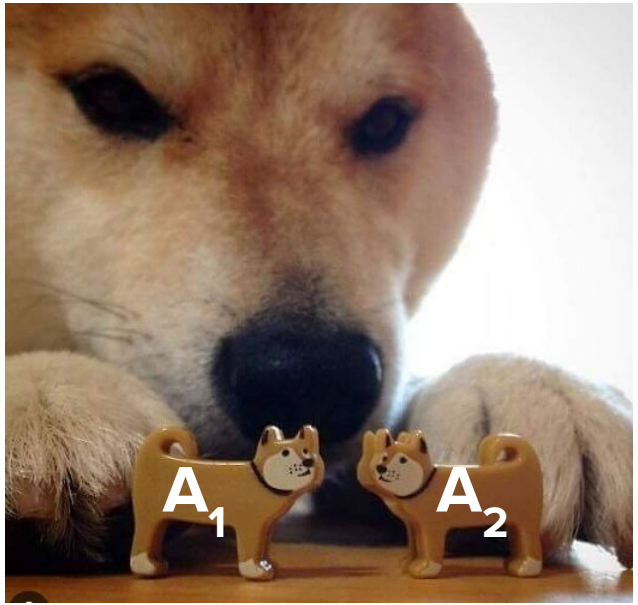
1. **Languages accepted by DFAs are closed under union**
2. **Languages accepted by DFAs are closed under intersection**

Strategy:

Closure

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

Strategy: Parallel Computation



Motivating example : Σ = {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 :

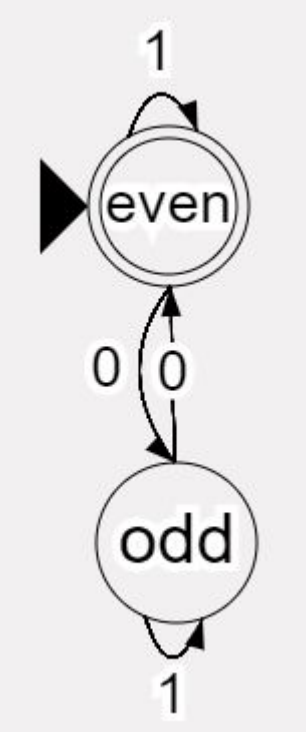
1. $L(A) = L(A_1) \cup L(A_2)$
2. $L(A) = L(A_1) \cap L(A_2)$

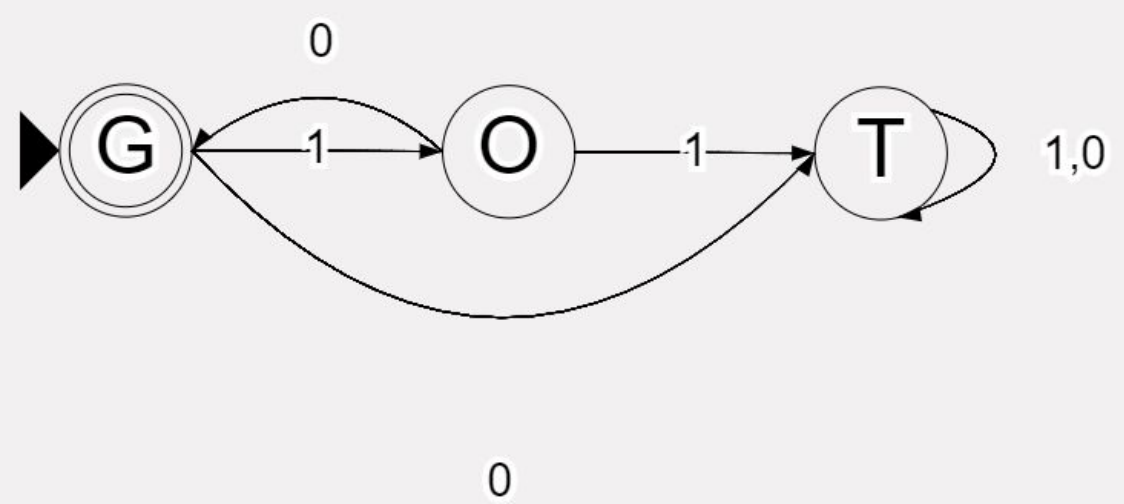
Let us develop some informal intuition !

$L(A_1)$: 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

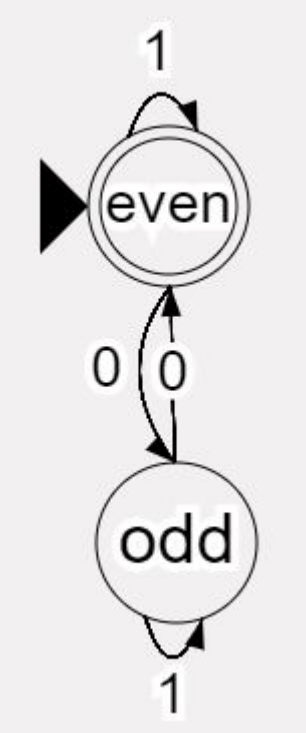
A_1 (L) and A_2 (R)



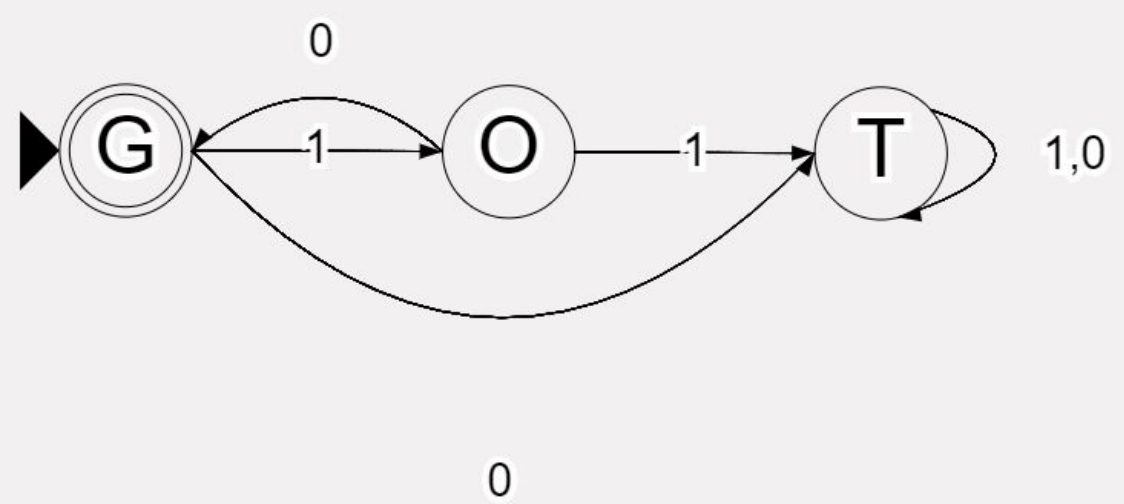


G - (G)ood to go !
O - I read a (O)ne from G !
T - (T)rapped - no returns !

A_1 (L) and A_2 (R)

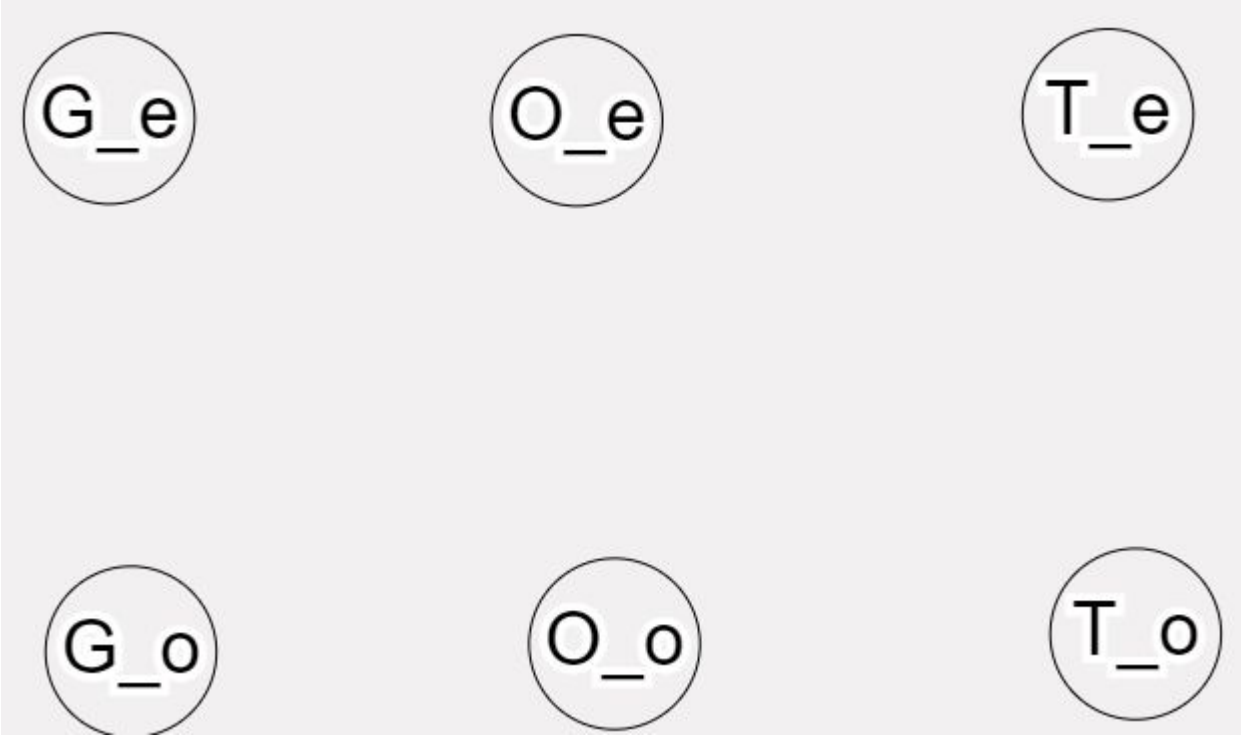


You don't have to actually label your states like this, but it is good to have an idea what each state indicates, especially when you are drawing out smaller state diagrams like these !

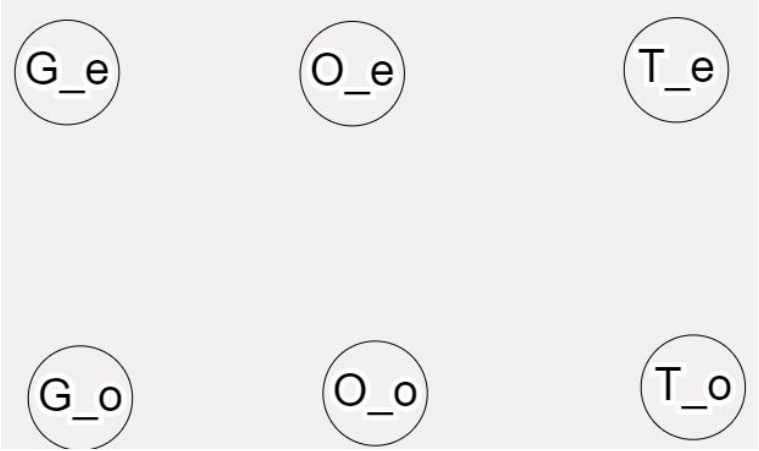


G - (G)ood to go !
O - I read a (O)ne from G !
T - (T)rapped - no returns !

1: Identify states (Q)



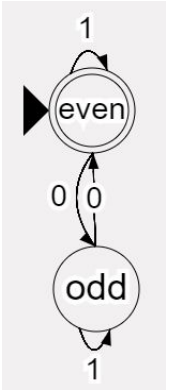
1: Identify states (Q)



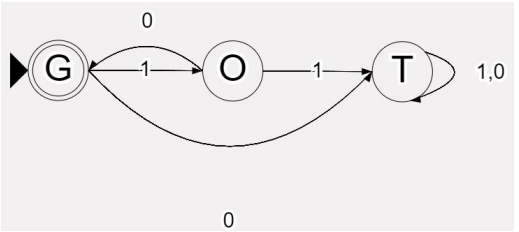
Think and answer :

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

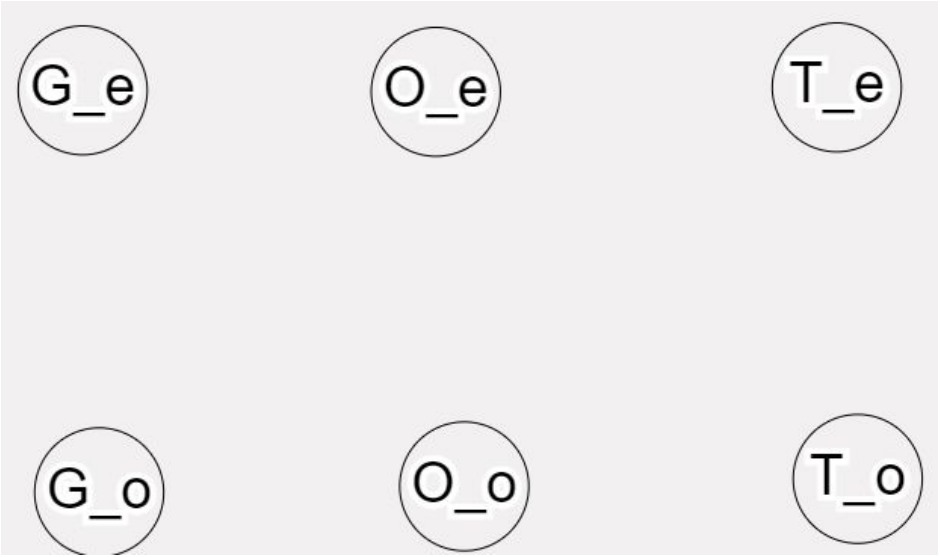
2: Identify q_0



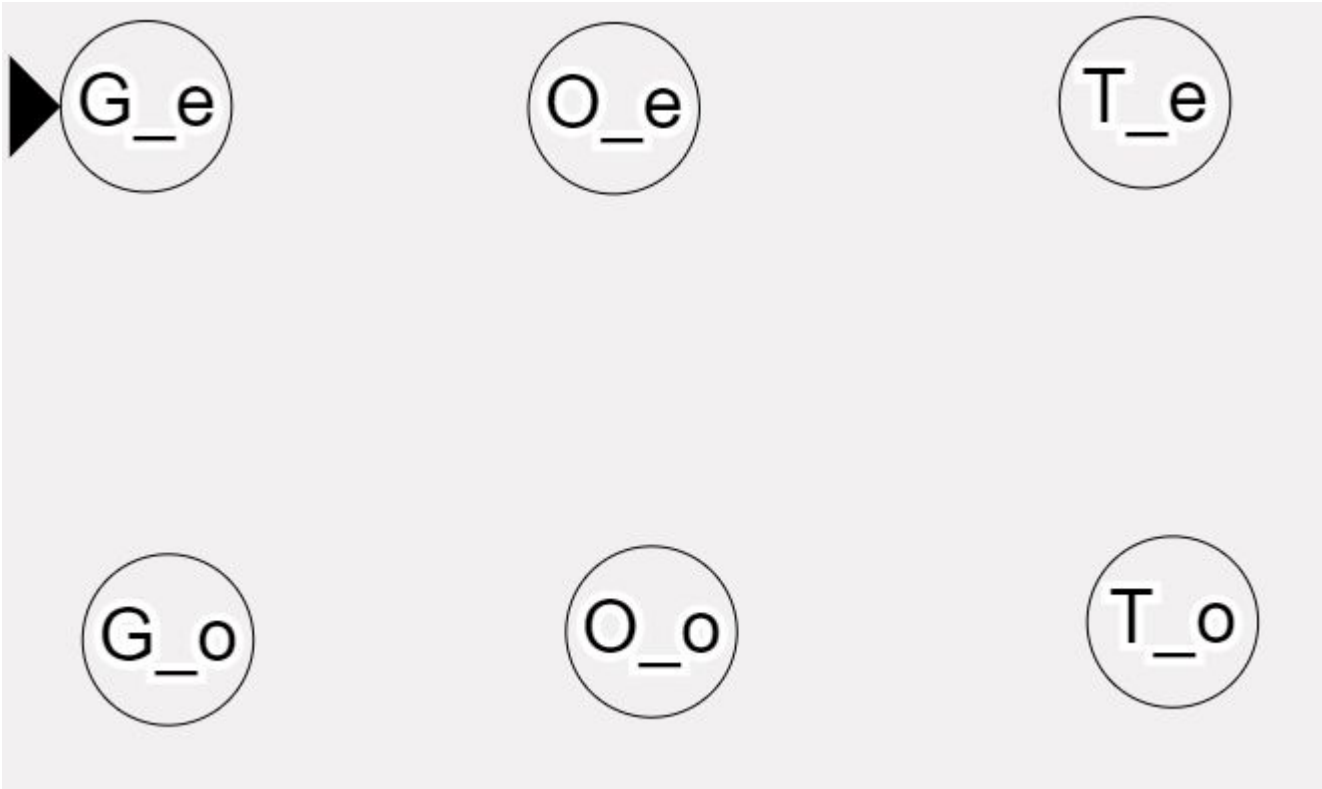
A_1



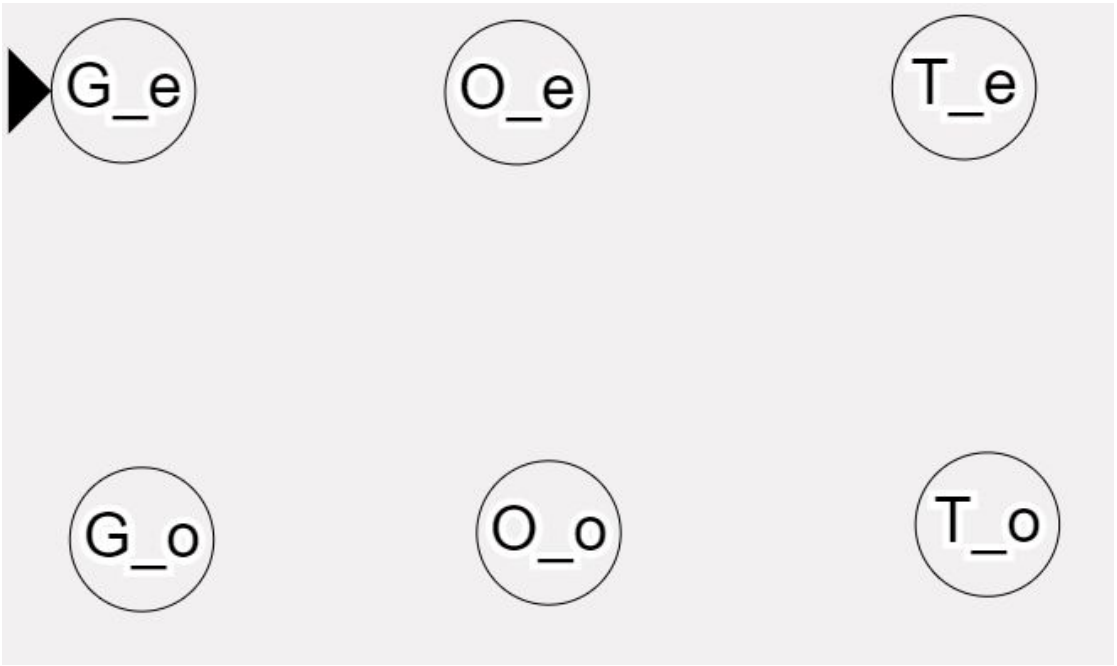
A_2

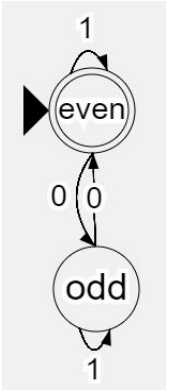


2: Identify q_0

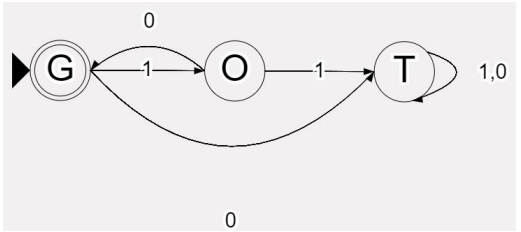


3: Identify δ



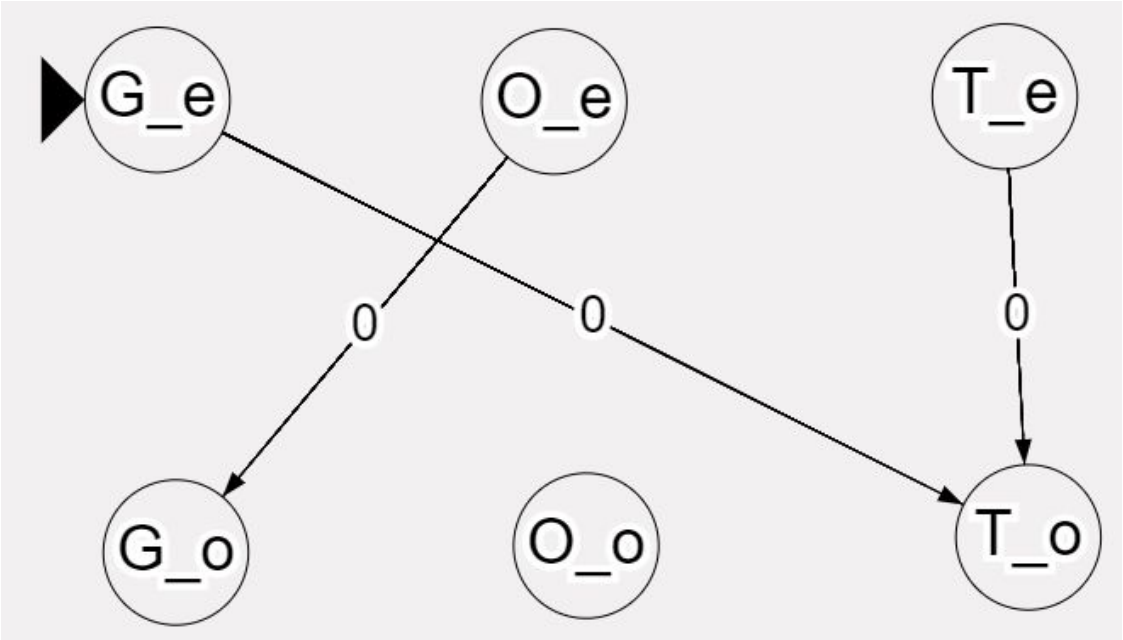


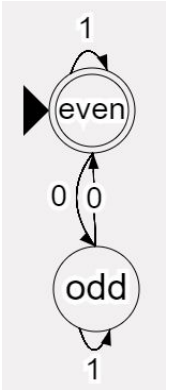
A_1



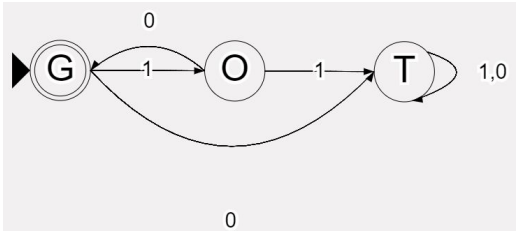
A_2

3: Identify δ



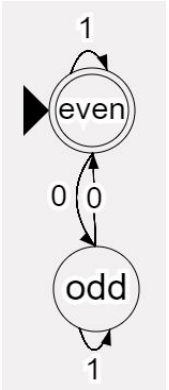


A_1

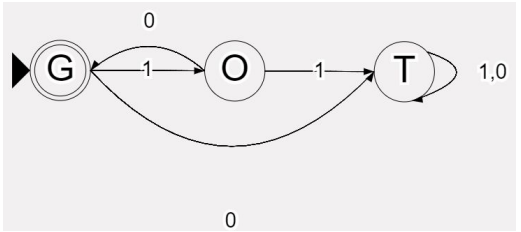


A_2

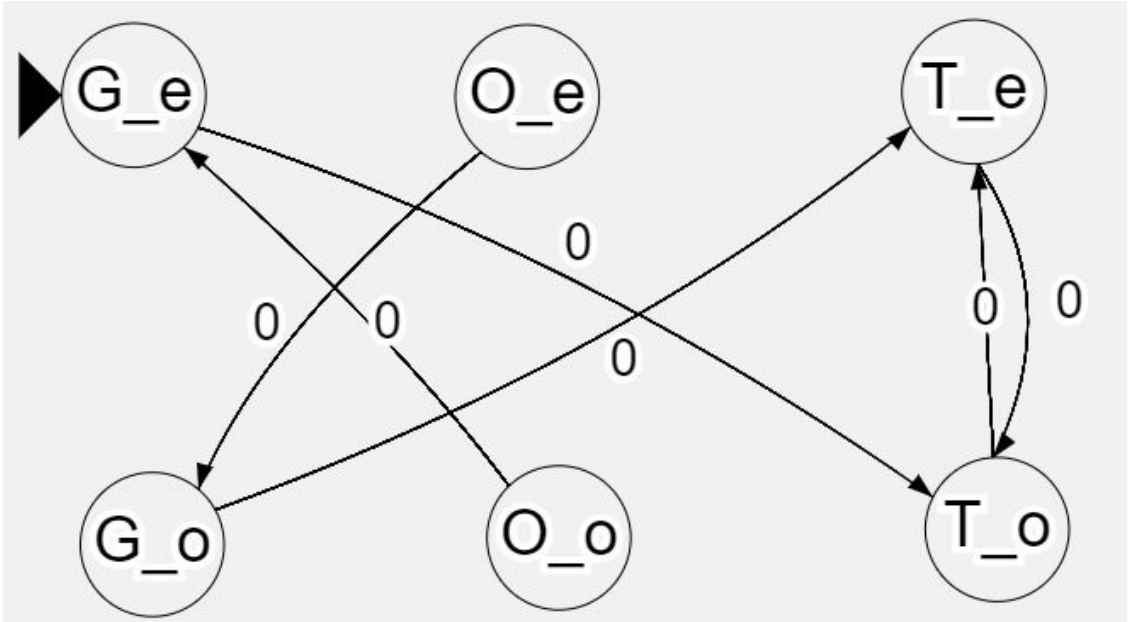
3: Identify δ



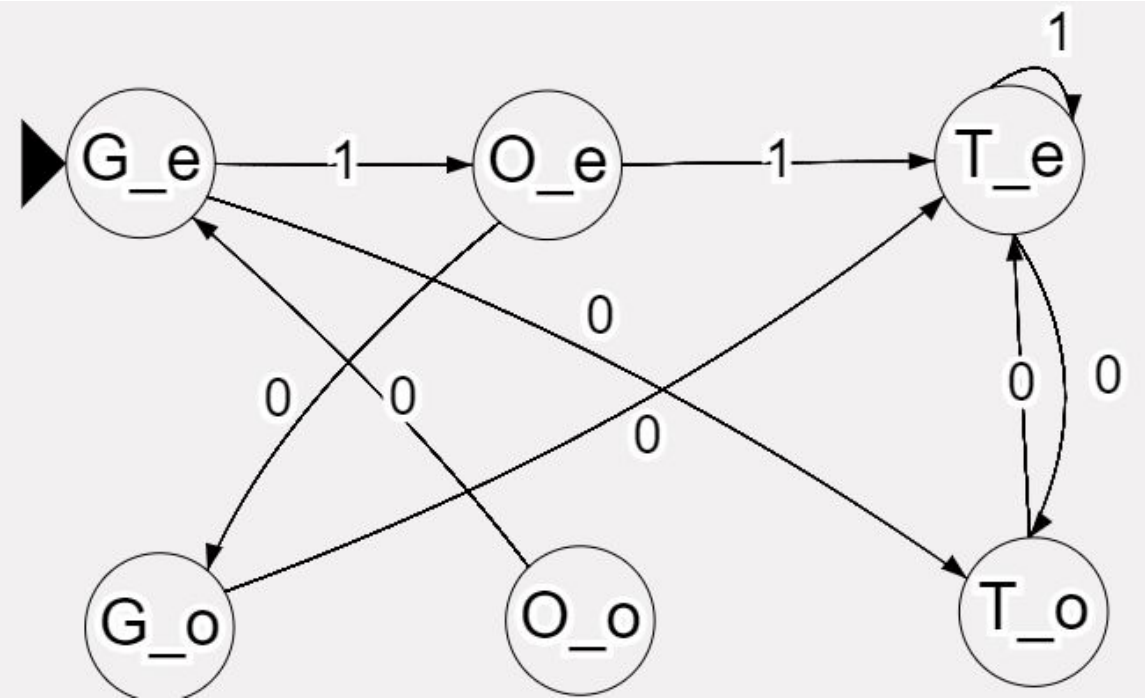
A_1

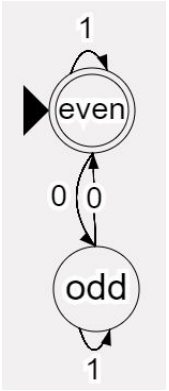


A_2

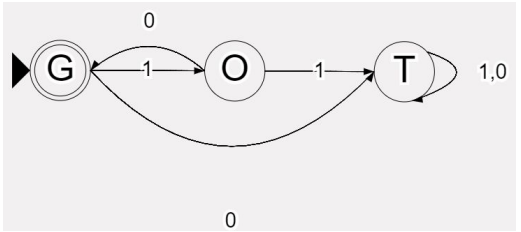


3: Identify δ



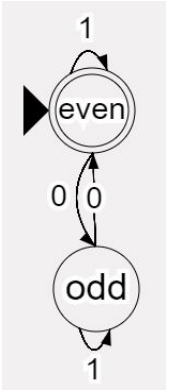


A_1

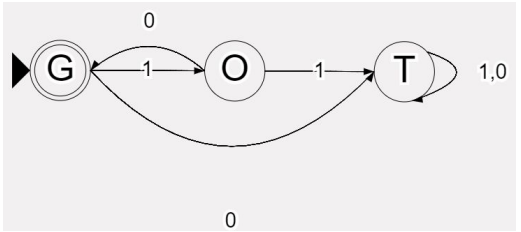


A_2

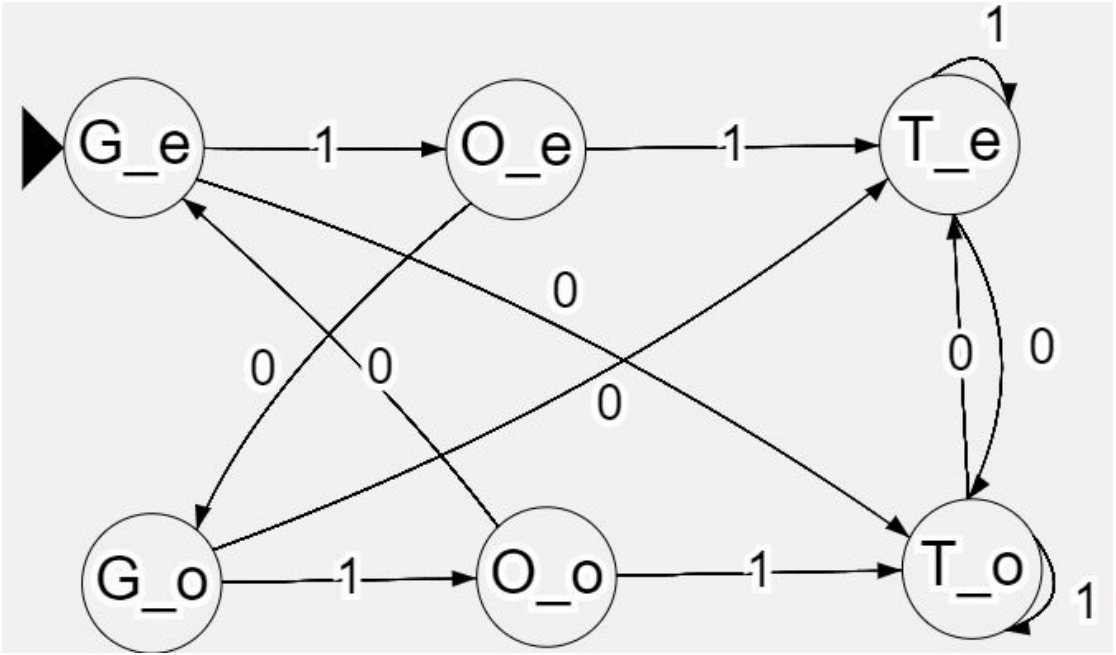
3: Identify δ



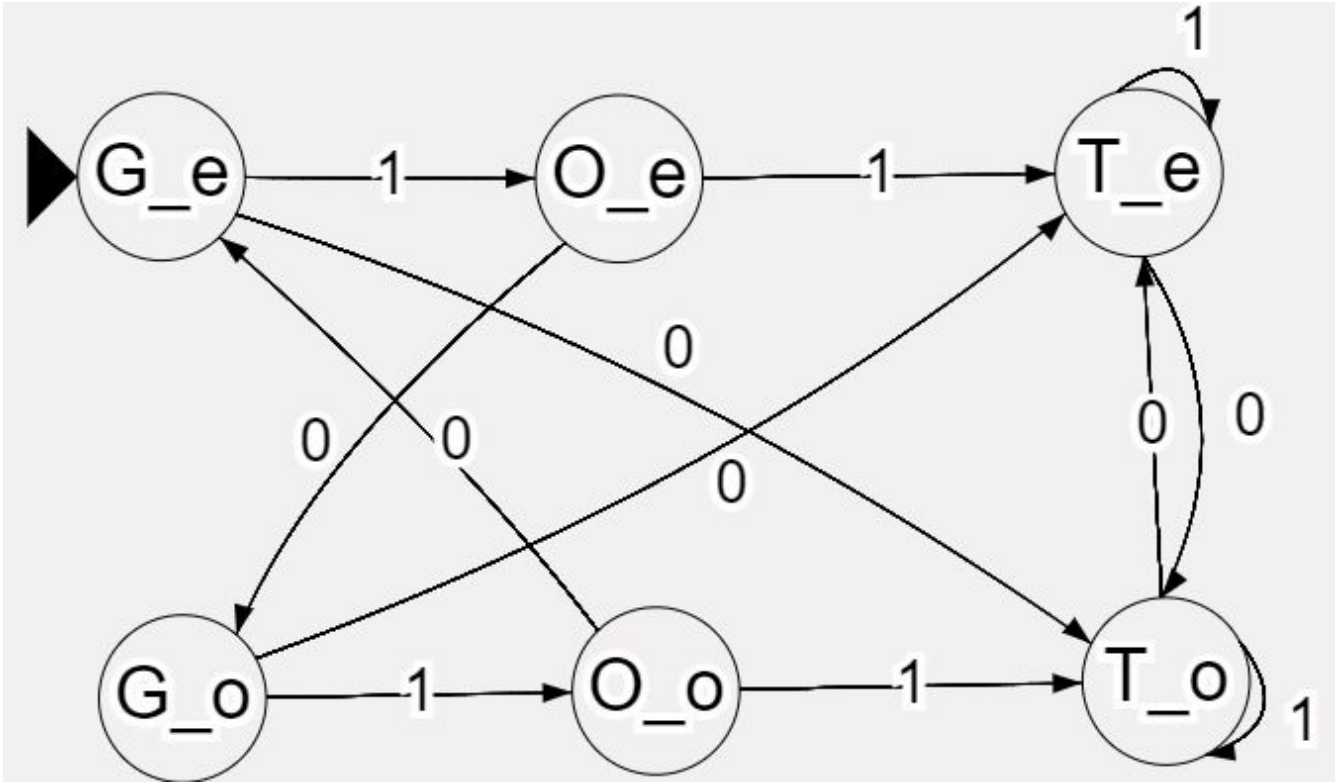
A_1



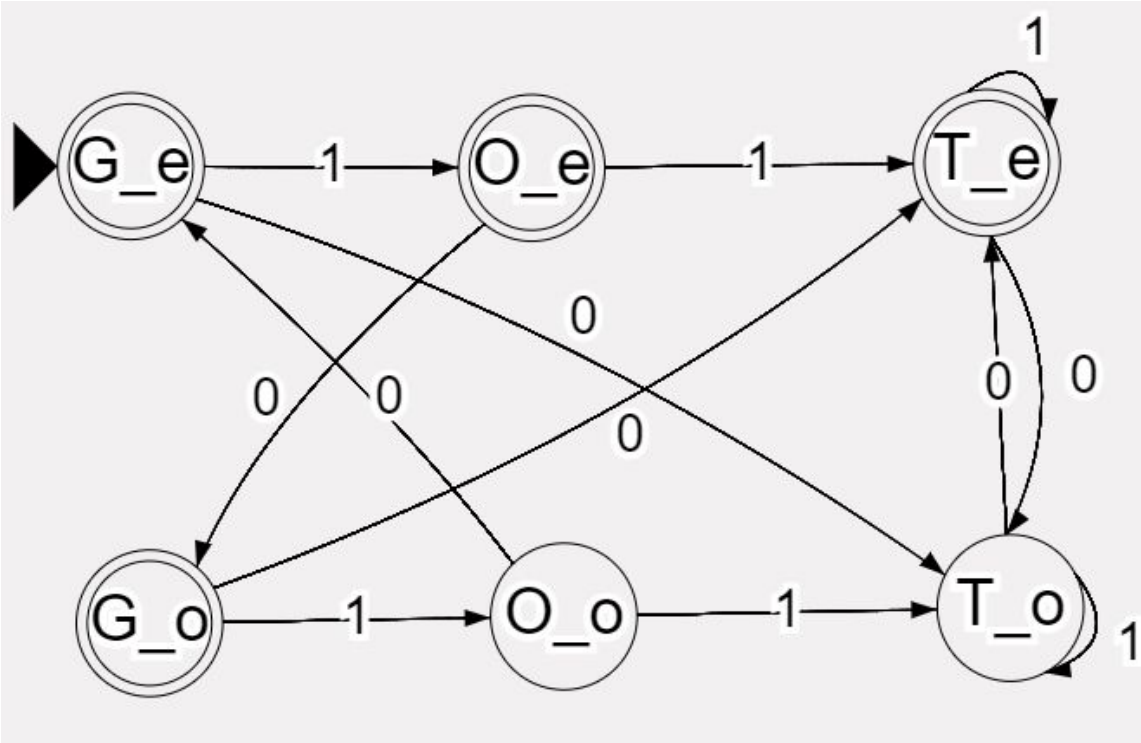
A_2



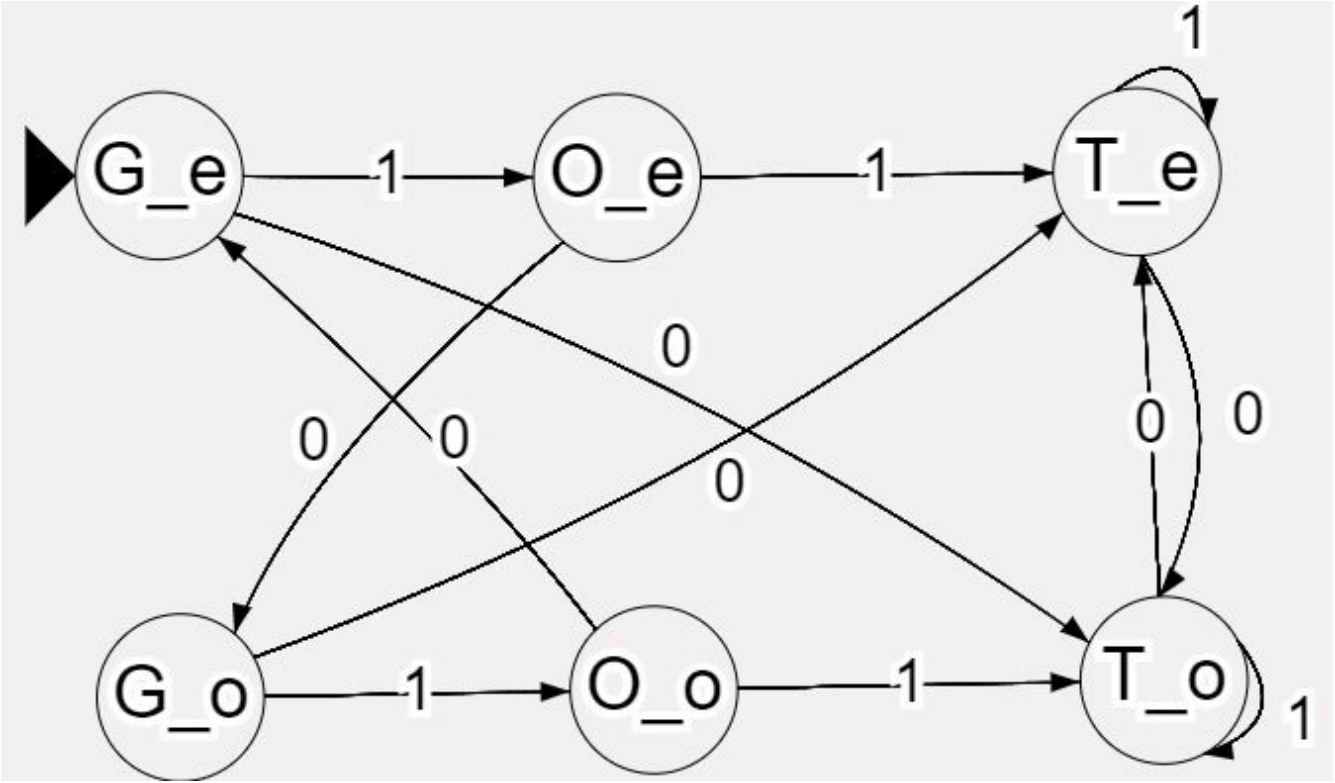
4: Identify F : A_1UA_2



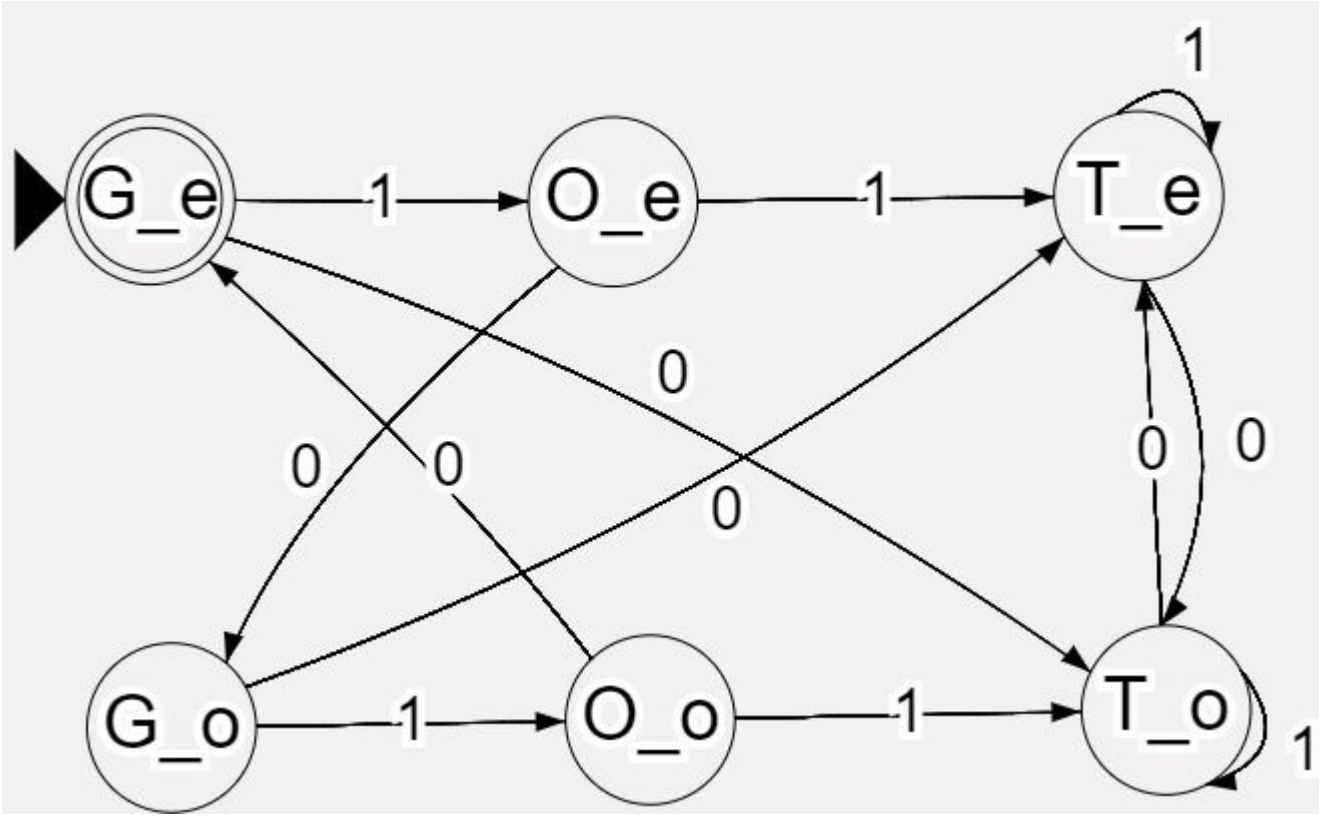
4: Identify F : A_1UA_2



4: Identify F : $A_1 \cap A_2$



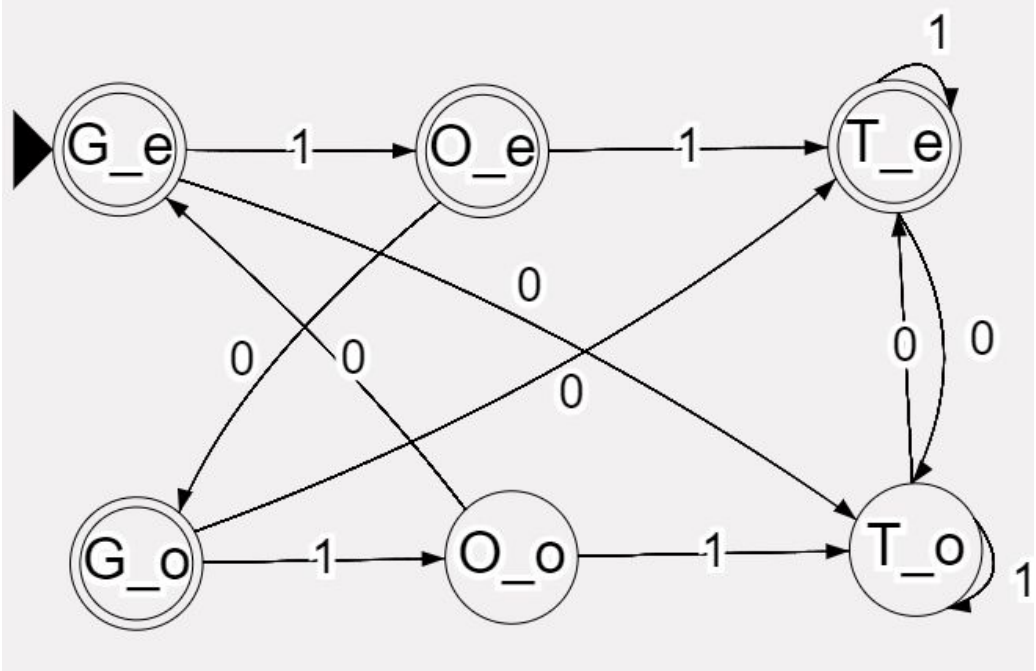
4: Identify F : $A_1 \cap A_2$



Reading strings over this automaton

Think and answer :

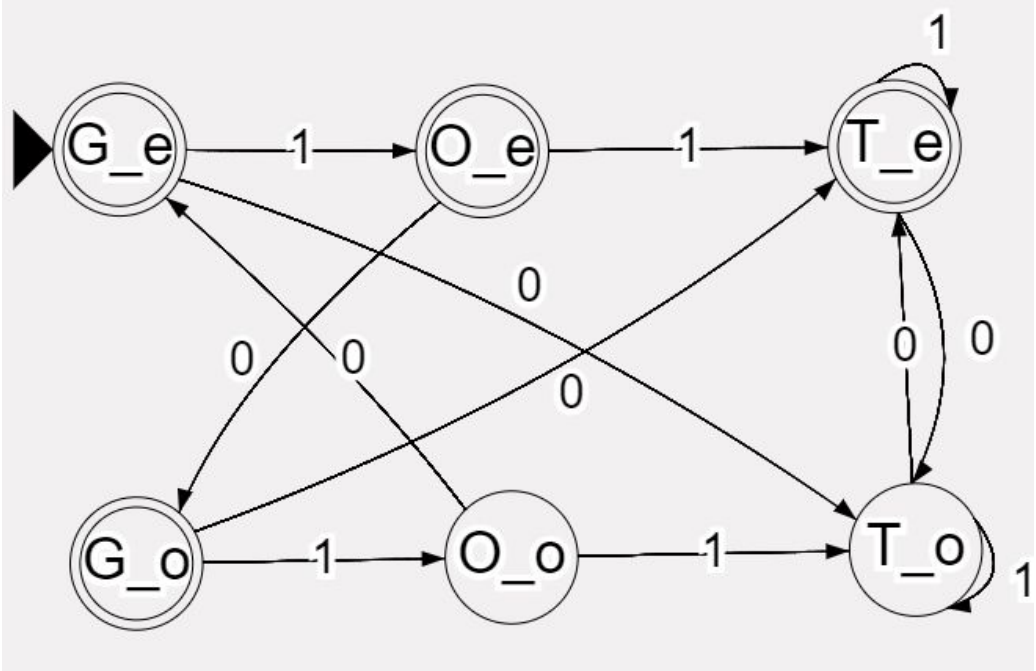
- What strings will end at state G_e
- What strings will end at state T_o ?
- What strings will end at state G_o?
- What strings will end at state O_o?
- What strings will end at state O_e ?



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 :

Set operations over L(NFAs)

Languages accepted by NFAs are closed under concatenation

Strategy :

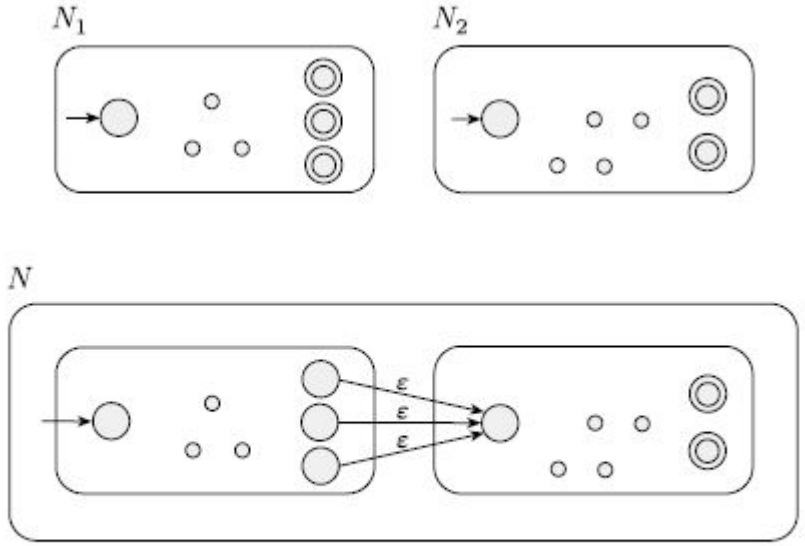


FIGURE 1.48
Construction of N to recognize $A_1 \circ A_2$

Practice: write out the formal definition!

Set operations over L(NFAs)

Languages accepted by NFAs are closed under Kleene *

Strategy :

Set operations over L(NFAs)

Languages accepted by NFAs are closed under Kleene *

Strategy :

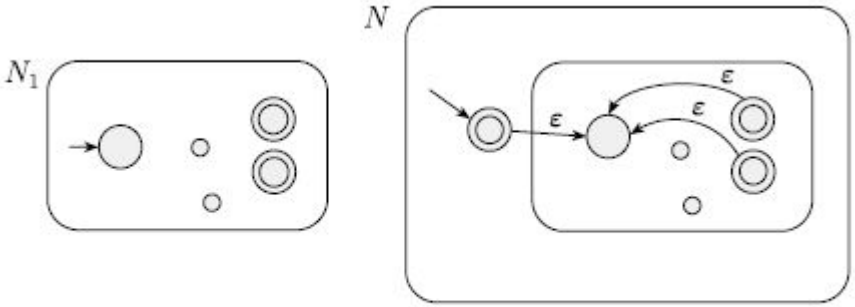


FIGURE 1.50
Construction of N to recognize A^*

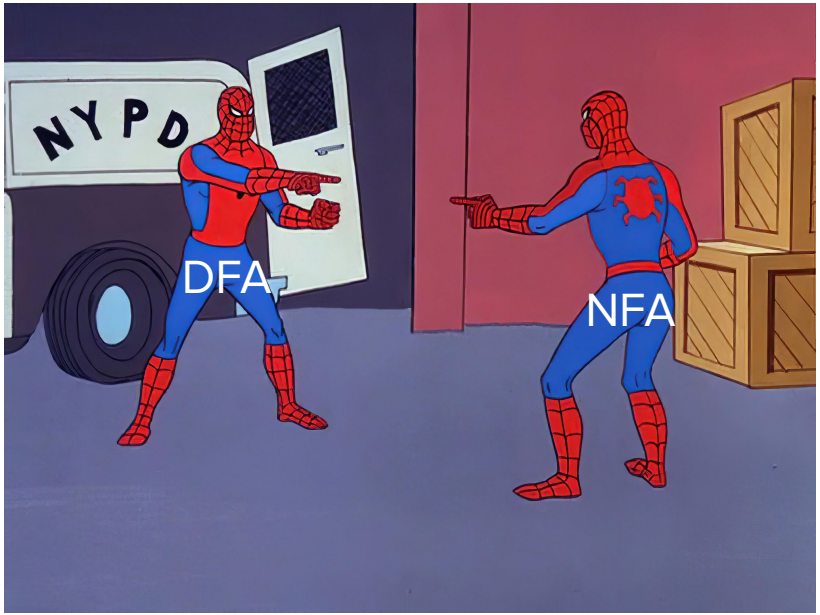
Practice: write out the formal definition!

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

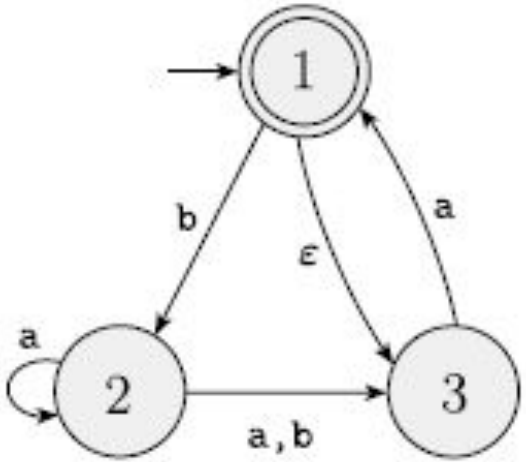
Bob : “To find a DFA which is equivalent to a NFA is slightly harder. I should have paid attention during lecture today and I possibly need to revise the material from Sipser pg 54-58”

True or False ?

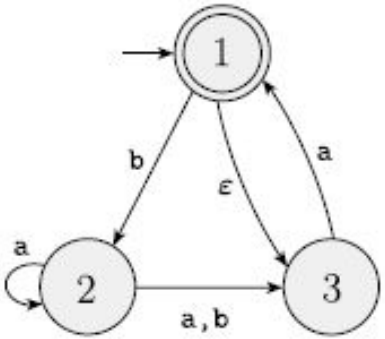
Let us start with DFA and NFA equivalence

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

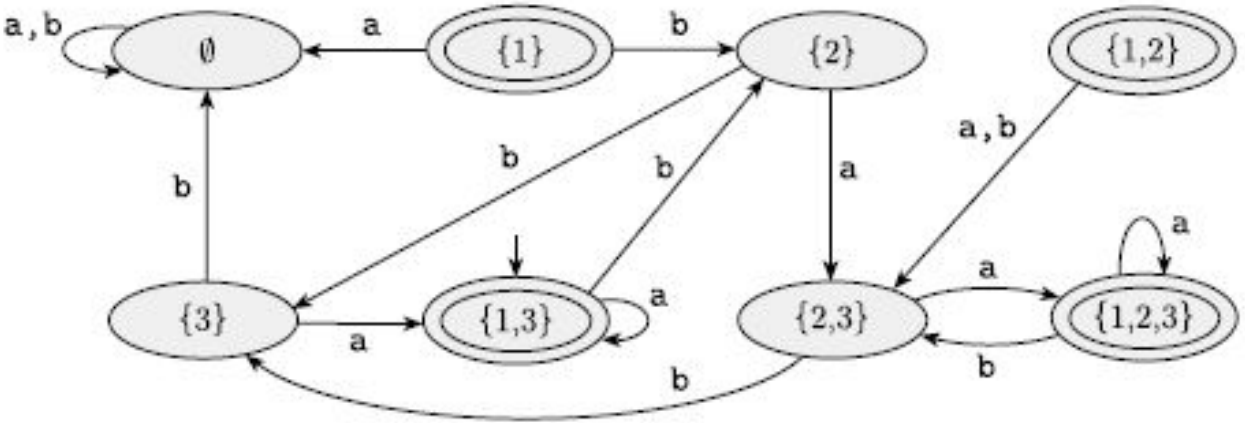
Sipser pg 57



Sipser pg 57



NFA N



DFA D recognizing L(N)

Now, NFA and RegEx equivalence

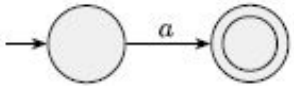
Regex to NFA

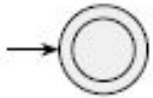
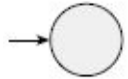
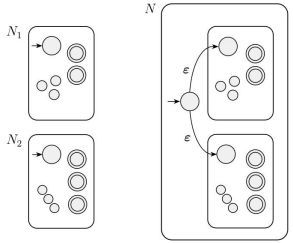
Recall :

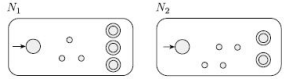
1. $R = a$, where $a \in \Sigma$
2. $R = \varepsilon$
3. $R = \emptyset$
4. $R = (R_1 \cup R_2)$, where R_1, R_2 are themselves regular expressions
5. $R = (R_1 \circ R_2)$, where R_1, R_2 are themselves regular expressions
6. (R_1^*), where R_1 is a regular expression.

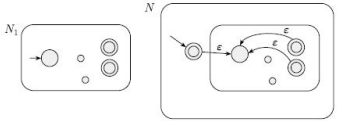
Regex to NFA

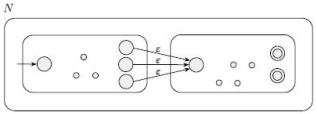
Recall :

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 

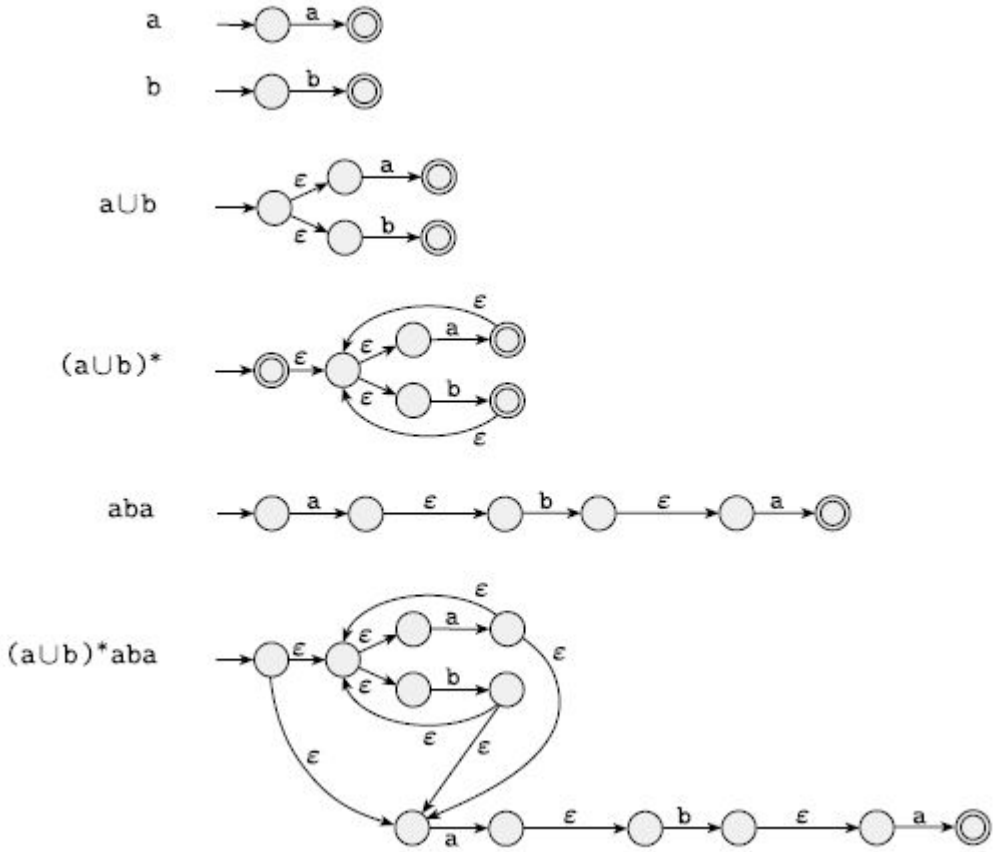
6. (R_1^*), where R_1 is a regular expression. 



Practice : (aUb)*aba

Example 1.58 Sipser, pg 69

Practice : (aUb)*aba

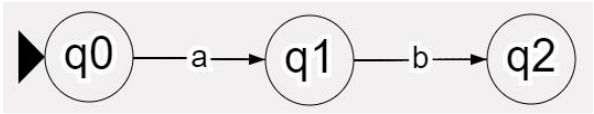


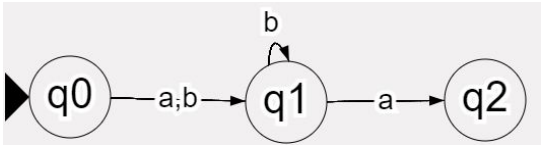
Example 1.58 Sipser, pg 69

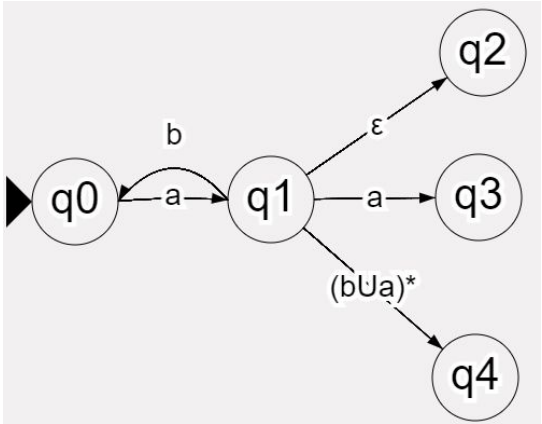
NFA/DFA to Regex

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)