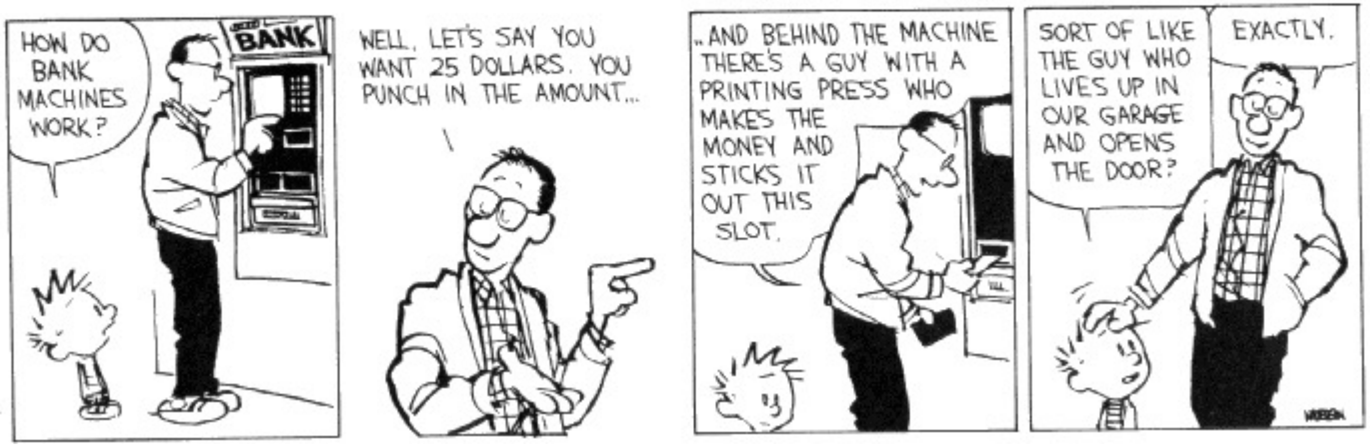
CSE 311: Foundations of Computing

Lecture 24: Directed Graphs and NFAs

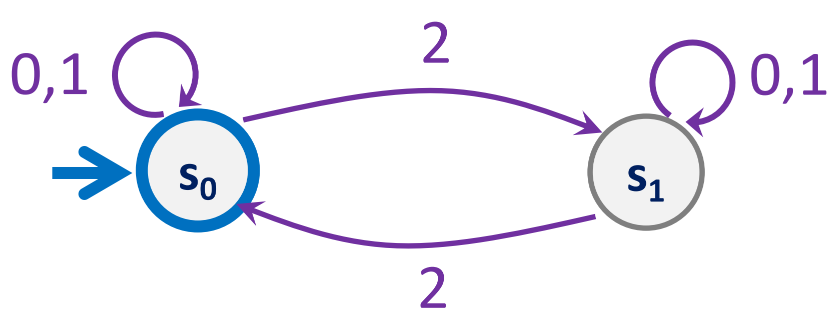


Recap: Finite State Machines (DFAs)

A DFA consists of:

- States
- Transitions on input symbols
- Start state and final states
- The "language recognized" by the machine is the set of strings that reach a final state from the start

Example: Strings with an even number of 2's

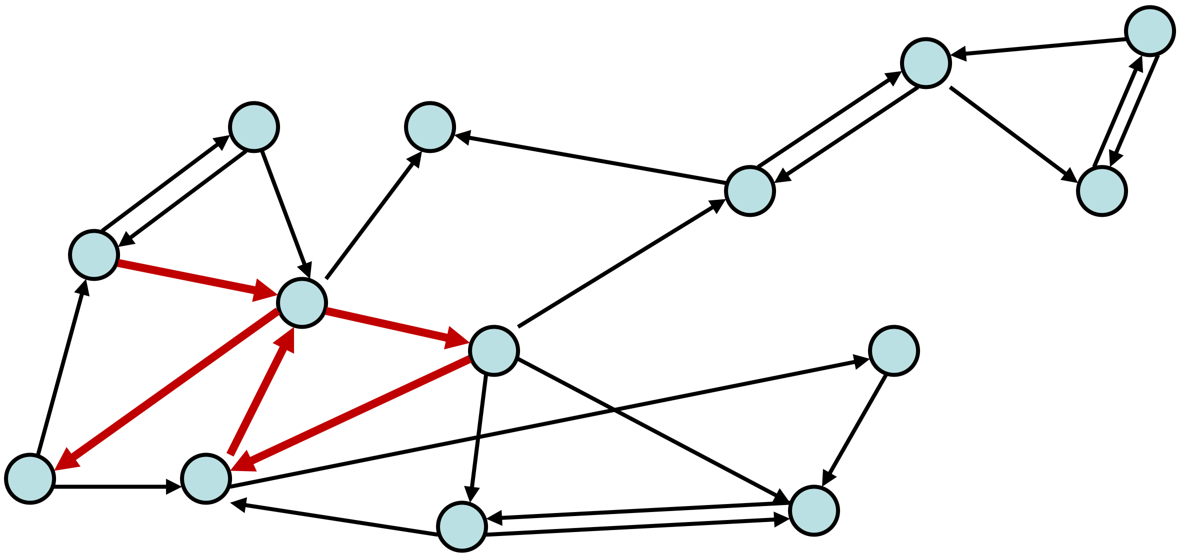


Recap: Directed Graphs

G = (V, E)   V – vertices
E – edges, ordered pairs of vertices

Path: $v_0, v_1, \ldots, v_k$ with each (v_i, v_{i+1}) in E

Simple Path: none of $\mathbf{v_0}, \ldots, \mathbf{v_k}$ repeated
Cycle: $\mathbf{v_0} = \mathbf{v_k}$
Simple Cycle: $\mathbf{v_0} = \mathbf{v_k}$, none of $\mathbf{v_1}, \ldots, \mathbf{v_k}$ repeated



Connectivity In Graphs

Defn: Two vertices in a graph are **connected** iff there is a path between them.

Let $\boldsymbol{R}$ be a relation on a set $\boldsymbol{A}$. The **connectivity** relation $\boldsymbol{R}^*$ consists of the pairs (a,b) such that there is a path from a to b in $\boldsymbol{R}$.

$$R^* = \bigcup_{k=0}^{\infty} R^k$$

Note: The text uses the wrong definition of this quantity. What the text defines (ignoring k=0) is usually called R^+

How Properties of Relations show up in Graphs

Let R be a relation on A.

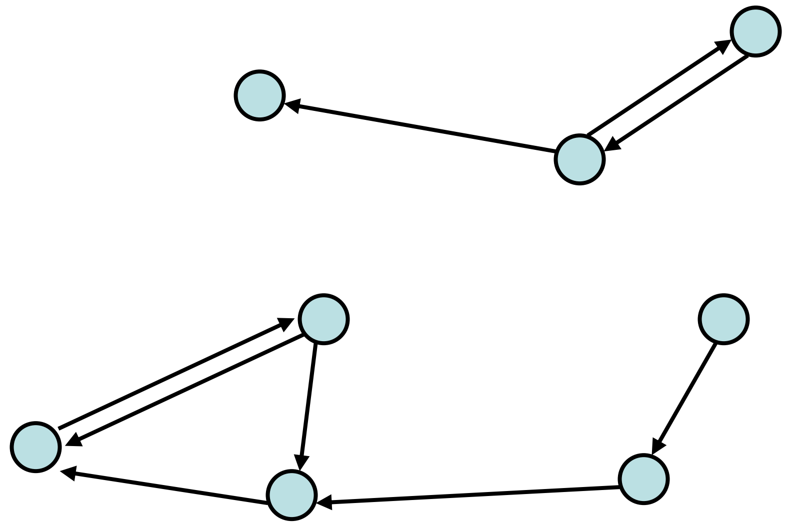
R is **reflexive** iff $(a,a) \in R$ for every $a \in A$

R is **symmetric** iff $(a,b) \in R$ implies $(b, a) \in R$

R is **antisymmetric** iff $(a,b) \in R$ and $a \neq b$ implies $(b,a) \notin R$

R is **transitive** iff $(a,b) \in R$ and $(b, c) \in R$ implies $(a, c) \in R$

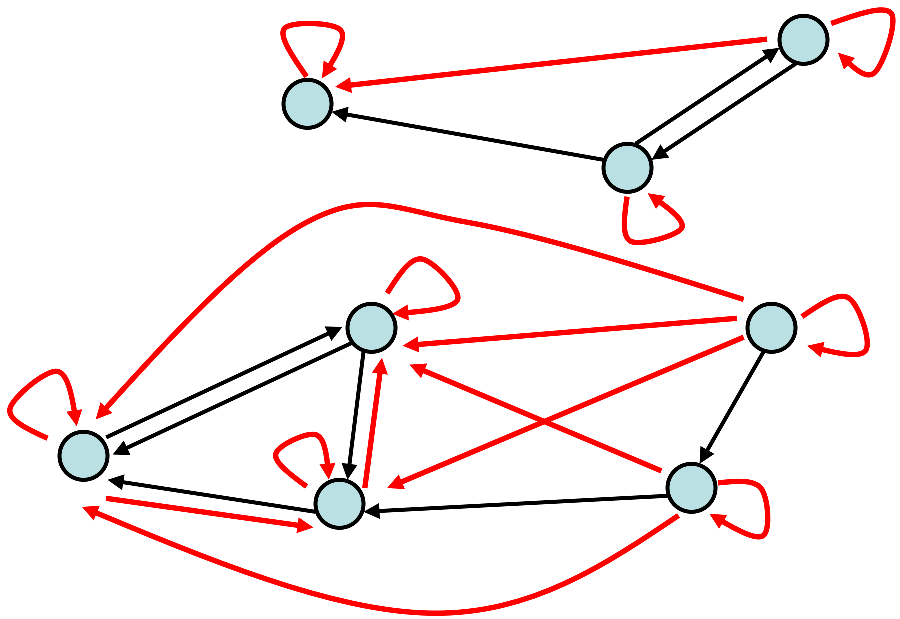
Transitive-Reflexive Closure



Add the **minimum possible** number of edges to make the relation transitive and reflexive.

The **transitive-reflexive closure** of a relation $\boldsymbol{R}$ is the connectivity relation $\boldsymbol{R^*}$

Transitive-Reflexive Closure



Relation with the **minimum possible** number of **extra edges** to make the relation both transitive and reflexive.

The **transitive-reflexive closure** of a relation $\boldsymbol{R}$ is the connectivity relation $\boldsymbol{R}^*$

n-ary Relations

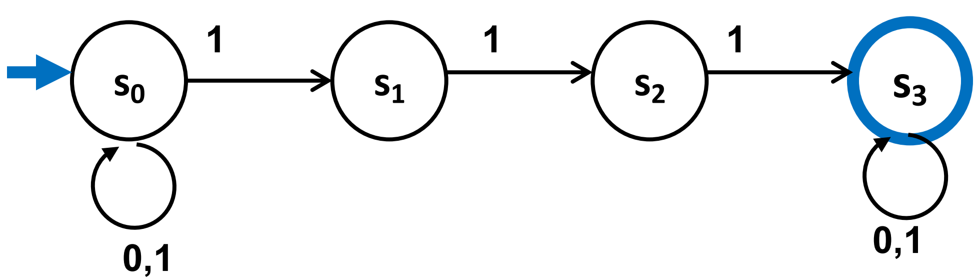
Let $A_1, A_2, \dots, A_n$ be sets. An **n-ary** relation on these sets is a subset of $A_1 \times A_2 \times \cdots \times A_n$.

Example application: Database theory

Student_Name	ID_Number	Office	GPA
Knuth	328012098	022	4.00
Von Neuman	481080220	555	3.78
Russell	238082388	022	3.85
Einstein	238001920	022	2.11
Newton	1727017	333	3.61
Karp	348882811	022	3.98
Bernoulli	2921938	022	3.21

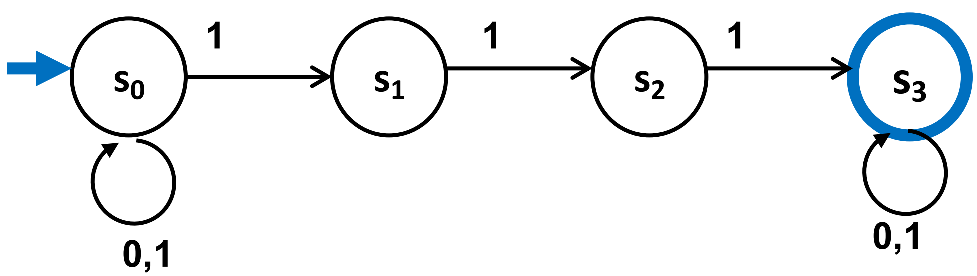
Nondeterministic Finite Automata (NFA)

- Graph with start state, final states, edges labeled by symbols (like DFA) but
  - Not required to have exactly 1 edge out of each state labeled by each symbol— can have 0 or >1
  - Also can have edges labeled by empty string ε
- **Definition:** x is in the language recognized by an NFA if and only if <u>some</u> valid execution of the machine gets to an accept state



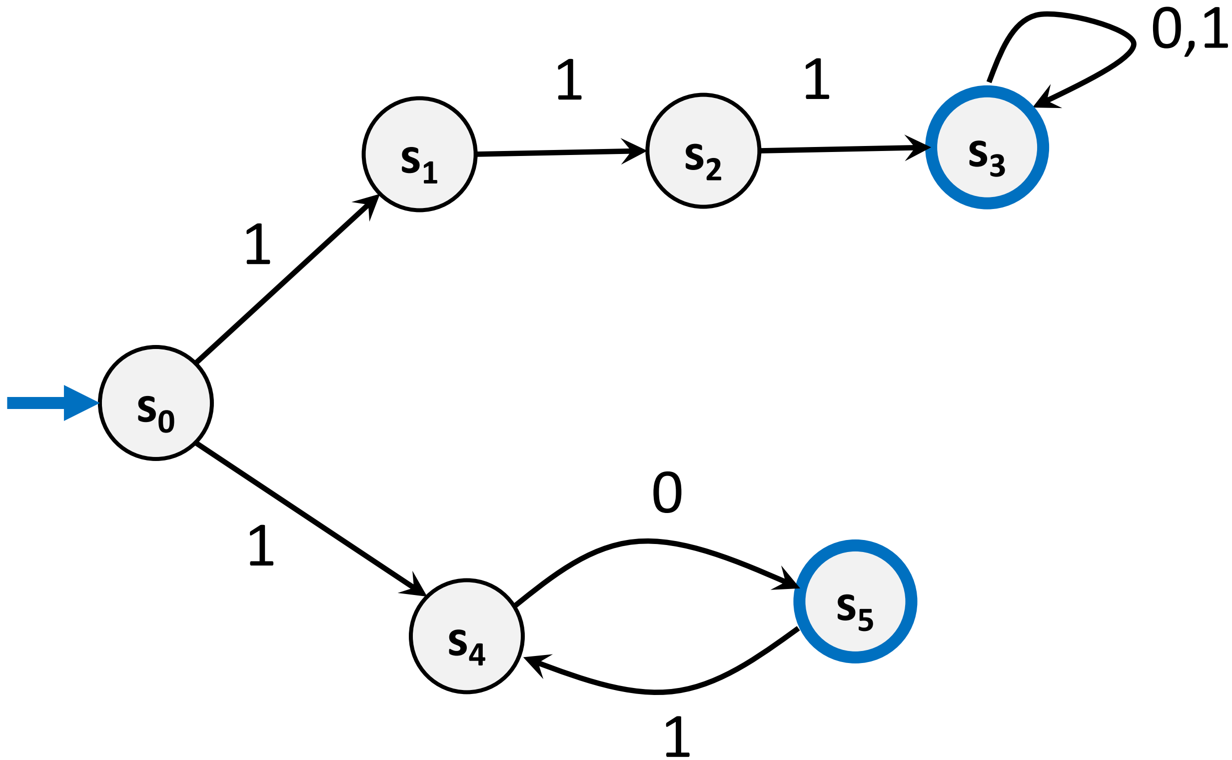
Nondeterministic Finite Automata (NFA)

- Graph with start state, final states, edges labeled by symbols (like DFA) but
  - Not required to have exactly 1 edge out of each state labeled by each symbol— can have 0 or >1
  - Also can have edges labeled by empty string ε
- **Definition:** x is in the language recognized by an NFA if and only if <u>some</u> valid execution of the machine gets to an accept state



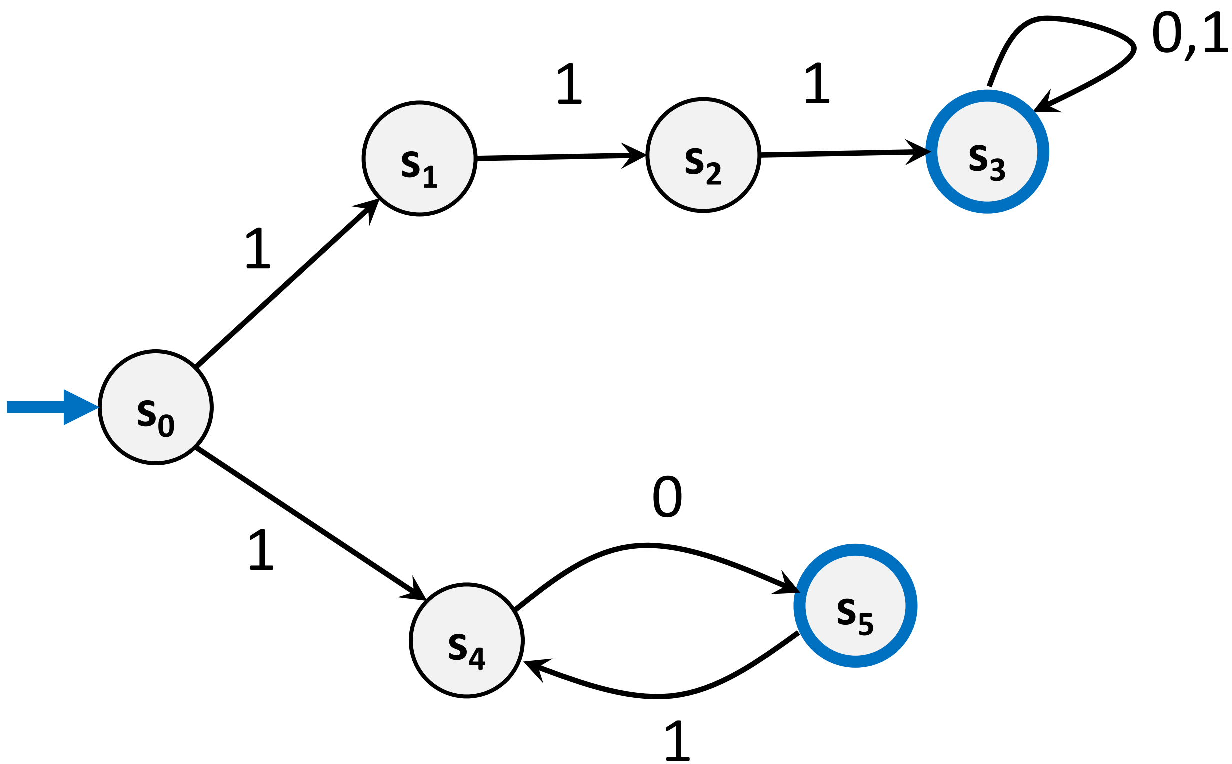
Recognized language: $(0 \cup 1)^*111(0 \cup 1)^*$ as RE

Consider This NFA



What language does this NFA accept?

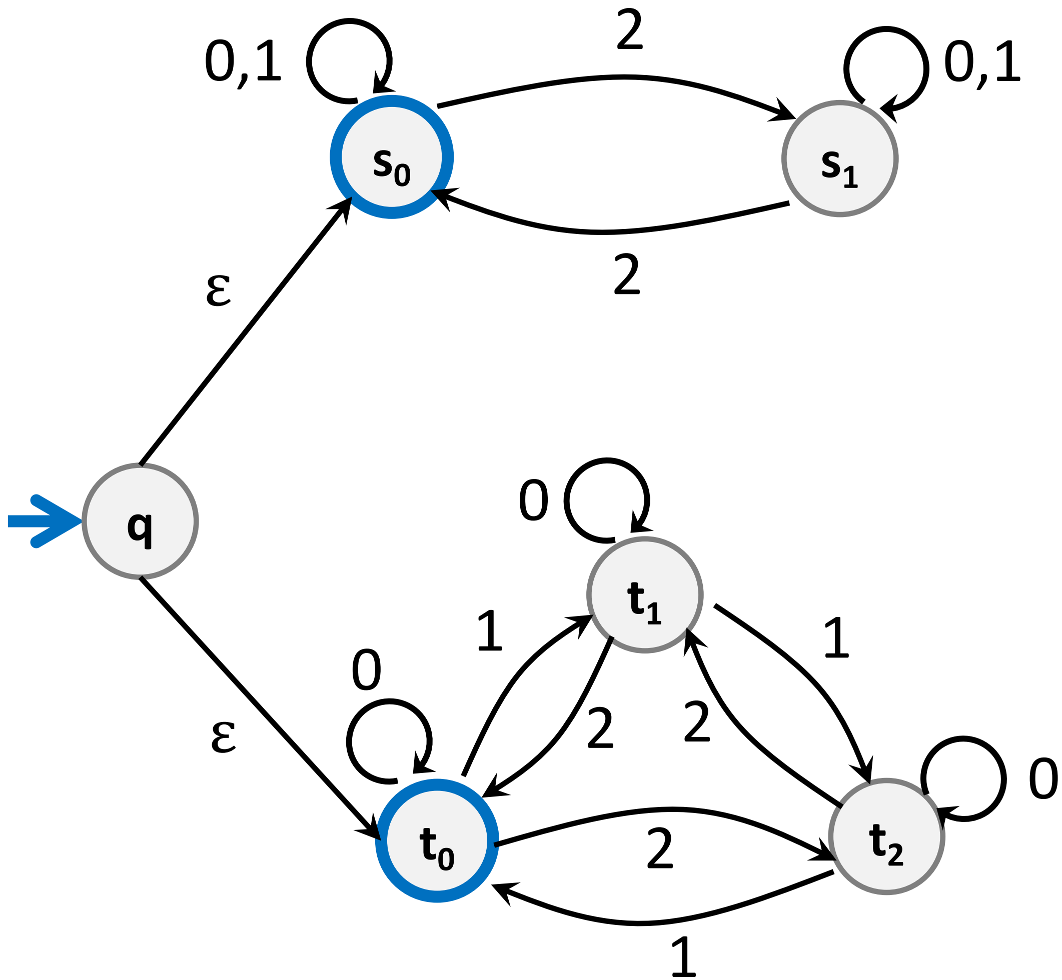
Consider This NFA



What language does this NFA accept?

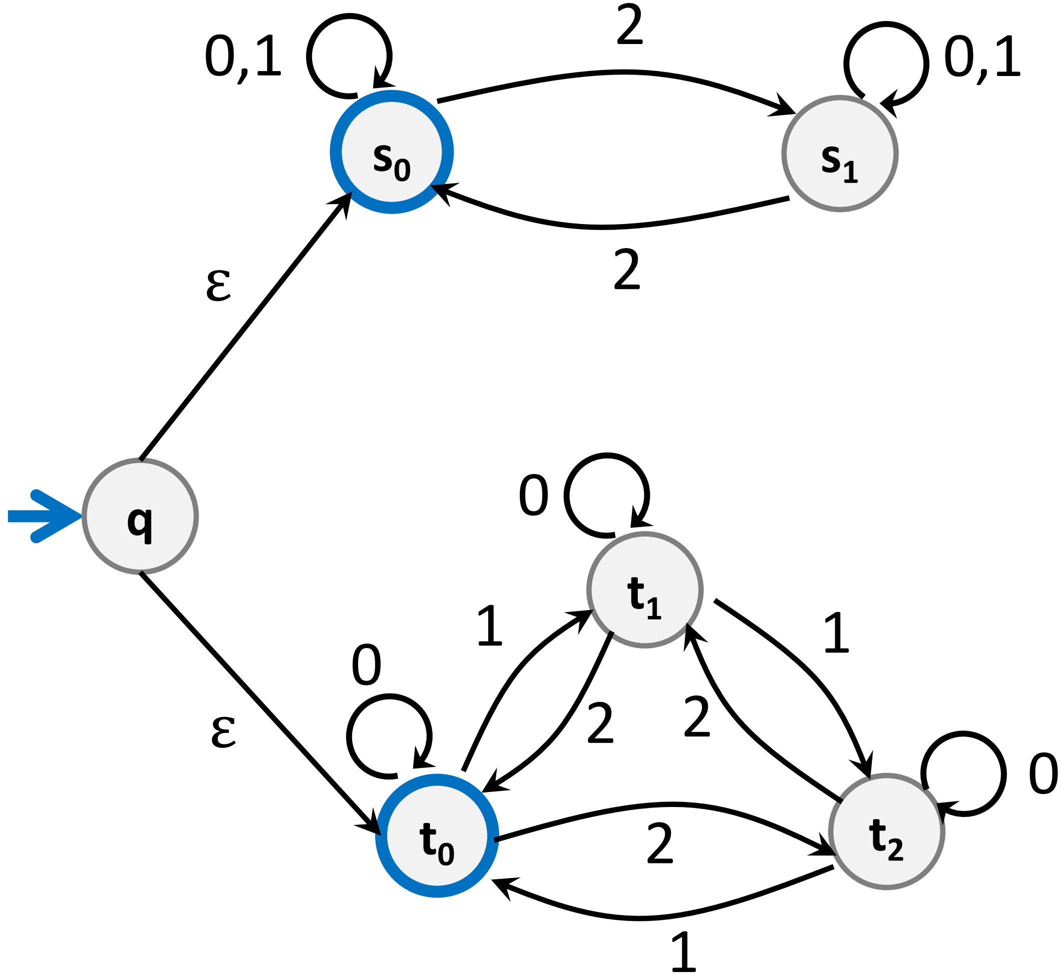
10(10)* ∪ 111 (0 ∪ 1)*

NFA ε-moves



NFA ε-moves

Strings over {0,1,2} w/even # of 2's OR sum to 0 mod 3



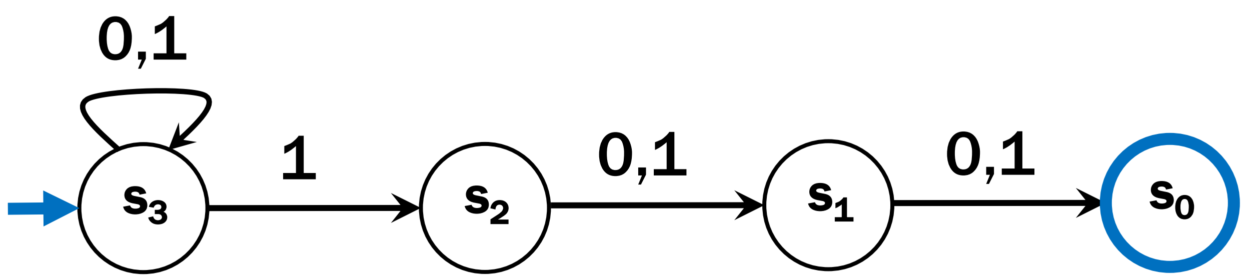
Lecture 24 Activity

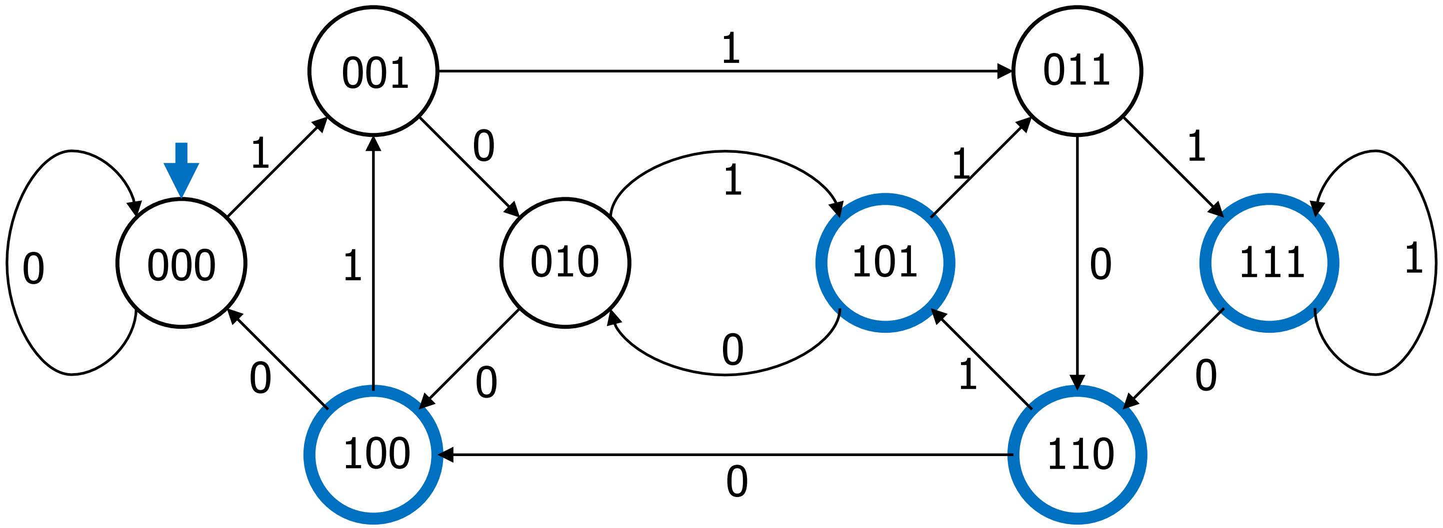
- You will be assigned to **breakout rooms**. Please:
- Introduce yourself
- Choose someone to share screen, showing this PDF

Construct an NFA for the set of binary strings with a 1 in the 3rd position from the end

Fill out a poll everywhere for Activity Credit!
Go to pollev.com/thomas311 and login
with your UW identity

Compare with the smallest DFA



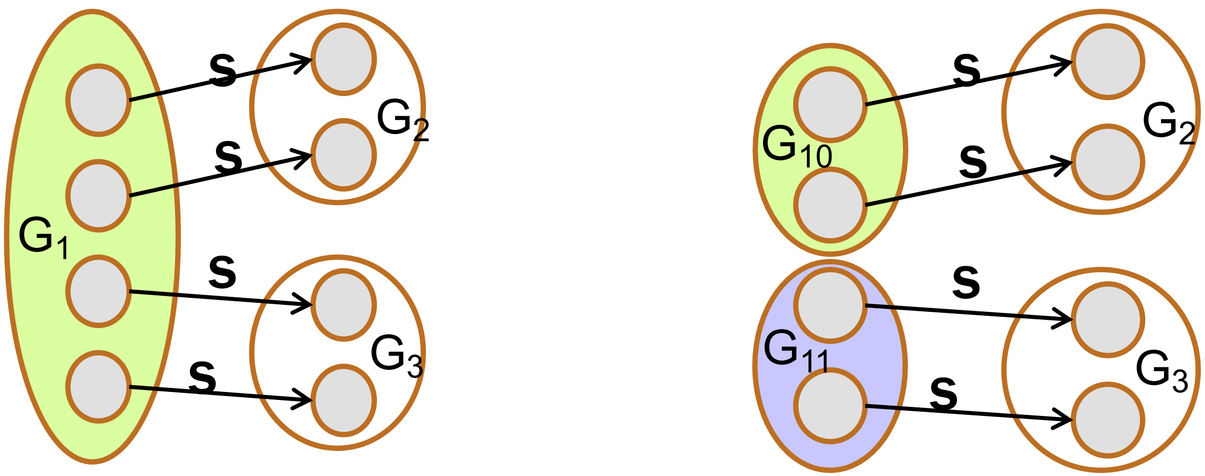


State Minimization

- Many different FSMs (DFAs) for the same problem
- Take a given FSM and try to reduce its state set by combining states
  - Algorithm will always produce the unique minimal equivalent machine (up to renaming of states) but we won't prove this

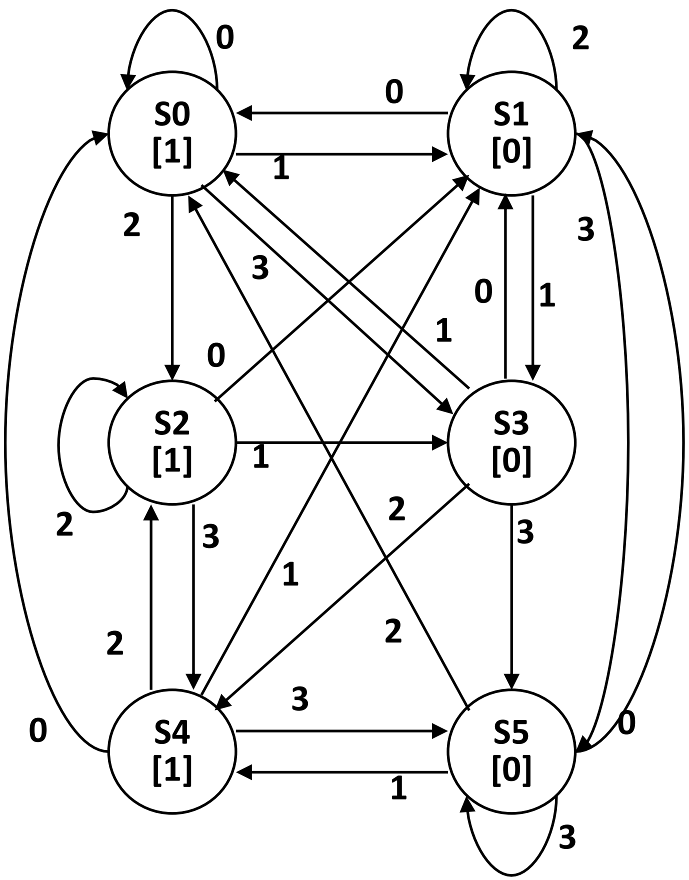
State Minimization Algorithm

1. Put states into groups based on their outputs (or whether they are final states or not)
2. Repeat the following until no change happens
   a. If there is a symbol s so that not all states in a group G agree on which group s leads to, split G into smaller groups based on which group the states go to on s



3. Finally, convert groups to states

State Minimization Example

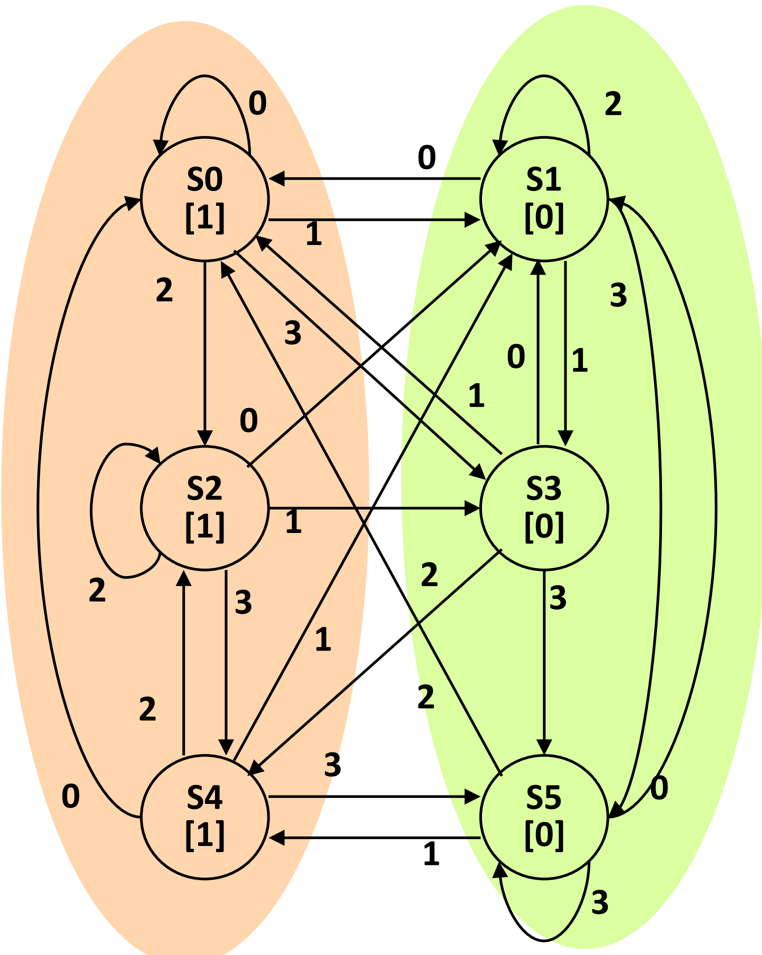


present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

state
transition table

Put states into groups based on their outputs (or whether they are final states or not)

State Minimization Example

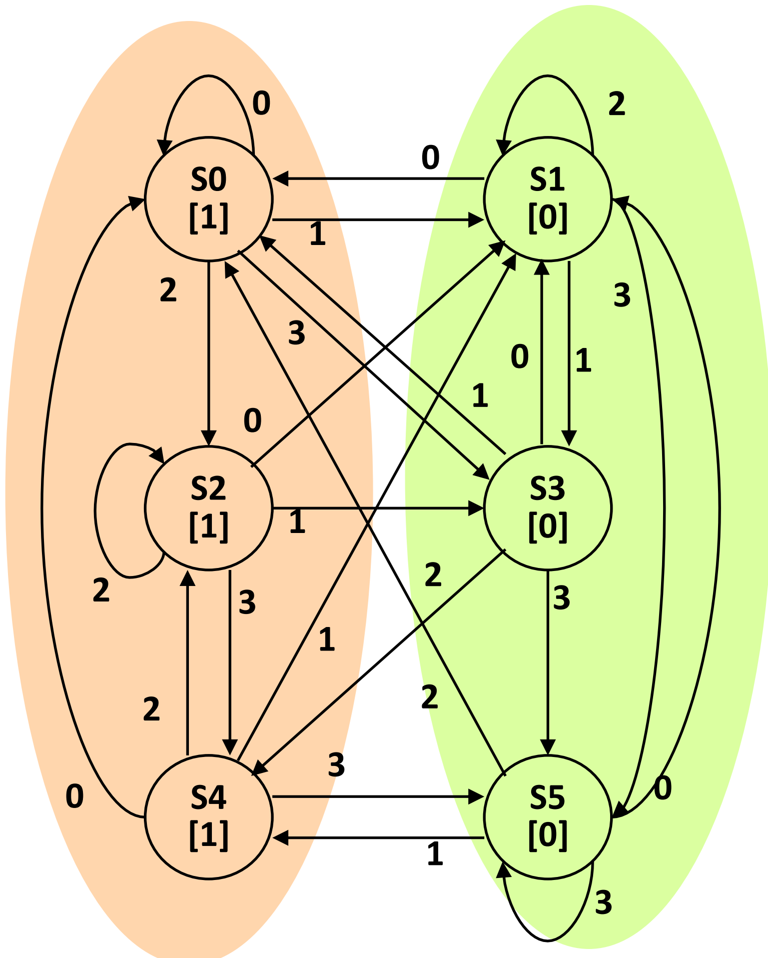


present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

state
transition table

Put states into groups based on their outputs (or whether they are final states or not)

State Minimization Example



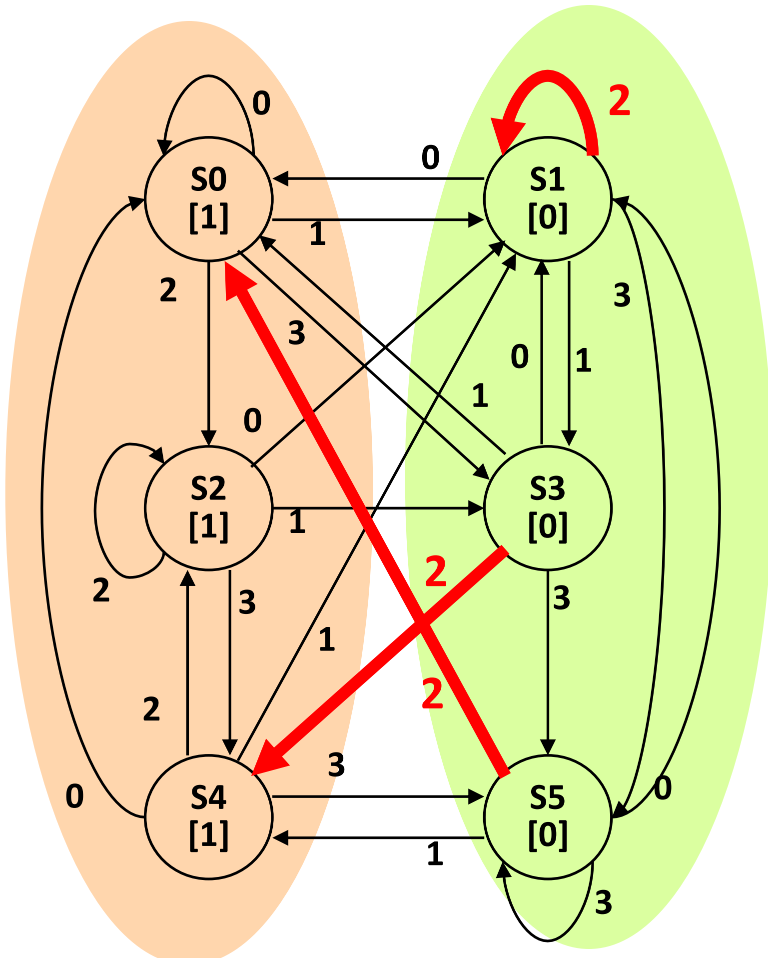
present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

state
transition table

Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol **s** so that not all states in a group **G** agree on which group **s** leads to, split **G** based on which group the states go to on **s**

State Minimization Example



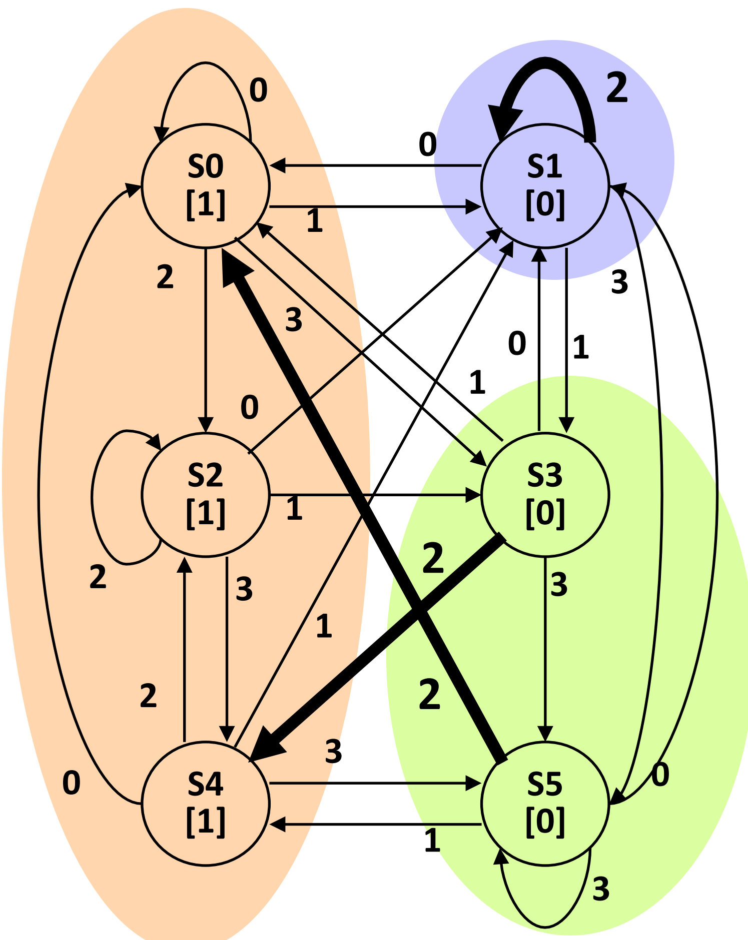
present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

state
transition table

Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol **s** so that not all states in a group **G** agree on which group **s** leads to, split **G** based on which group the states go to on **s**

State Minimization Example



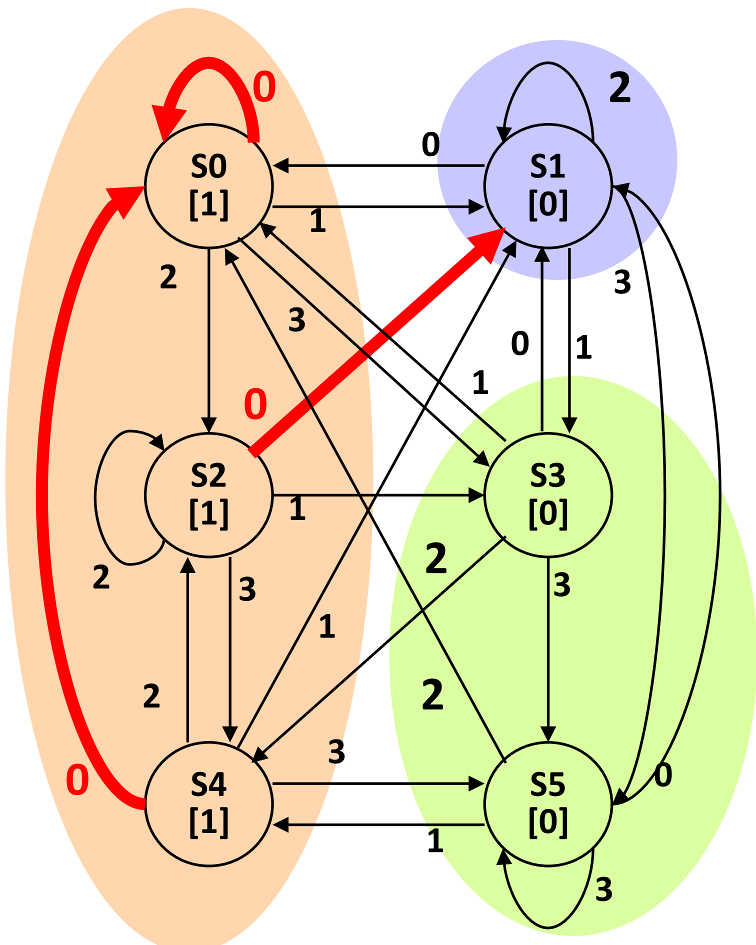
present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

state
transition table

Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol **s** so that not all states in a group **G** agree on which group **s** leads to, split **G** based on which group the states go to on **s**

State Minimization Example



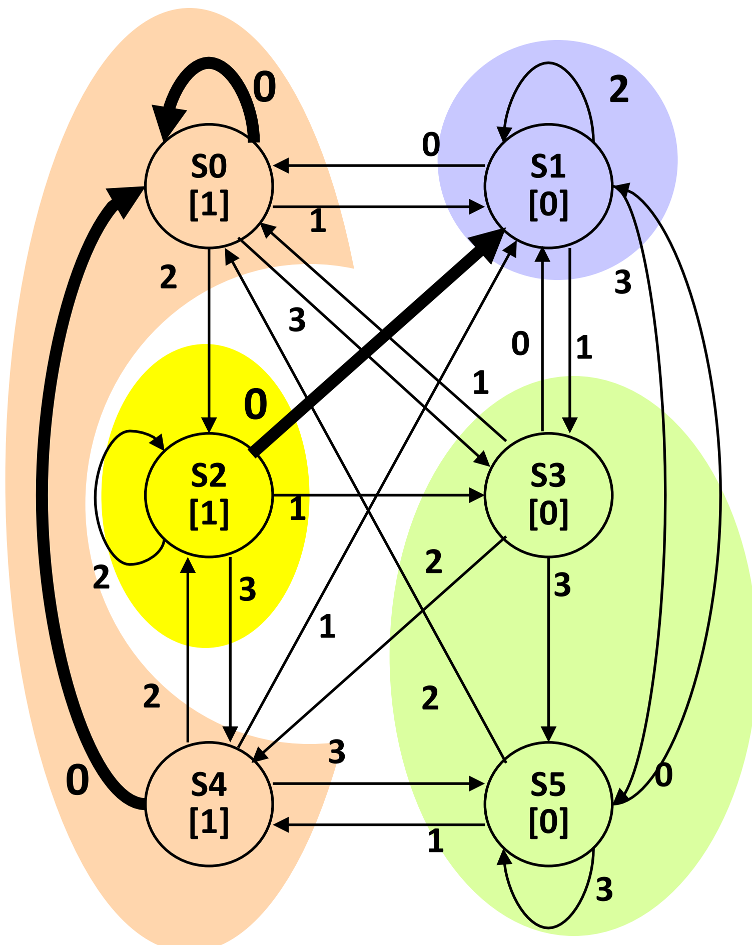
present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

state
transition table

Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol **s** so that not all states in a group **G** agree on which group **s** leads to, split **G** based on which group the states go to on **s**

State Minimization Example



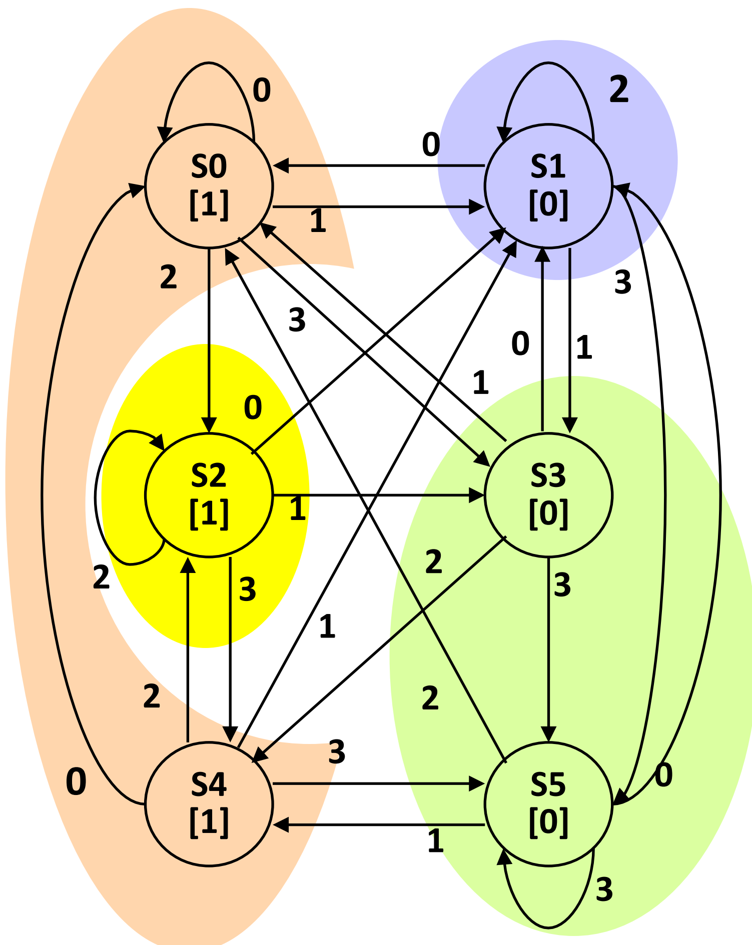
present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

state transition table

Put states into groups based on their outputs (or whether they are final states or not)

If there is a symbol **s** so that not all states in a group **G** agree on which group **s** leads to, split **G** based on which group the states go to on **s**

State Minimization Example



present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

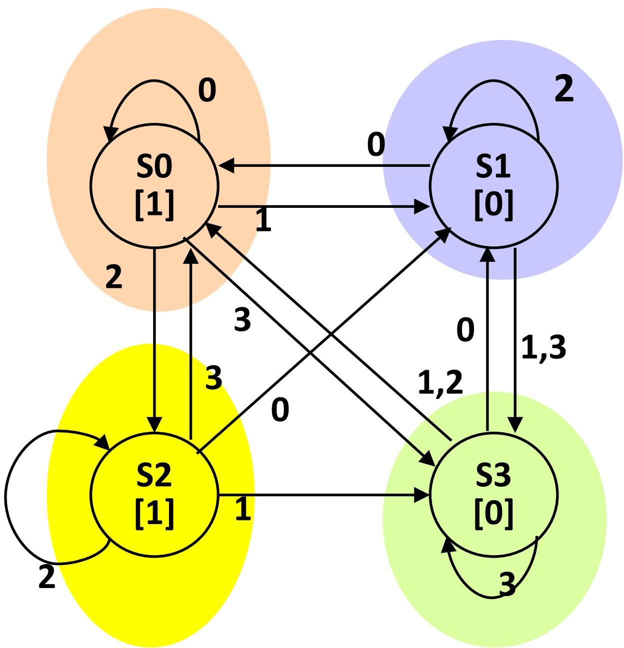
state transition table

Finally convert groups to states:

Can combine states S0-S4 and S3-S5.

In table replace all S4 with S0 and all S5 with S3

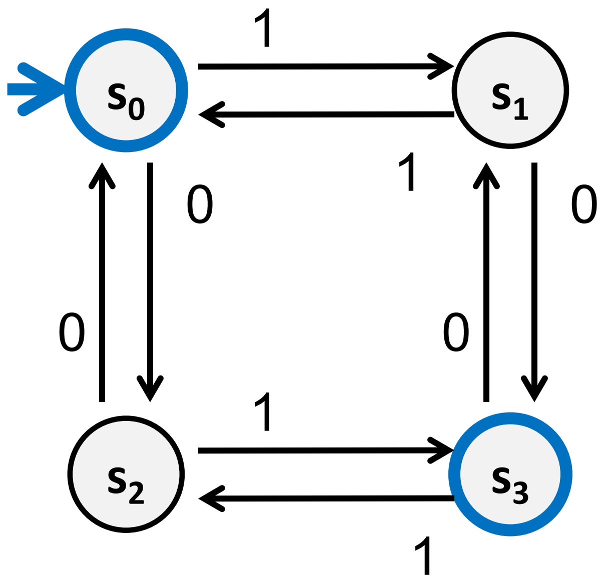
Minimized Machine



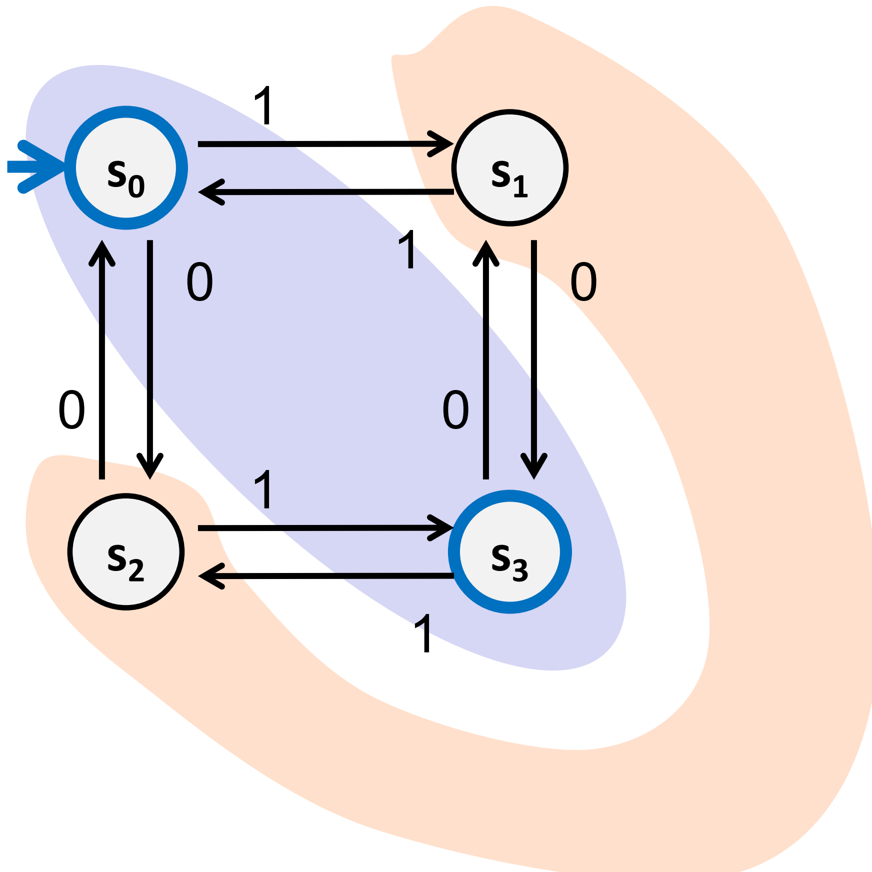
present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S3	0
S2	S1	S3	S2	S0	1
S3	S1	S0	S0	S3	0

state
transition table

A Simpler Minimization Example



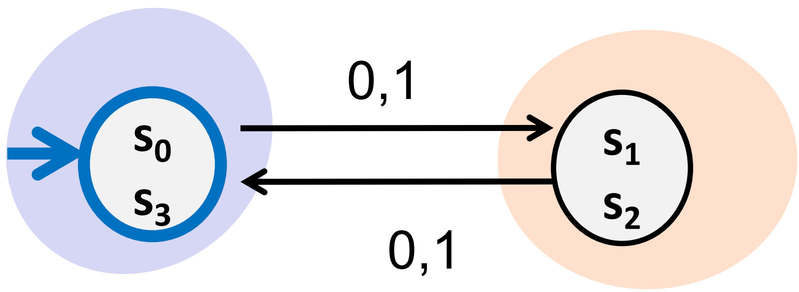
A Simpler Minimization Example



Split states into final/non-final groups

Every symbol causes the DFA to go from one group to the other so neither group needs to be split

Minimized DFA



Partial Correctness of Minimization Algorithm

- **Prove this claim: after processing input x, if the old machine was in state q, then the new machine is in the state S with $q \in S$**
  - **True after 0 characters processed**
  - **If true after k characters processed, then it's true after k+1 characters processed:**

    By inductive hypothesis, after k steps, old machine is in state q and new one in state S with $q \in S$

    By construction, every $r \in S$ is taken to the same state S' on input x_{k+1}, so q is taken to some $q' \in S'$.
- **At end, since every $r \in S$ is accepting or rejecting, new machine gives correct answer.**

Another way to look at DFAs

Definition: The label of a path in a DFA is the concatenation of all the labels on its edges in order

Lemma: x is in the language recognized by a DFA iff x labels a path from the start state to some final state