Adam Blank Spring 2017

CSE 311

Foundations of Computing I

* All slides are a combined effort between instructors of the course

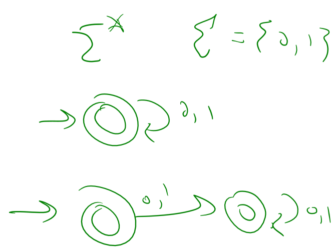
CSE 311: Foundations of Computing

Lecture 22: DFA Minimization!

Hi!

State Minimization

- Many different FSMs (DFAs) for the same problem

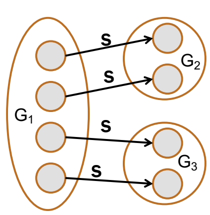


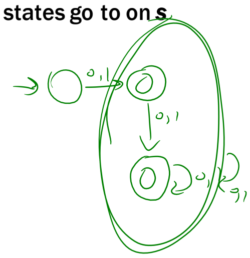
State Minimization

- 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

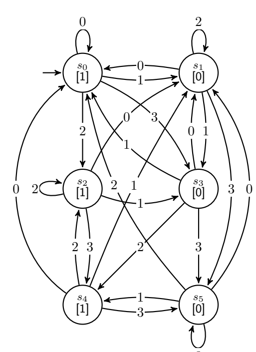
State Minimization Algorithm

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***.





State Minimization Example

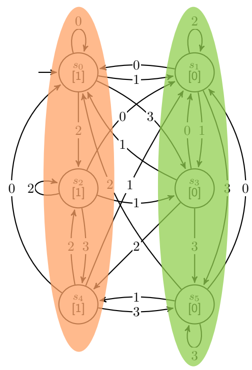


present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

state transition table

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

State Minimization Example

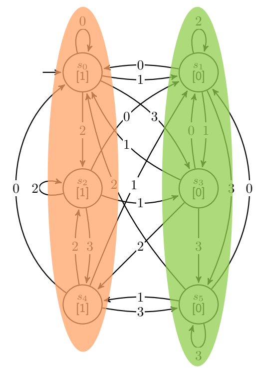


present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

state
transition table

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

State Minimization Example



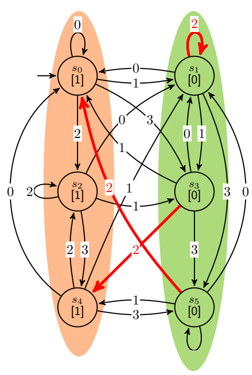
present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

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



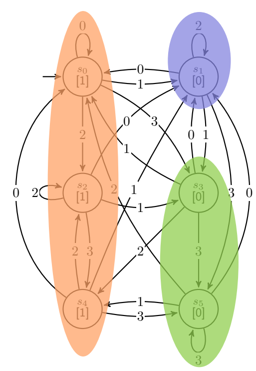
present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

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



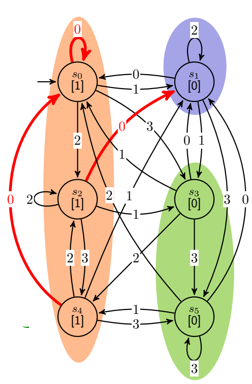
present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

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



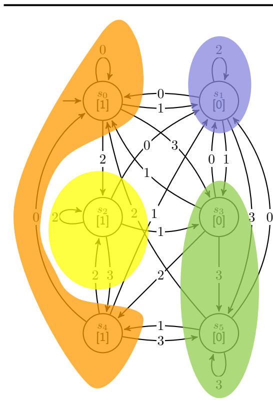
present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

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



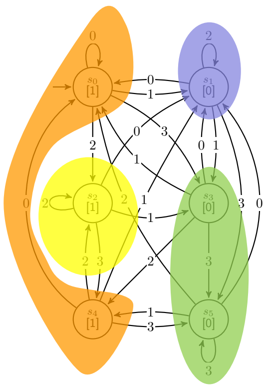
present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

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



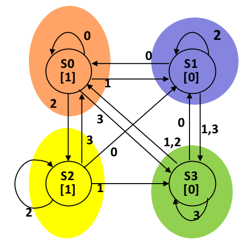
present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S5	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

state transition table

Can combine states S0-S4 and S3-S5.

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

Minimized Machine



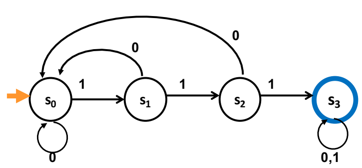
present state	next state 0	1	2	3	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S3	0
S2	S1	S3	S2	S0	1
S3	S1	S0	S0	S3	0

state transition table

Another way to look at DFAs

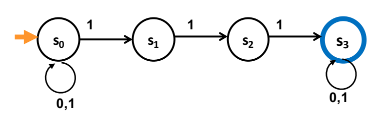
Definition: The label of a path in a DFA is the concatenation of all the labels on its edges in order

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

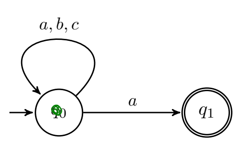


Nondeterministic Finite Automata (NFA)

- 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 x labels a path from the start state to some final state



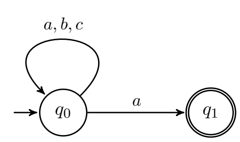
Consider This NFA



Is it a DFA?

What language does it accept?

Consider This NFA



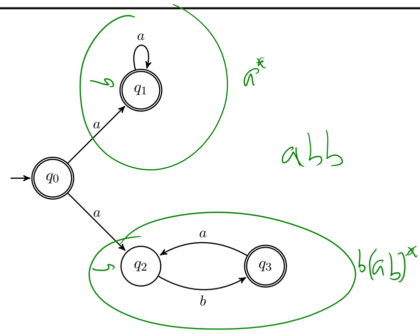
Is it a DFA?

It is not a DFA. It has two “out arrows” from q0 and no out arrows from q1.

What language does it accept?

NFAs accept whenever *any* “token” ends in an accept state. So, as long as the last character is an a, some “token” will end in q1.

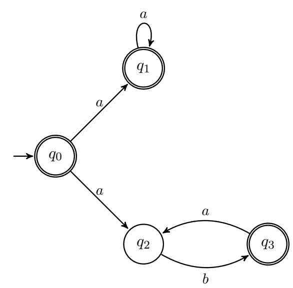
Consider This NFA



What language does this NFA accept?

$\varepsilon \cup a(a^* \cup (b(ab)^*))$

Consider This NFA



What language does this NFA accept?

This is a "union" NFA. When a string starts with an a, both "machines" see if it accepts. The top one is any number of a's. The bottom one is repetitions of "ab".