CSE 311 Lecture 24:
FSM Minimization and NFAs

Emina Torlak and Kevin Zatloukal

Topics

FSM minimization

Algorithm and examples.

Nondeterministic finite automata (NFAs)

Definition and examples.

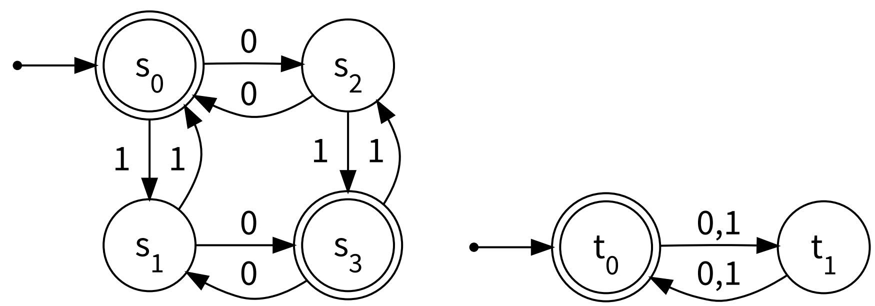
FSM minimization

Algorithm and examples.

How would you compare two FSMs?

Suppose we are given two FSMs over the same input alphabet.

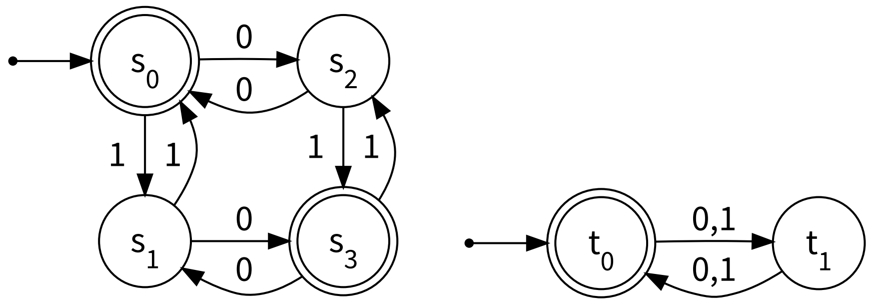
How can we tell if they are *equivalent*, i.e., accept the same language?



How would you compare two FSMs?

Suppose we are given two FSMs over the same input alphabet.

How can we tell if they are *equivalent*, i.e., accept the same language?



This is exactly the question answered by `grinch`! And many important practical applications beyond grading homeworks, e.g., efficient implementation of `grep` :)

FSM minimization

Unique minimal FSM

Each finite state machine M has a *unique minimal equivalent machine* (up to a renaming of states) $M_{\min}$ that accepts the same language as M.

FSM minimization

Unique minimal FSM

Each finite state machine M has a *unique minimal equivalent machine* (up to a renaming of states) $M_{\min}$ that accepts the same language as M.

The above definition means that ...

No FSM that is equivalent to M has fewer states than $M_{\min}$.

If an FSM is equivalent to M and has the same number of states as $M_{\min}$, then it is equal to $M_{\min}$ up to a renaming of states.

FSM minimization

Unique minimal FSM

Each finite state machine M has a *unique minimal equivalent machine* (up to a renaming of states) $M_{\min}$ that accepts the same language as M.

The above definition means that ...

No FSM that is equivalent to M has fewer states than $M_{\min}$.

If an FSM is equivalent to M and has the same number of states as $M_{\min}$, then it is equal to $M_{\min}$ up to a renaming of states.

So to check if two FSMs are equivalent:

(1) Minimize them, and

(2) Compare the minimal FSMs for equality (modulo renaming of states).

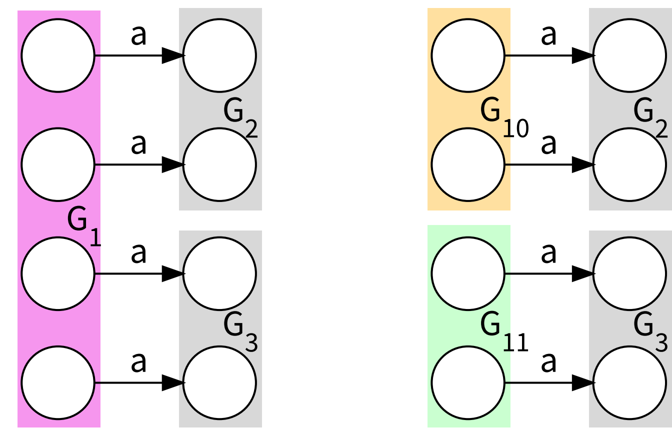
FSM minimization algorithm

1. Given an FSM, partition its states into groups according to their outputs (for FSMs with output) or whether they are final states or not (for DFAs).

FSM minimization algorithm

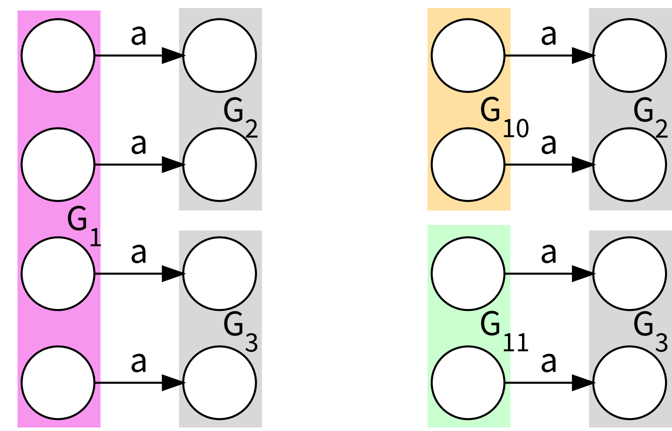
1. Given an FSM, partition its states into groups according to their outputs (for FSMs with output) or whether they are final states or not (for DFAs).
2. Repeat the following until *fixed point* (no change happens):

   - If there is a symbol a in a source group G such that some states in G disagree on which target group a leads to, partition G into smaller groups based on the target group for a.



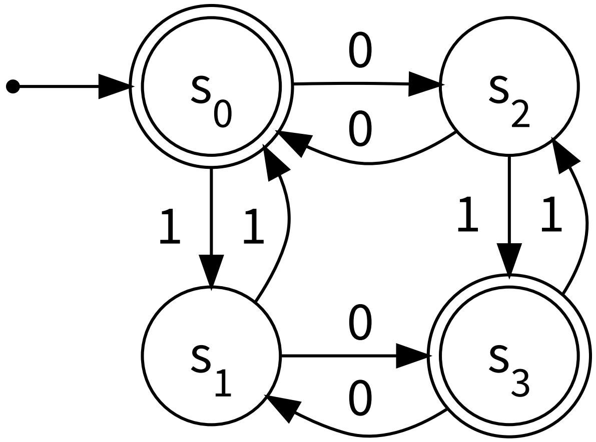
FSM minimization algorithm

1. Given an FSM, partition its states into groups according to their outputs (for FSMs with output) or whether they are final states or not (for DFAs).
2. Repeat the following until *fixed point* (no change happens):
   - If there is a symbol a in a source group G such that some states in G disagree on which target group a leads to, partition G into smaller groups based on the target group for a.

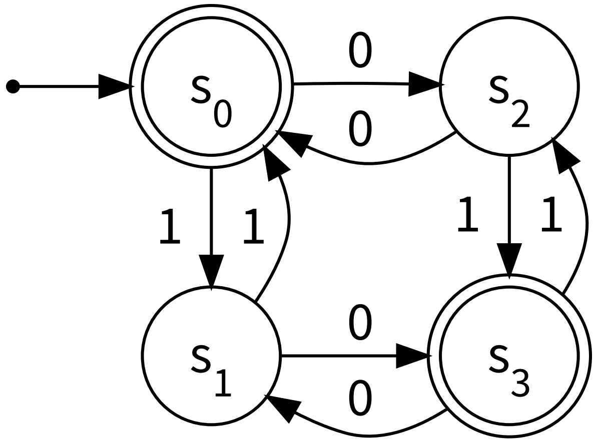


3. Convert groups to states and collapse edges with corresponding labels.

Example: minimizing a DFA

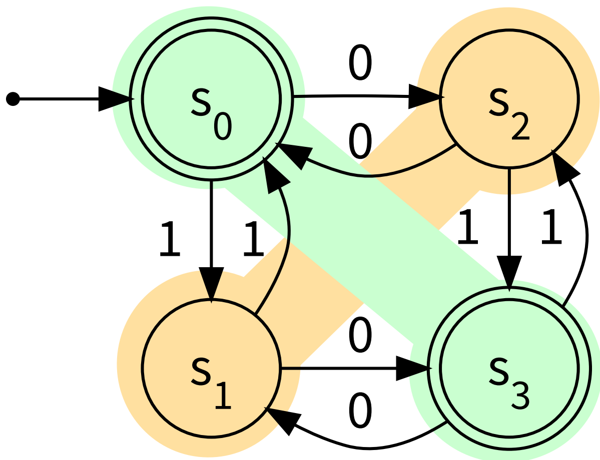


Example: minimizing a DFA



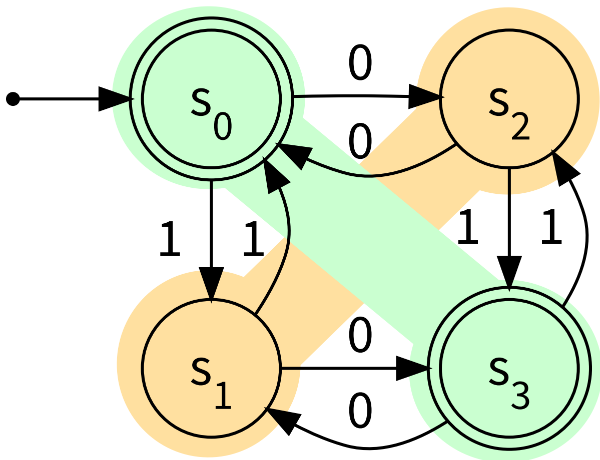
1. Partition the states into groups according to whether they are final states or not.

Example: minimizing a DFA



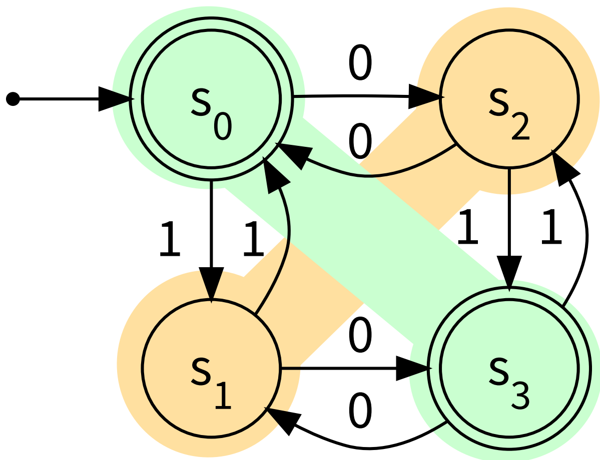
1. Partition the states into groups according to whether they are final states or not.

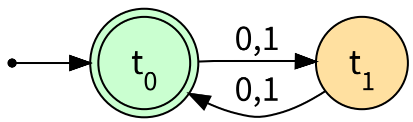
Example: minimizing a DFA



1. Partition the states into groups according to whether they are final states or not.
2. Every symbol causes the DFA to go from one group to the other so no partitioning needed.

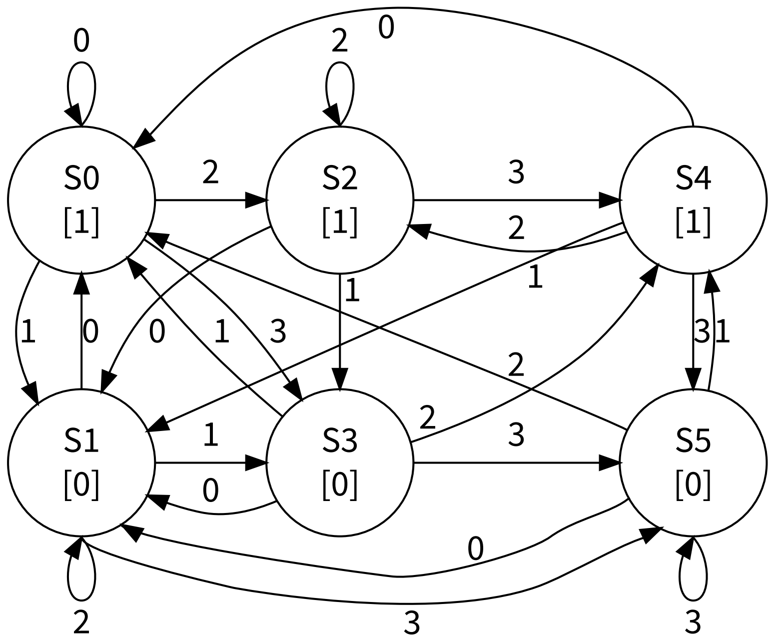
Example: minimizing a DFA



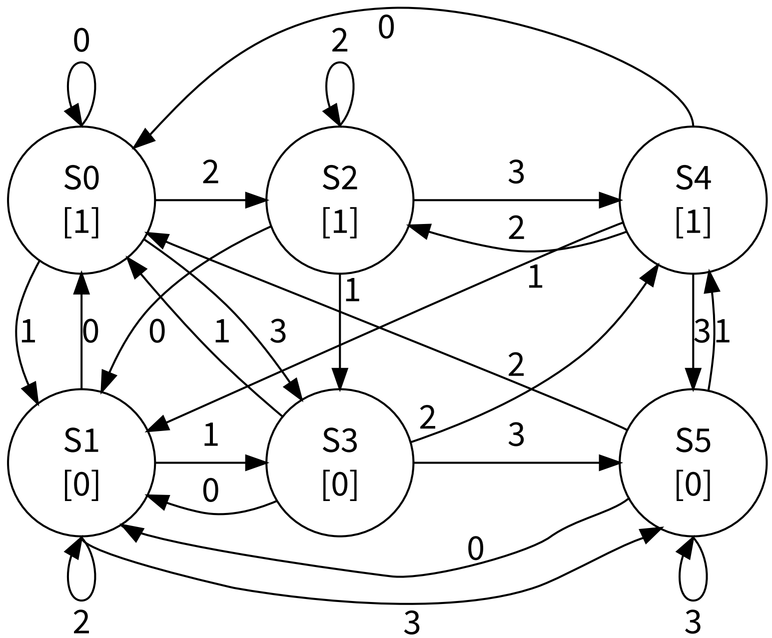


1. Partition the states into groups according to whether they are final states or not.
2. Every symbol causes the DFA to go from one group to the other so no partitioning needed.
3. Convert groups to states and collapse edges with corresponding labels.

Example: minimizing an FSM with output

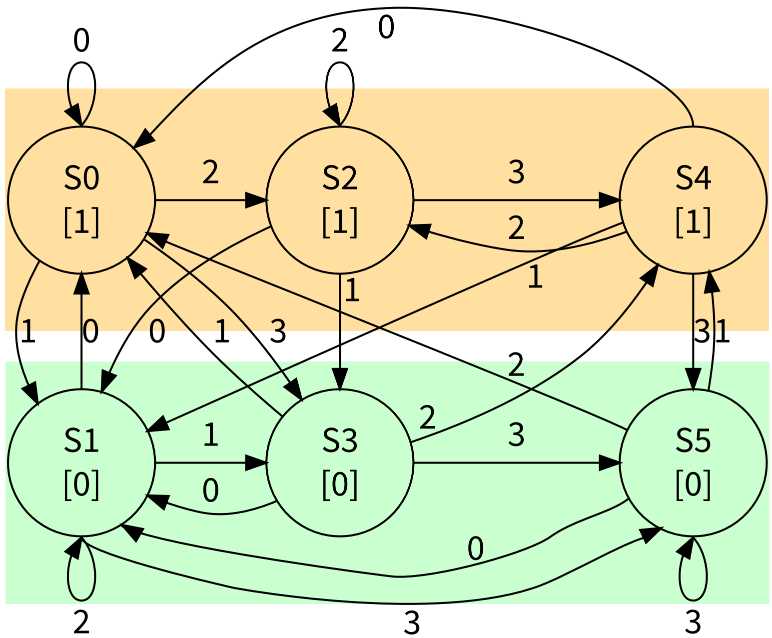


Example: minimizing an FSM with output



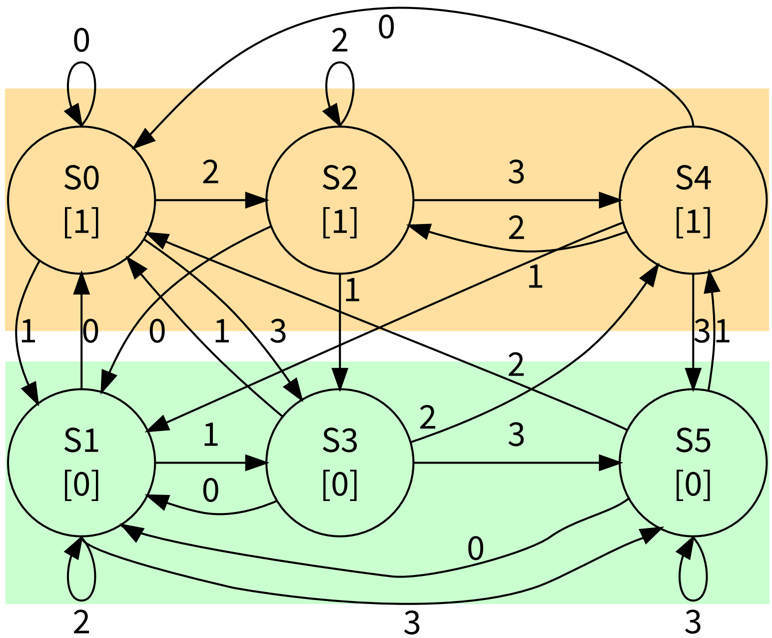
1. Partition the states into groups according to their outputs.

Example: minimizing an FSM with output



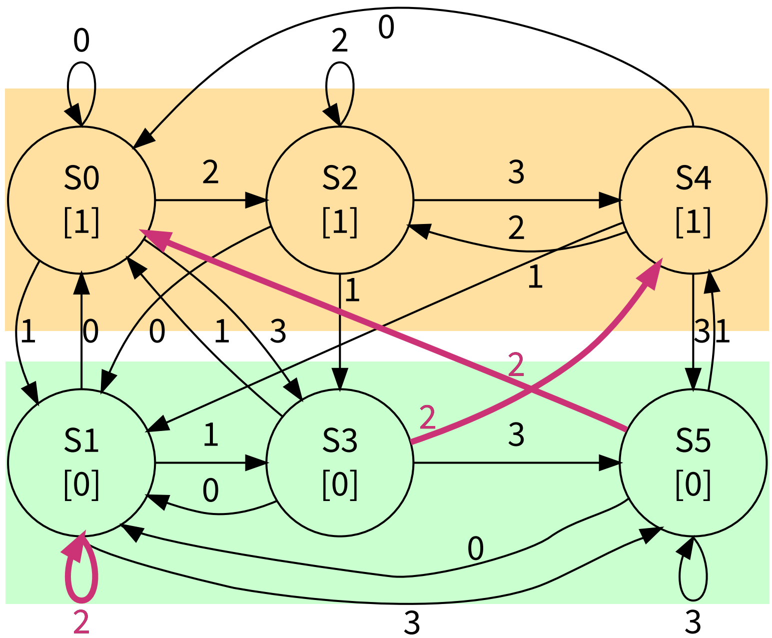
1. Partition the states into groups according to their outputs.

Example: minimizing an FSM with output



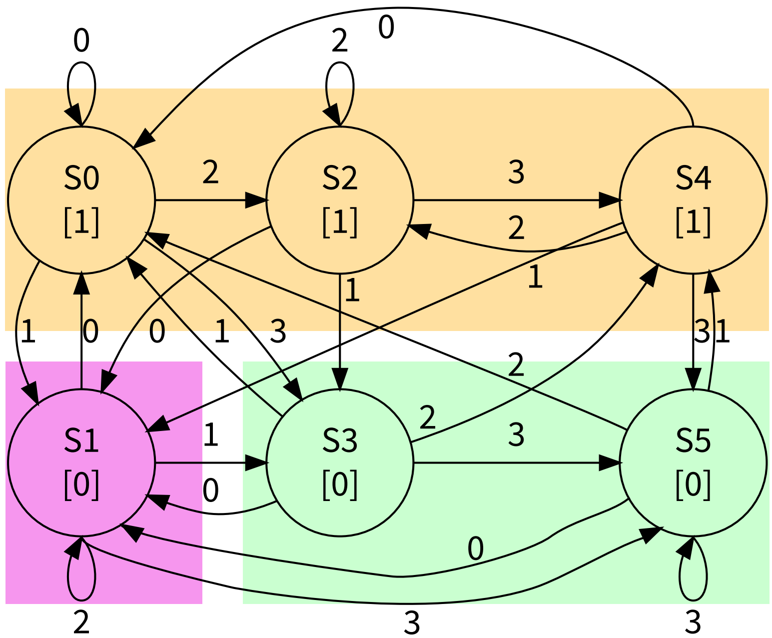
1. Partition the states into groups according to their outputs.
2. Repeat until fixed point:
   If there is a symbol a in a source group G such that some states in G disagree on which target group a leads to, partition G into smaller groups based on the target group for a.

Example: minimizing an FSM with output



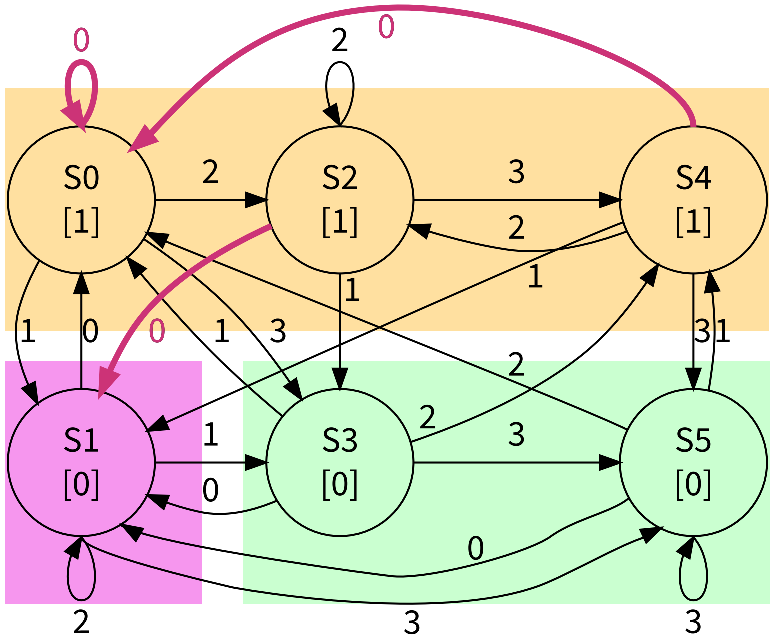
1. Partition the states into groups according to their outputs.
2. Repeat until fixed point:
   If there is a symbol a in a source group G such that some states in G disagree on which target group a leads to, partition G into smaller groups based on the target group for a.

Example: minimizing an FSM with output



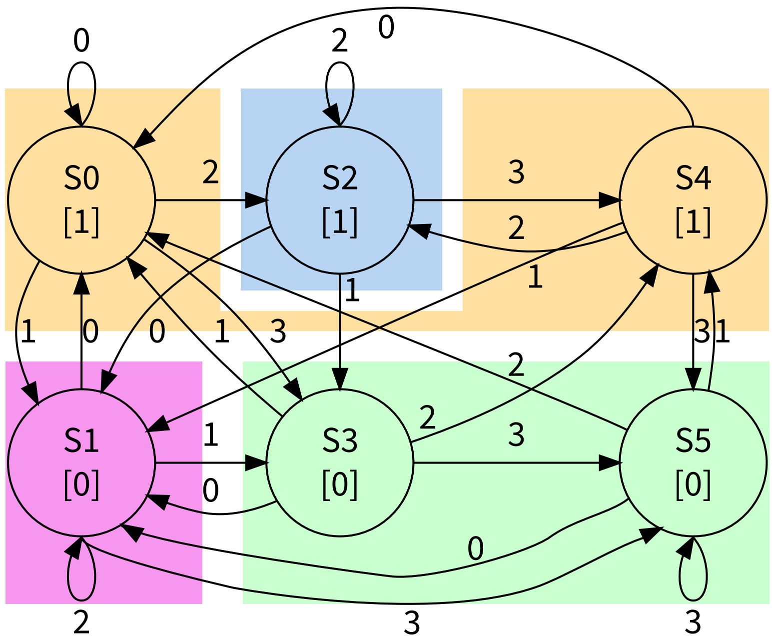
1. Partition the states into groups according to their outputs.
2. Repeat until fixed point:
   If there is a symbol a in a source group G such that some states in G disagree on which target group a leads to, partition G into smaller groups based on the target group for a.

Example: minimizing an FSM with output



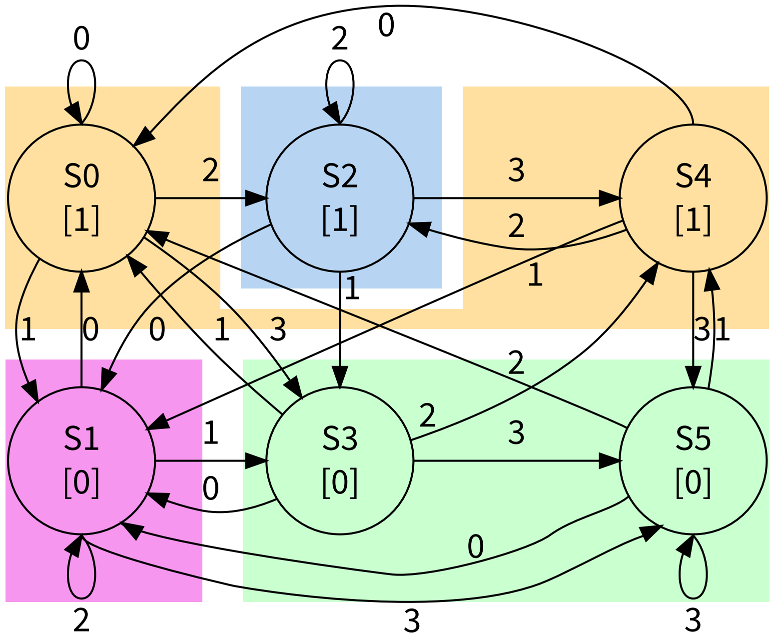
1. Partition the states into groups according to their outputs.
2. Repeat until fixed point:
   If there is a symbol a in a source group G such that some states in G disagree on which target group a leads to, partition G into smaller groups based on the target group for a.

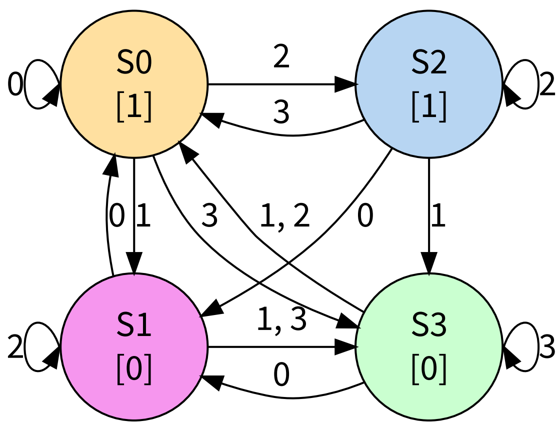
Example: minimizing an FSM with output



1. Partition the states into groups according to their outputs.
2. Repeat until fixed point:
   If there is a symbol a in a source group G such that some states in G disagree on which target group a leads to, partition G into smaller groups based on the target group for a.

Example: minimizing an FSM with output





1. Partition the states into groups according to their outputs.
2. Repeat until fixed point:
   If there is a symbol a in a source group G such that some states in G disagree on which target group a leads to, partition G into smaller groups based on the target group for a.
3. Convert groups to states and collapse edges with corresponding labels.

Nondeterministic finite automata (NFAs)

Definition and examples.

Recall the definition of a DFA

Deterministic finite automaton (DFA)

A *deterministic finite automaton* (DFA) $M = (S, \Sigma_{\text{in}}, f, s_0, F)$ consists of a finite set of *states* S, a finite *input alphabet* Σ_{in}, a *transition function* f that maps each state in S and input in Σ_{in} to a state in S, a *start state* $s_0 \in S$, and a set of *final states* $F \subseteq S$.

Recall the definition of a DFA

Deterministic finite automaton (DFA)

A *deterministic finite automaton* (DFA) $M = (S, \Sigma_{\text{in}}, f, s_0, F)$ consists of a finite set of *states* S, a finite *input alphabet* Σ_{in}, a *transition function* f that maps each state in S and input in Σ_{in} to a state in S, a *start state* $s_0 \in S$, and a set of *final states* $F \subseteq S$.

The label of a path in a DFA

The label of a path in a DFA is the concatenation of the labels of its edges in order.

Recall the definition of a DFA

Deterministic finite automaton (DFA)

A *deterministic finite automaton* (DFA) $M = (S, \Sigma_{\text{in}}, f, s_0, F)$ consists of a finite set of *states* S, a finite *input alphabet* Σ_{in}, a *transition function* f that maps each state in S and input in Σ_{in} to a state in S, a *start state* $s_0 \in S$, and a set of *final states* $F \subseteq S$.

The label of a path in a DFA

The label of a path in a DFA is the concatenation of the labels of its edges in order.

Lemma: strings recognized by a DFA

A string x is in the language recognized by a DFA if and only if x labels a path from the start state to some final state.

Recall the definition of a DFA

Deterministic finite automaton (DFA)

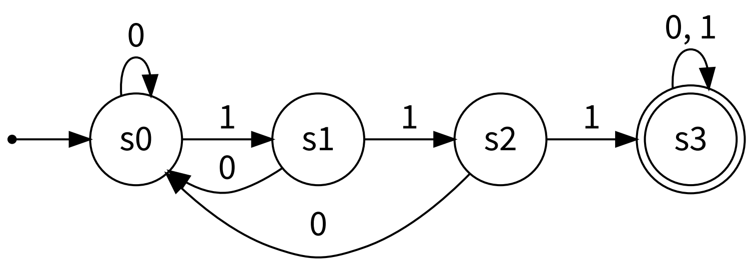
A *deterministic finite automaton* (DFA) $M = (S, \Sigma_{\text{in}}, f, s_0, F)$ consists of a finite set of *states* S, a finite *input alphabet* Σ_{in}, a *transition function* f that maps each state in S and input in Σ_{in} to a state in S, a *start state* $s_0 \in S$, and a set of *final states* $F \subseteq S$.

The label of a path in a DFA

The label of a path in a DFA is the concatenation of the labels of its edges in order.

Lemma: strings recognized by a DFA

A string x is in the language recognized by a DFA if and only if x labels a path from the start state to some final state.



What language does this DFA accept?

Recall the definition of a DFA

Deterministic finite automaton (DFA)

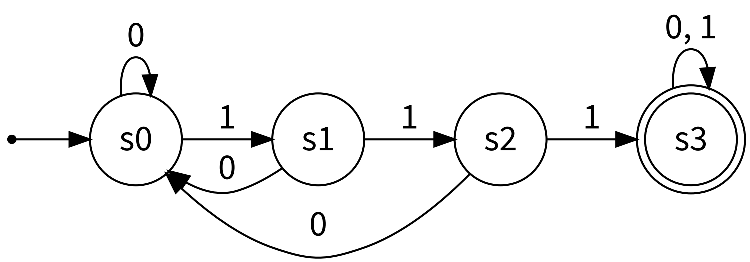
A *deterministic finite automaton* (DFA) $M = (S, \Sigma_{\text{in}}, f, s_0, F)$ consists of a finite set of *states* S, a finite *input alphabet* Σ_{in}, a *transition function* f that maps each state in S and input in Σ_{in} to a state in S, a *start state* $s_0 \in S$, and a set of *final states* $F \subseteq S$.

The label of a path in a DFA

The label of a path in a DFA is the concatenation of the labels of its edges in order.

Lemma: strings recognized by a DFA

A string x is in the language recognized by a DFA if and only if x labels a path from the start state to some final state.



What language does this DFA accept?

The set of all binary strings that contain 111.

Defining an NFA

Nondeterministic finite automaton (NFA)

A *nondeterministic finite automaton* (NFA) $M = (S, \Sigma_{\text{in}} \cup \varepsilon, f, s_0, F)$ consists of a finite set of *states* S, a finite *input alphabet* Σ_{in}, a *transition function* f that maps each state in S and input in $\Sigma_{\text{in}} \cup \varepsilon$ to a set of states in S, a *start state* $s_0 \in S$, and a set of *final states* $F \subseteq S$.

The label of a path in an NFA

The label of a path in an NFA is the concatenation of the labels of its edges in order.

Definition: strings recognized by an NFA

A string 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.

Defining an NFA

Nondeterministic finite automaton (NFA)

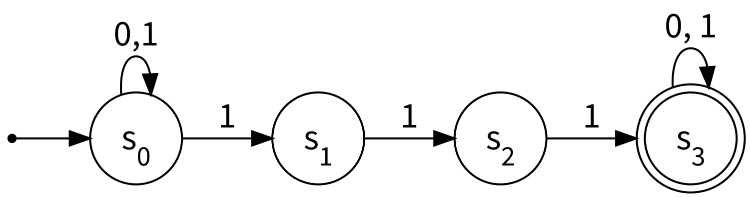
A *nondeterministic finite automaton* (NFA) $M = (S, \Sigma_{\text{in}} \cup \varepsilon, f, s_0, F)$ consists of a finite set of *states* S, a finite *input alphabet* Σ_{in}, a *transition function* f that maps each state in S and input in $\Sigma_{\text{in}} \cup \varepsilon$ to a set of states in S, a *start state* $s_0 \in S$, and a set of *final states* $F \subseteq S$.

The label of a path in an NFA

The label of a path in an NFA is the concatenation of the labels of its edges in order.

Definition: strings recognized by an NFA

A string 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.



What language does this NFA accept?

Defining an NFA

Nondeterministic finite automaton (NFA)

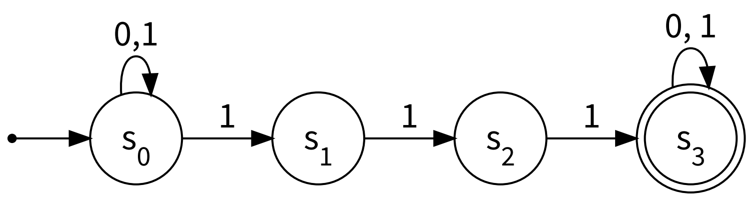
A *nondeterministic finite automaton* (NFA) $M = (S, \Sigma_{\text{in}} \cup \varepsilon, f, s_0, F)$ consists of a finite set of *states* S, a finite *input alphabet* Σ_{in}, a *transition function* f that maps each state in S and input in $\Sigma_{\text{in}} \cup \varepsilon$ to a set of states in S, a *start state* $s_0 \in S$, and a set of *final states* $F \subseteq S$.

The label of a path in an NFA

The label of a path in an NFA is the concatenation of the labels of its edges in order.

Definition: strings recognized by an NFA

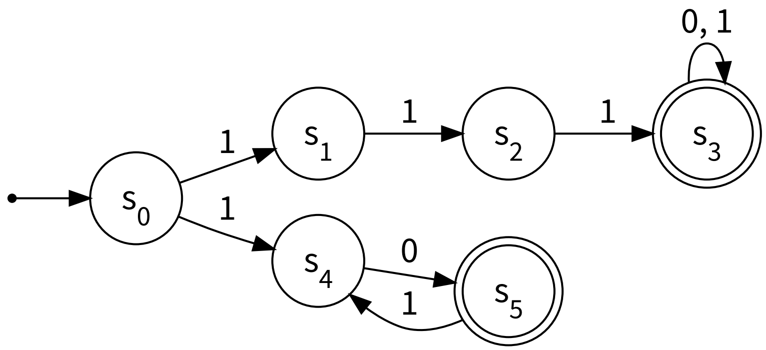
A string x is in the language recognized by an NFA if and only if x labels a path from the start state to some final state.



What language does this NFA accept?

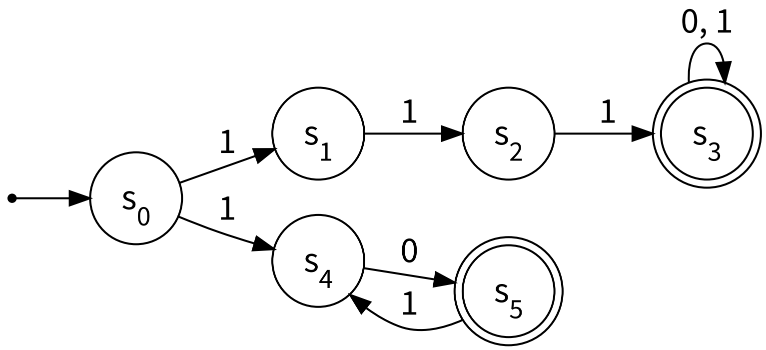
The set of all binary strings that contain 111.

Example NFAs



What language does this NFA accept?

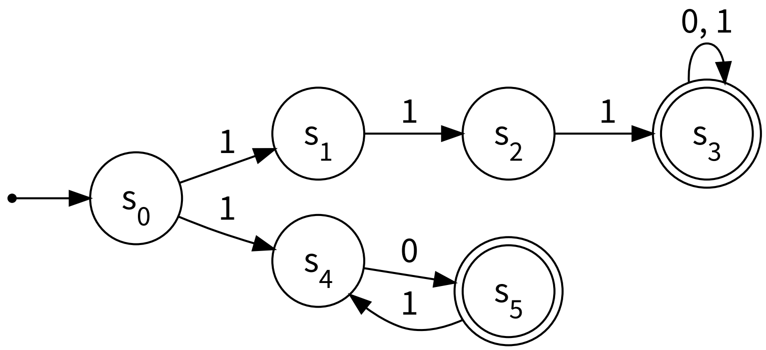
Example NFAs



What language does this NFA accept?

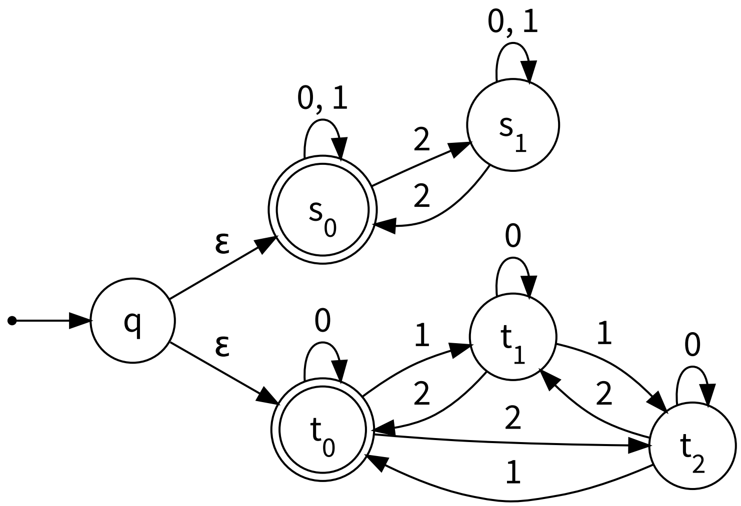
$$10(10)^* \cup 111(0 \cup 1)^*$$

Example NFAs



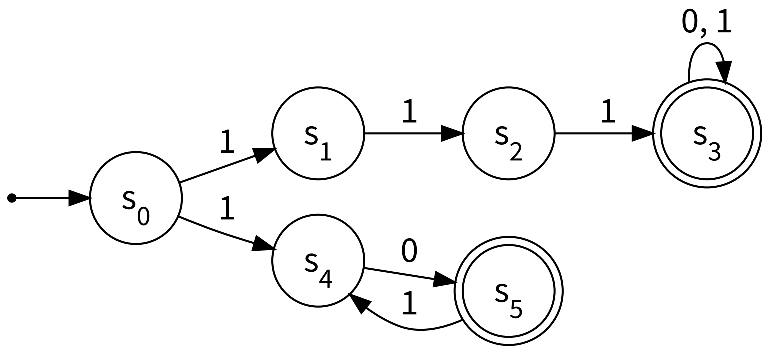
What language does this NFA accept?

$10(10)^* \cup 111(0 \cup 1)^*$



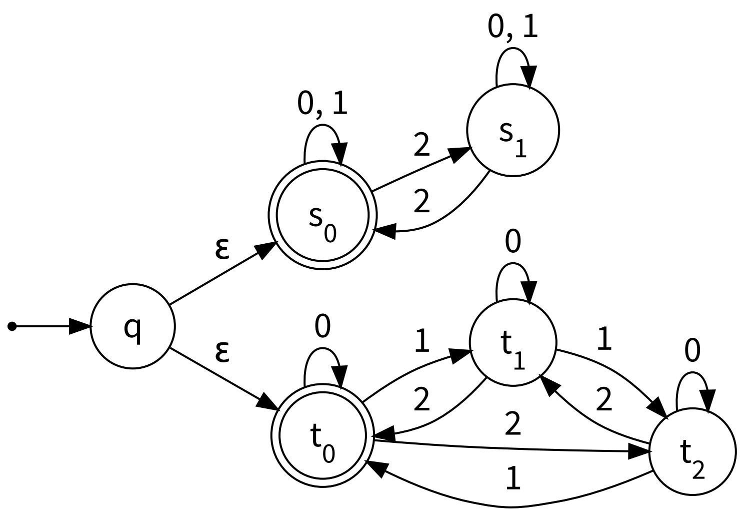
What language does this NFA accept?

Example NFAs



What language does this NFA accept?

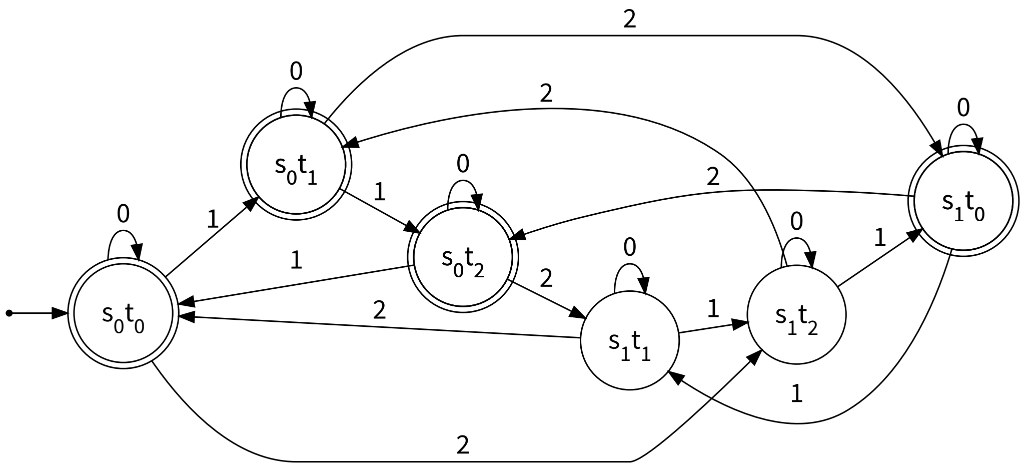
$10(10)^* \cup 111(0 \cup 1)^*$

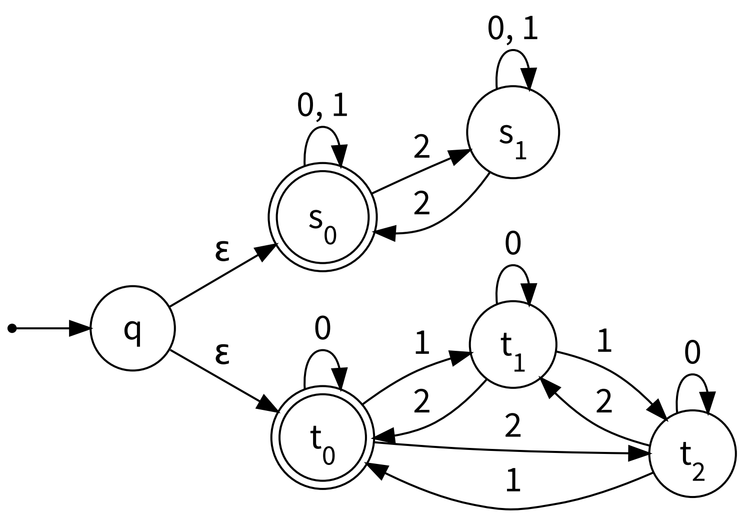


What language does this NFA accept?

Strings over {0, 1, 2} with an even number of 2's *or* with digits summing to 0 mod 3.

NFAs make it easy to union languages

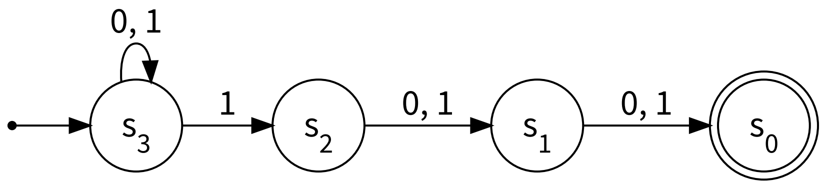




A DFA and NFA for the same language.

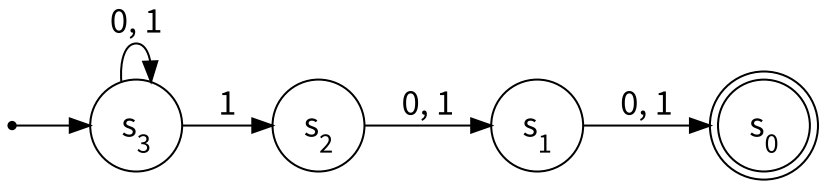
Strings over {0, 1, 2} with an even number of 2's *or* with digits summing to 0 mod 3.

NFAs can be much smaller than DFAs



What language does this NFA accept?

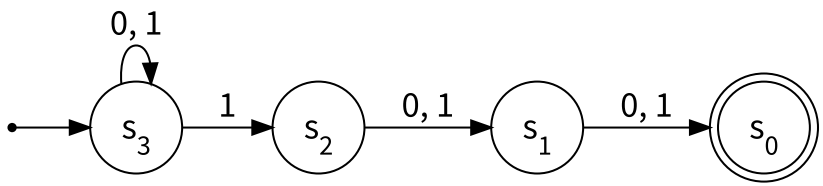
NFAs can be much smaller than DFAs



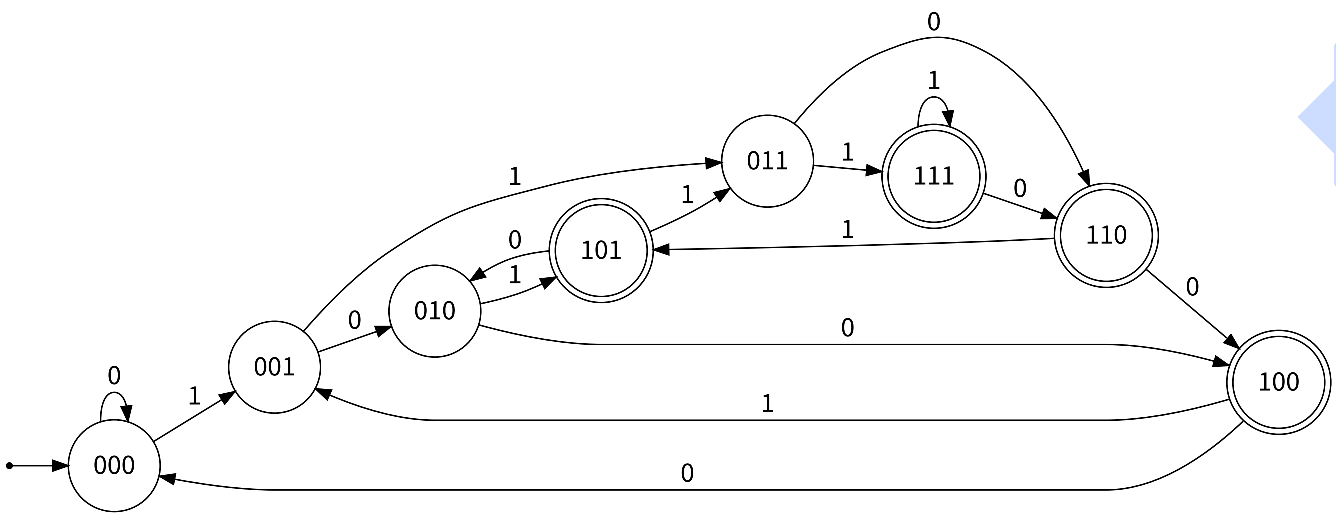
What language does this NFA accept?

Binary strings with a 1 in the 3rd position from the end.

NFAs can be much smaller than DFAs



What language does this NFA accept?
Binary strings with a 1 in the 3rd position from the end.



The smallest DFA that accepts the same language.

Three ways to understand NFAs

Outside observer

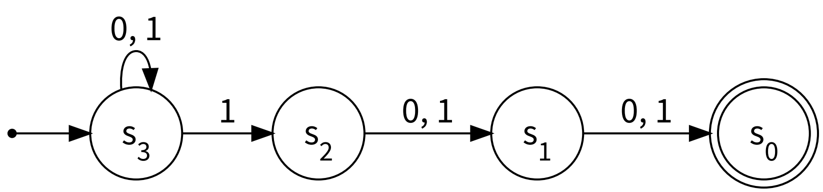
Is there a path labeled by x from the start state to some final state?

Perfect guesser (oracle)

Given an input x, the NFA guesses the right edge to take (if one exists) whenever there is a choice to be made.

Parallel exploration

The NFA runs all possible computations on x in parallel.



Summary

Every FSM has a unique minimal equivalent FSM (modulo state names).

- We can compute the minimal FSM using the algorithm from this lecture.
- We can use this to check if two FSMs are equivalent!

An NFA recognizes a set of strings (language).

- An NFA differs from a DFA in that the transition function maps each state and input symbol to a *set* of states.
- Determining if an NFA accepts string boils down to checking if there is a path from the start state to some final state, where the path consists of the edges labeled by the string's characters.