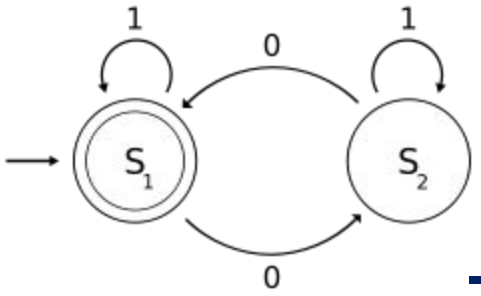
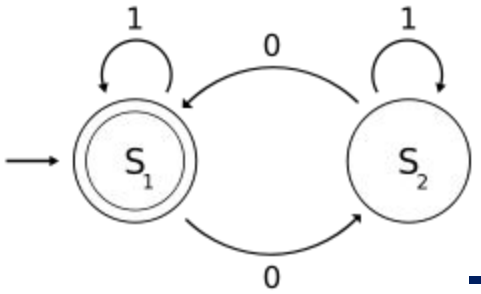


Equivalence of NFAs and DFAs

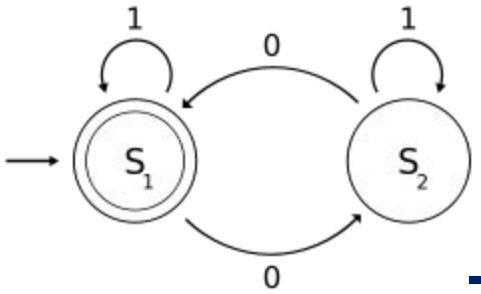
Sipser: Section 1.2 pages 47 - 54



Today's adventure

- Reminder of community norms
- Nondeterministic computation and examples
- Equivalence of DFAs and NFAs

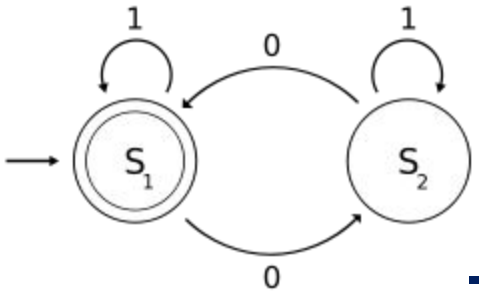
Weekly tip:



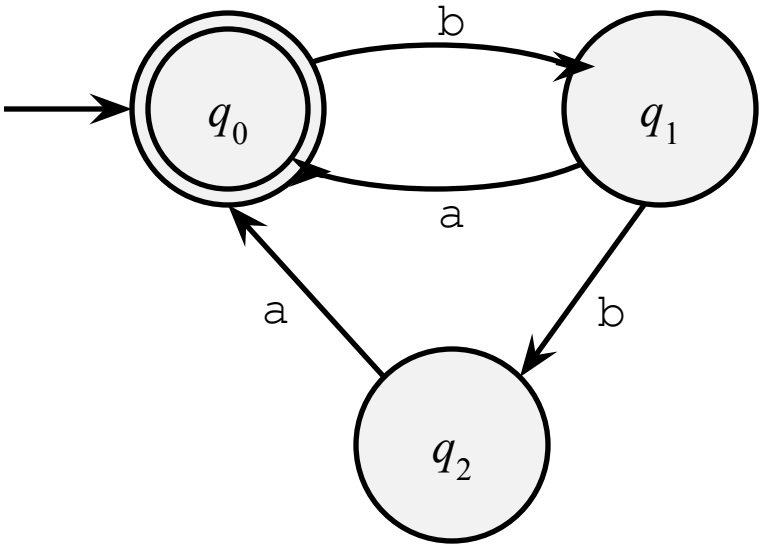
Community norms

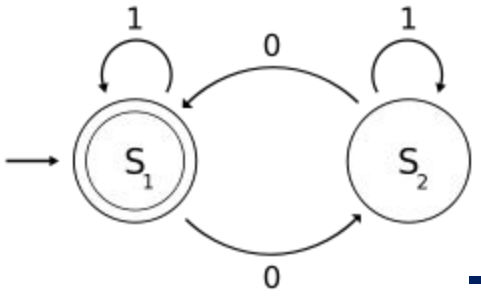
This course will strive to create an inclusive learning environment in which everyone feels like they belong in the class. The following norms are designed to help us collectively reach that ideal. Please make a good effort to live out these norms throughout the semester.

- Be present, and strive to be your best self
- Listen with the possibility of being changed. Speak with the promise of being heard.
- Everyone has something to learn.
- Everyone has expertise to offer.
- You have the right to ask for help, and the duty to assist.
- Be willing to experience discomfort.
- Expect and accept non-closure.

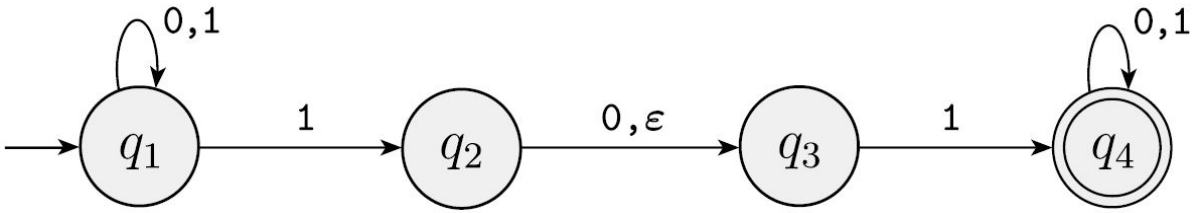


Another Example



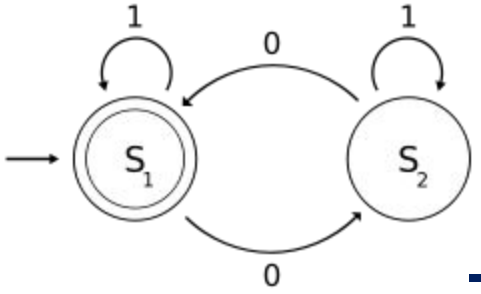


Formally

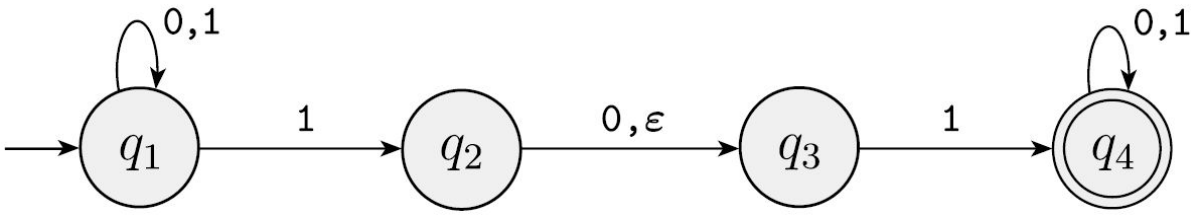


A **nondeterministic finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the **states**,
2. Σ is a finite set called the **alphabet**,
3. $\delta: Q \times \Sigma_\varepsilon \rightarrow P(Q)$ is the **transition function**,
4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the **set of accept states**.

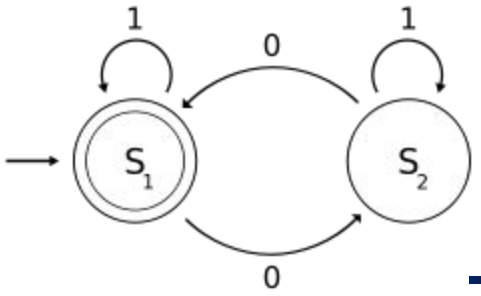


Nondeterministic Automata Computation



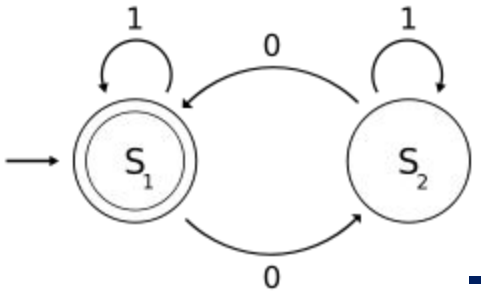
Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA and w a string over the alphabet Σ. Then N **accepts** w if we can write w as $w = y_1 y_2 \cdots y_m$, where each y_i is a member of Σ_ε and a sequence of states $r_0, r_1, \ldots, r_m$ exists in Q with three conditions:

1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i = 0, \ldots, m\text{-}1$, and
3. $r_m \in F$.



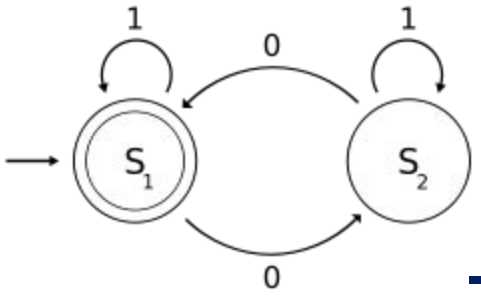
Nondeterminism is Your Friend

Build an NFA that recognizes the language $B = \{ w \mid w$ is a string of *a*s and *b*s that starts and ends with the same symbol and contains at least two symbols $\}$.



Warmup

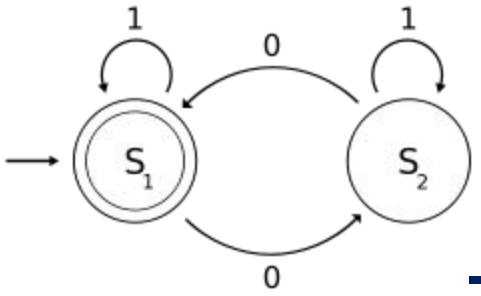
Build an NFA that recognizes the language $\{0^k \mid k \text{ is a multiple of 2 or 3}\}$.



DFAs and NFAs are equivalent

Theorem. A language is regular if and only if it is accepted by a nondeterministic finite automaton.

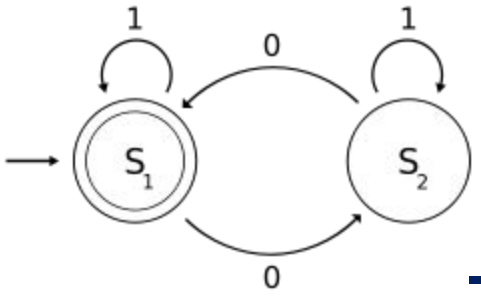
Proof. ($\Rightarrow$) Let A be a regular language ...



Recall DFA definition

A **finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

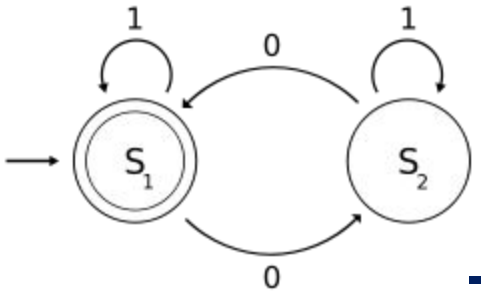
1. Q is a finite set called the **states**,
2. Σ is a finite set called the **alphabet**,
3. $\delta: Q \times \Sigma \rightarrow Q$ is the **transition function**,
4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the **set of accept states**.



Recall NFA definition

A **nondeterministic finite automaton** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the **states**,
2. Σ is a finite set called the **alphabet**,
3. $\delta: Q \times \Sigma_\varepsilon \rightarrow P(Q)$ is the **transition function**,
4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the **set of accept states**.



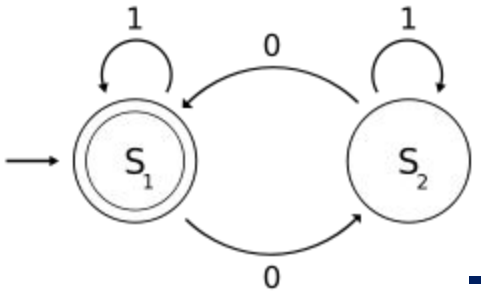
The other direction is more fun!

Theorem. A language is regular if and only if it is accepted by a nondeterministic finite automaton.

Proof. (⇐) Suppose *A* is accepted by a nondeterministic finite automaton. We want to show there is a DFA that accepts A...

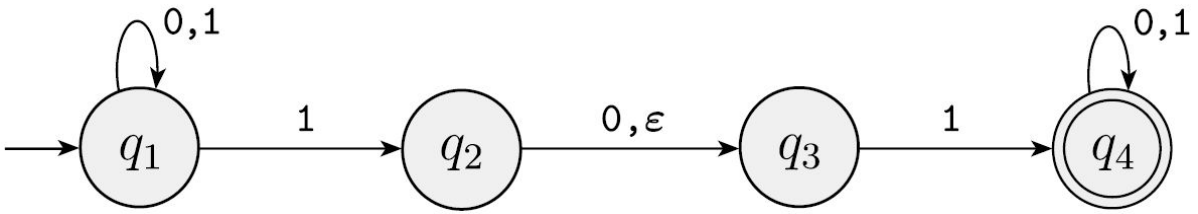
Definition. Two machines are ***equivalent*** if they recognize the same language.

So actually, we'll prove every NFA has an equivalent DFA!

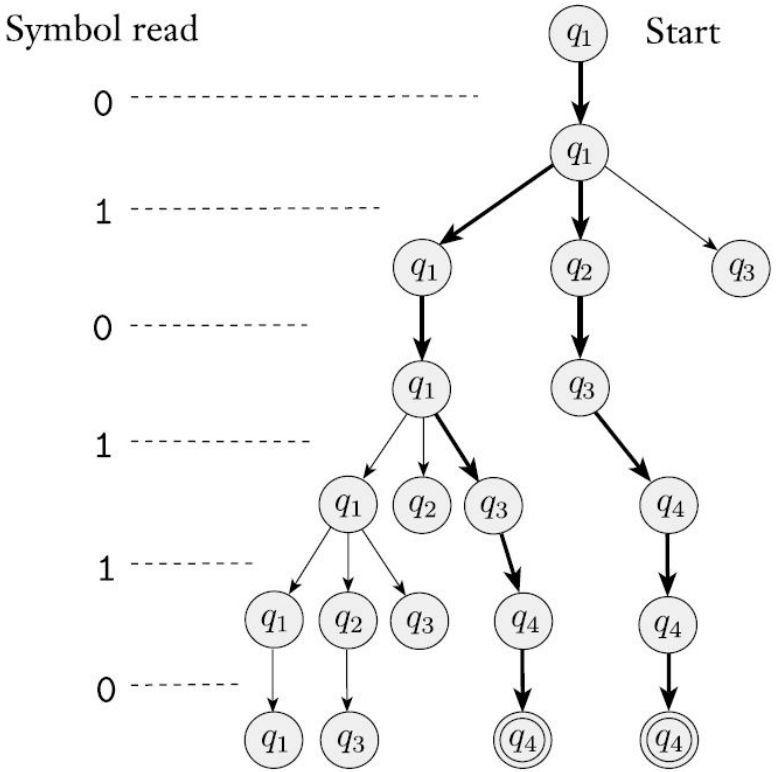


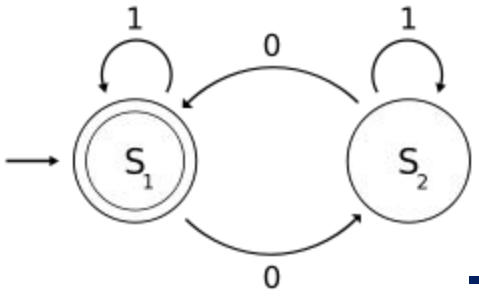
You Only Go Around Once

N_1

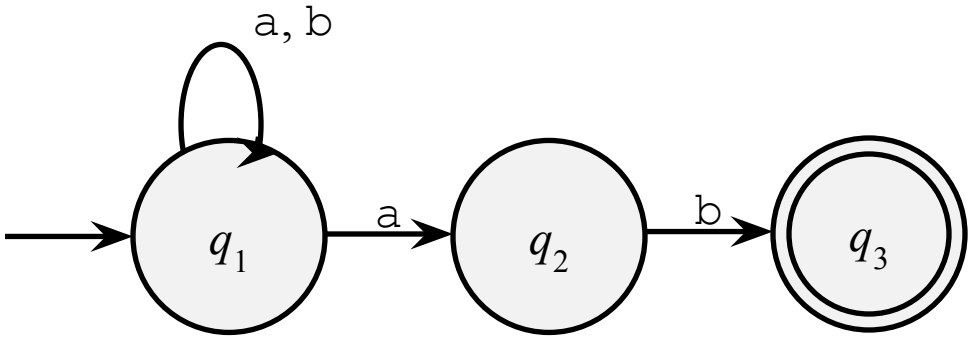


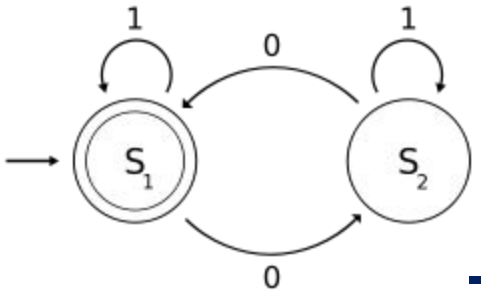
Input 010110





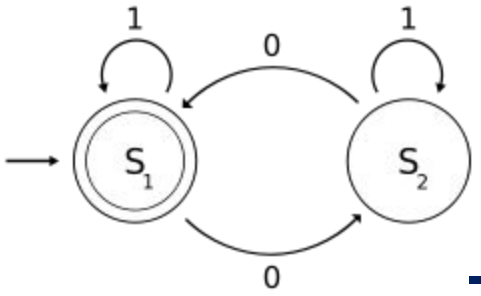
A Simpler Example



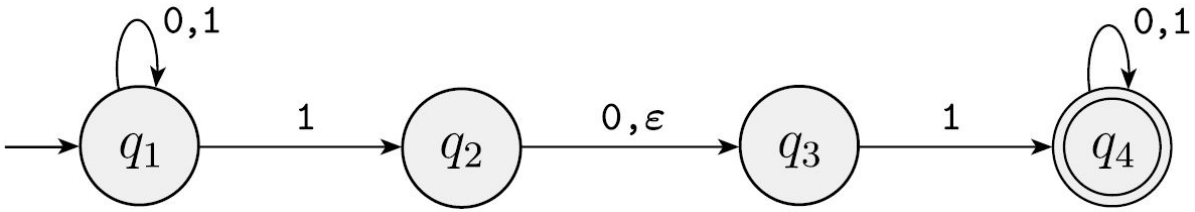


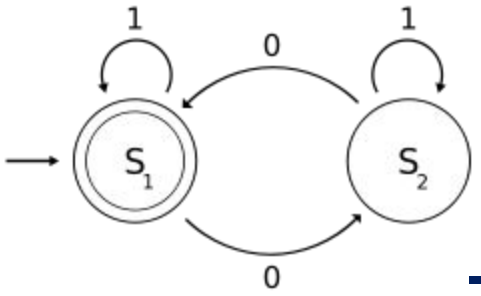
First Stage: Removing Choice

Almost Proof.



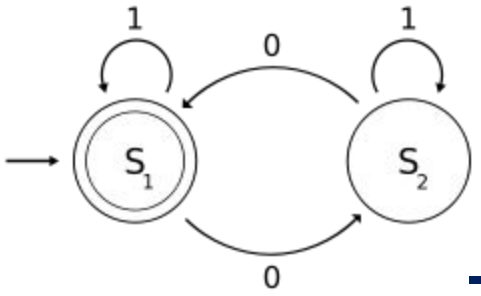
Second stage: Deal with ε Arrows





Modifying Our Construction

Proof.



Equivalent DFA?