Week2 monday

Review: Formal definition of DFA: $M = (Q, \Sigma, \delta, q_0, F)$

- Finite set of states Q
- Alphabet Σ
- Transition function δ
- Start state q_0
- Accept (final) states F

In the state diagram of M, how many outgoing arrows are there from each state?

$M = (\{q, r, s\}, \{a, b\}, \delta, q, \{s\})$ where δ is (rows labelled by states and columns labelled by symbols):

δ	a	b
q	r	q
r	r	s
s	s	s

The state diagram for M is

Give two examples of strings that are accepted by M and two examples of strings that are rejected by M:

Add "labels" for states in the state diagram, e.g. "have not seen any of desired pattern yet" or "sink state".

We can use the analysis of the roles of the states in the state diagram to describe the language recognized by the DFA.

$L(M) =$

A regular expression describing $L(M)$ is

Let the alphabet be $\Sigma_1 = \{0, 1\}$.

A state diagram for a DFA that recognizes $\{w \mid w \text{ contains at most two 1's}\}$ is

A state diagram for a DFA that recognizes $\{w \mid w \text{ contains more than two 1's}\}$ is

Extra example: A state diagram for DFA recognizing

$$\{w \mid w \text{ is a string over } \{0,1\} \text{ whose length is not a multiple of } 3\}$$

Let n be an arbitrary positive integer. What is a formal definition for a DFA recognizing

$$\{w \mid w \text{ is a string over } \{0,1\} \text{ whose length is not a multiple of } n\}?$$

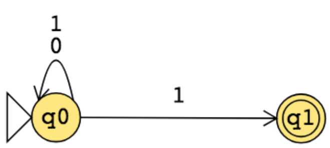
Note: On Wednesday, we'll see a new kind of finite automaton. It will be helpful to distinguish it from the machines we've been talking about so we'll use **Deterministic Finite Automaton** (DFA) to refer to the machines from Section 1.1.

Week2 wednesday

Nondeterministic finite automaton (Sipser Page 53) Given as $M = (Q, \Sigma, \delta, q_0, F)$	
Finite set of states Q	Can be labelled by any collection of distinct names. Default: $q0, q1, \ldots$
Alphabet Σ	Each input to the automaton is a string over Σ.
Arrow labels Σ_ε	$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$. Arrows in the state diagram are labelled either by symbols from Σ or by ε
Transition function δ	$\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ gives the **set of possible next states** for a transition from the current state upon reading a symbol or spontaneously moving.
Start state q_0	Element of Q. Each computation of the machine starts at the start state.
Accept (final) states F	$F \subseteq Q$.

M accepts the input string $w \in \Sigma^*$ if and only if **there is** a computation of M on w that processes the whole string and ends in an accept state.

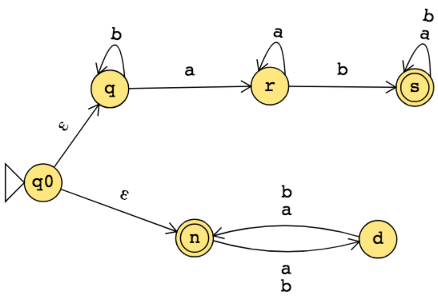
The formal definition of the NFA over $\{0, 1\}$ given by this state diagram is:



The language over $\{0, 1\}$ recognized by this NFA is:

Change the transition function to get a different NFA which accepts the empty string (and potentially other strings too).

The state diagram of an NFA over $\{a, b\}$ is below. The formal definition of this NFA is:



The language recognized by this NFA is:

Week2 friday

Warmup: Design a DFA (deterministic finite automaton) and an NFA (nondeterministic finite automaton) that each recognize each of the following languages over $\{a, b\}$

$$\{w \mid w \text{ has an } a \text{ and ends in } b\}$$

$$\{w \mid w \text{ has an } a \text{ or ends in } b\}$$

Strategy: To design DFA or NFA for a given language, identify patterns that can be built up as we process strings and create states for intermediate stages. Or: decompose the language to a simpler one that we already know how to recognize with a DFA or NFA.

Recall (from Wednesday of last week, and in textbook Exercise 1.14): if there is a DFA M such that $L(M) = A$ then there is another DFA, let's call it M', such that $L(M') = \overline{A}$, the complement of A, defined as $\{w \in \Sigma^* \mid w \notin A\}$.

Let's practice defining automata constructions by coming up with other ways to get new automata from old.

Suppose A_1, A_2 are languages over an alphabet Σ. **Claim:** if there is a NFA N_1 such that $L(N_1) = A_1$ and NFA N_2 such that $L(N_2) = A_2$, then there is another NFA, let's call it N, such that $L(N) = A_1 \cup A_2$.

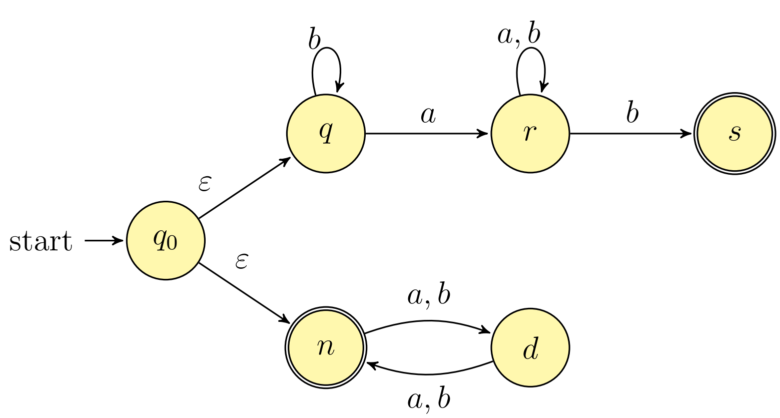
Proof idea: Use nondeterminism to choose which of N_1, N_2 to run.

Formal construction: Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ and assume $Q_1 \cap Q_2 = \emptyset$ and that $q_0 \notin Q_1 \cup Q_2$. Construct $N = (Q, \Sigma, \delta, q_0, F_1 \cup F_2)$ where

- $Q =$
- $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ is defined by, for $q \in Q$ and $x \in \Sigma_\varepsilon$:

Proof of correctness would prove that $L(N) = A_1 \cup A_2$ by considering an arbitrary string accepted by N, tracing an accepting computation of N on it, and using that trace to prove the string is in at least one of A_1, A_2; then, taking an arbitrary string in $A_1 \cup A_2$ and proving that it is accepted by N. Details left for extra practice.

Example: The language recognized by the NFA over $\{a, b\}$ with state diagram



is:

Could we do the same construction with DFA?

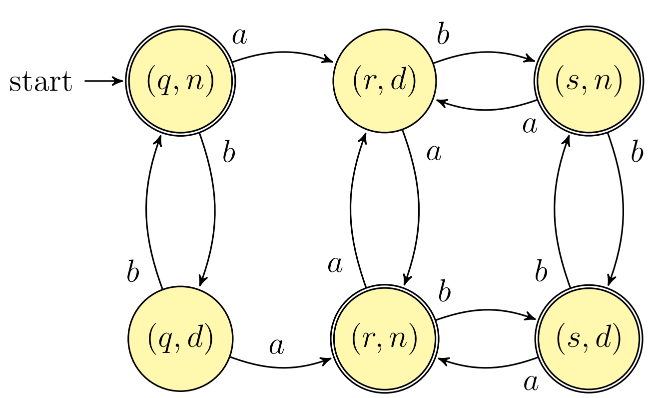
Happily, though, an analogous claim is true!

Suppose A_1, A_2 are languages over an alphabet Σ. **Claim:** if there is a DFA M_1 such that $L(M_1) = A_1$ and DFA M_2 such that $L(M_2) = A_2$, then there is another DFA, let's call it M, such that $L(M) = A_1 \cup A_2$. *Theorem 1.25 in Sipser, page 45*

Proof idea:

Formal construction:

Example: When $A_1 = \{w \mid w \text{ has an } a \text{ and ends in } b\}$ and $A_2 = \{w \mid w \text{ is of even length}\}$.



Suppose A_1, A_2 are languages over an alphabet Σ. **Claim:** if there is a DFA M_1 such that $L(M_1) = A_1$ and DFA M_2 such that $L(M_2) = A_2$, then there is another DFA, let's call it M, such that $L(M) = A_1 \cap A_2$. *Sipser Theorem 1.25, page 45*

Proof idea:

Formal construction:

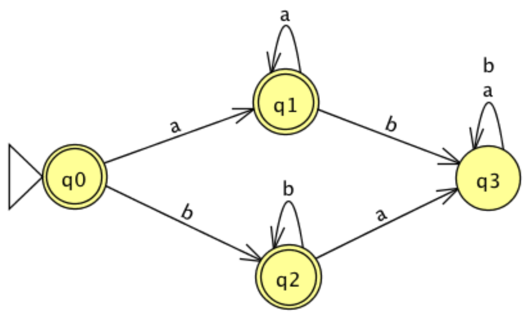
Week1 friday

This definition was in the pre-class reading A finite automaton (FA) is specified by $M = (Q, \Sigma, \delta, q_0, F)$. This 5-tuple is called the **formal definition** of the FA. The FA can also be represented by its state diagram: with nodes for the state, labelled edges specifying the transition function, and decorations on nodes denoting the start and accept states.

> Finite set of states Q can be labelled by any collection of distinct names. Often we use default state labels $q0, q1, \ldots$
>
> The alphabet Σ determines the possible inputs to the automaton. Each input to the automaton is a string over Σ, and the automaton "processes" the input one symbol (or character) at a time.
>
> The transition function δ gives the next state of the automaton based on the current state of the machine and on the next input symbol.
>
> The start state q_0 is an element of Q. Each computation of the machine starts at the start state.
>
> The accept (final) states F form a subset of the states of the automaton, $F \subseteq Q$. These states are used to flag if the machine accepts or rejects an input string.
>
> The computation of a machine on an input string is a sequence of states in the machine, starting with the start state, determined by transitions of the machine as it reads successive input symbols.
>
> The finite automaton M accepts the given input string exactly when the computation of M on the input string ends in an accept state. M rejects the given input string exactly when the computation of M on the input string ends in a nonaccept state, that is, a state that is not in F.
>
> The language of M, $L(M)$, is defined as the set of all strings that are each accepted by the machine M. Each string that is rejected by M is not in $L(M)$. The language of M is also called the language recognized by M.

What is **finite** about all finite automata? (Select all that apply)

- ☐ The size of the machine (number of states, number of arrows)
- ☐ The length of each computation of the machine
- ☐ The number of strings that are accepted by the machine

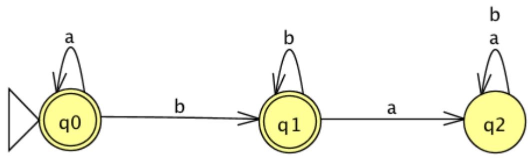


The formal definition of this FA is

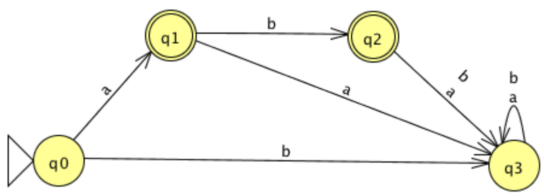
Classify each string $a, aa, ab, ba, bb, \varepsilon$ as accepted by the FA or rejected by the FA.

Why are these the only two options?

The language recognized by this automaton is



The language recognized by this automaton is



The language recognized by this automaton is