Finite Automata

A *finite automaton* is a mathematical model of a simple computing device with a finite amount of memory.

- Very simple, but of great use in practical applications:
  - Devices with simple behavior (vending machines, elevators, appliances, *etc.*)
  - Matching patterns in strings (editors, spam filters, search engines, compilers, *etc.*)
- Any physically realizable computer is technically a finite automaton.

Note: Singular: *“automaton”*, Plural: *“automata”*

Finite Automata as Language Recognizers

When used as a *language recognizer*, a finite automaton consists of:

- *Finite Control:* stores the internal state
- *Input Tape:* holds an input string to be scanned
- *Read Head:* marks the current input position.

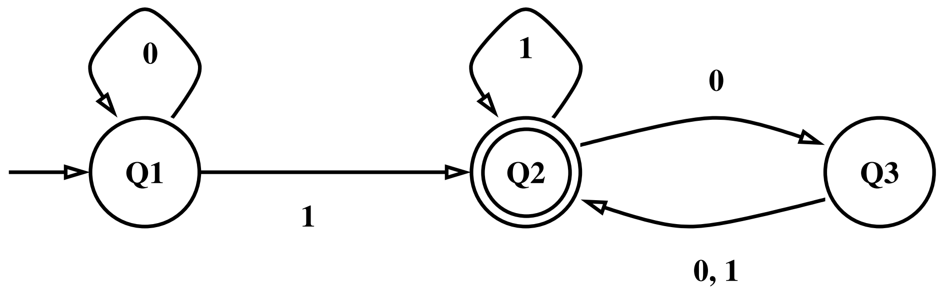
Operation:

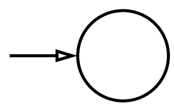
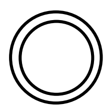
- The automaton scans the input one symbol at a time from left to right (no backing up).
- At each step, the finite control tells what state to go to, based on the current state and the input symbol being scanned.
- When the end of the input is reached, the input is either *accepted* or *rejected*, depending on the current state.

The *language recognized* by the automaton is the set of all input strings that the automaton accepts.

State Diagrams

Finite automata can be represented graphically as a *state diagram*:



- *Start state:* 
- *Accept state:* 
- *Transition:* 

What language does this recognize?

Other Ways to Model Computing Devices

Language recognizers are not the only way to model computing devices. Other possibilities are:

- *Language generators:* Symbol are emitted on transitions, instead of consuming them.
- *Transducers:* Symbols are both consumed and emitted on transitions.
- *Interactive:* The automaton interacts with its environment on each transition.

Interactive perhaps corresponds most closely to practice, but is not what is traditionally studied in ToC.

Formal Definition of Finite Automaton

Def: *(Sipser, Def. 1.5)* A (deterministic) *finite automaton* is a 5-tuple:

$$(Q, \Sigma, \delta, q_0, F),$$

where

- Q is a finite set called the *states*
- Σ is a finite set called the *alphabet*
- $\delta : Q \times \Sigma \to Q$ is the *transition function*
- $q_0 \in Q$ is the *start state*
- $F \subseteq Q$ is the set of *accept states*.

Notes

- Recall: $Q \times \Sigma$ denotes the *cartesian product* of Q and Σ, so δ maps a current state/current symbol pair (q, x) to the next state $\delta(q, x)$.

- Since $\delta : Q \times \Sigma \to Q$ is a function, for *every* (q, x) there is a *unique* r such that $\delta(q, x) = r$.

Formal Definition of the Language Recognized by a FA

Def: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a FA and let $w = w_1 w_2 \dots w_n$ over alphabet Σ. Then M *accepts* w if there exists a sequence of states $r_0, r_1, \dots, r_n$ in Q such that the following three conditions are satisfied:

1. $r_0 = q_0$
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, 1, \dots, n-1$
3. $r_n \in F$.

The set $\{w \in \Sigma^*. \; M \text{ accepts } w\}$ is called the *language recognized by M*.

Examples

- Define a FA that recognizes the empty language.
- Define a FA that recognizes Σ^*.
- Define a FA that recognizes Σ (the subset of Σ^* that consists of single-symbol strings).
- Given any finite set $L \subseteq \Sigma*$, specified by listing its members, define a FA that recognizes L.
- Define a FA that accepts all strings over $\{a, b\}$ that contain an even number of b's.
- Define a FA that accepts all strings over $\{a, b\}$ that do not contain three consecutive b's.

Regular Languages

Def: *(Sipser, Def. 1.6)* A language is *regular* if some finite automaton recognizes it.
i.e.

$$\exists M.\ M \text{ is a FA } \wedge\ M \text{ recognizes } R.$$

To prove R is regular, we may use *(existsI)*. This requires:

1. *Define* a specific $M = (Q, \Sigma, \delta, q_0, F)$.
2. *Show* M is a FA (verify M satisfies the definition).
3. *Show* M recognizes R:
   (a) *Show* $\forall w \in \Sigma^*.\ w \in R \rightarrow M$ accepts w.
   (b) *Show* $\forall w \in \Sigma^*.\ M$ accepts $w \rightarrow w \in R$.

3(a):

Fix w

$w \in \Sigma^*$

$w \in R$

x. *Define* $r_0, r_1, \ldots, r_n$ (recursively) by

$$r_0 = q_0$$
$$r_{i+1} = \delta(r_i, w_{i+1}), i = 0, 1, \ldots, n-1$$

⋮

y. *Show* $r_n \in F$ (typically by induction).

M accepts w $\quad\exists$ **I:x–y**

$w \in R \to M$ accepts w $\quad\to$ **I:**

$w \in \Sigma^* \to w \in R \to M$ accepts w $\quad\to$ **I:**

$\forall w \in \Sigma^*.\ w \in R \to M$ accepts w $\quad\forall$ **I:**

3(b):

Fix w

$w \in \Sigma^*$

x. M accepts w

y. *Obtain* $r_0, r_1, \ldots, r_n$, where

$$r_0 = q_0$$
$$r_{i+1} = \delta(r_i, w_{i+1}), i = 0, 1, \ldots, n-1$$
$$r_n \in F$$

$\vdots$

z. $w \in R$ (typically by induction)

$w \in R$ $\exists$**E:x, y–z**

M accepts $w \rightarrow w \in R$ $\rightarrow$**I:**

$w \in \Sigma^* \rightarrow w \in R \rightarrow M$ accepts w $\rightarrow$**I:**

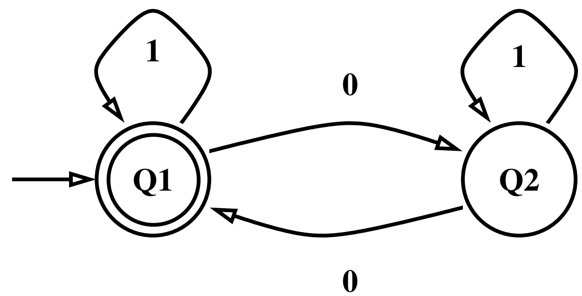
$\forall w \in \Sigma^*.\ M$ accepts $w \rightarrow w \in R$ $\forall$**I:**

Example

Consider the example FA

$$M = (\{Q_1, Q_2\}, \{0, 1\}, \delta, Q_1, \{Q_1\}),$$

where δ is given by the transition diagram:



Let $L = \{w \in \{0, 1\}^*.\ w$ has an even number of 0's$\}$.

Show: The language accepted by M is L.

Key idea: For all $n \geq 0$, the following statement $P(n)$ holds:

P(n): For all $w = w_1, w_2, \ldots w_n$, and all sequences $r_0, r_1, \ldots r_n$ such that

- $r_0 = Q_1$
- $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, 1, \ldots, n-1$,

we have $r_n \in \{Q_1\}$ if and only if w has an even number of 0's.

This is an assertion of the form $\forall n.\ P(n)$, which can be proved by induction . . .

Basis: If $n = 0$, then $r_n = r_0 \in \{Q_1\}$ and the sequences are empty, so w has no 0's, which is an even number.

Induction Step: Suppose (induction hypothesis) that $P(n)$ holds for some fixed, arbitrary n. We must show that $P(n+1)$ holds. Suppose we are given fixed, arbitrary $w = w_1, w_2, \ldots w_n, w_{n+1}$ and $r_0, r_1, \ldots r_n, r_{n+1}$ such that the conditions in $P(n)$ hold. Then $r_n \in \{Q_1\}$ if and only if $w_1, w_2, \ldots w_n$ has an even number of 0's, by $P(n)$. To show $P(n+1)$, we must show that $r_{n+1} \in \{Q_1\}$ if and only if w has an even number of 0's. We consider four cases, depending on whether $r_n \in \{Q_1\}$ and whether w_{n+1} is 0.

1. $r_n \in \{Q_1\}$ and w_{n+1} is 0. Then $r_{n+1} = Q_2 \notin \{Q_1\}$ and the number of 0's in w is odd.
2. $r_n \in \{Q_1\}$ and w_{n+1} is 1. Then $r_{n+1} = Q_1 \in \{Q_1\}$ and the number of 0's in w is even.

   $\ldots$

In each case, we have $r_{n+1} \in \{Q_1\}$ if and only if w has an even number of 0's, as required.

The Regular Operations

Def: *(Sipser, Def. 1.6)* The *regular operations* on languages are:

- *Union:* $A \cup B = \{x.\ x \in A \vee x \in B\}$
- *Concatenation:* $A \circ B = \{xy.\ x \in A \wedge y \in B\}$
- *(Kleene) Star:* $A^* = \{x_1 x_2 \ldots x_k.\ k \geq 0 \wedge \text{each } x_i \in A\}$.

Where do these come from? Kleene discovered that these could be used to characterize regular languages (as we shall see).

Alternative Ways to Express Concatenation and Star

Concatenation: $A \circ B = \{w.\ \exists xy.\ w = xy \land x \in A \land y \in B\}$

Star: $A^* = \bigcup\{A^i.\ i \geq 0\}$, where

- $A^0 = \{\epsilon\}$
- $A^{i+1} = A^i \circ A$, for $i \geq 0$.

Star: A^* is the unique smallest set containing ϵ and such that $A^* \circ A \subseteq A^*$; *i.e.* A^* is *closed under concatenation with A*.

So A^* is the result of "iterated concatenation with A".

Proof *of ($A^* \circ A \subseteq A^*$):*

Let $w \in A^* \circ A$ be an arbitrarily fixed element of $A^* \circ A$. Then we may choose x and y so that $x \in A^*$ and $y \in A$. Since $x \in A^*$ we may also choose $n \geq 0$ and $x_1 x_2 \ldots x_n$ such that $x_i \in A$ for $1 \leq i \leq n$ and $x = x_1 x_2 \ldots x_n$. But then if we take $x_{n+1} = y$ we have $x_1 x_2 \ldots x_n x_{n+1} \in A^*$ by definition of A^*. Thus, we have shown an arbitrary $w \in A^* \circ A$ is also in $A^\star$, hence $\forall w.\ w \in A^* \circ A \rightarrow w \in A^*$; *i.e.* $A^* \circ A \subseteq A^*$.

Proof *that A^* is smallest:*

We show that for all B, if $B \circ A \subseteq B$, then $A^* \subseteq B$. Fix B arbitrarily, and assume $B \circ A \subseteq B$. We claim that for all $n \geq 0$, if $x_i \in A$ for $1 \leq i \leq n$, then $x_1 x_2 \ldots x_n \in B$. The proof of the claim is by induction (exercise). Then, given arbitrary $w \in A^*$, we may obtain $n \geq 0$ and $x_1, x_2, \ldots x_n$ such that $x_i \in A$ for $1 \leq i \leq n$ and $w = x_1 x_2 \ldots x_n$. Then $w \in B$ by the claim. Since w was arbitrary, we conclude $\forall w.\ w \in A^* \rightarrow w \in B$; *i.e.* $A^* \subseteq B$.

Closure under Union

Theorem *(Sipser, 1.25):* The class of regular languages is closed under the union operation.

$$\forall A_1\ A_2.\ A_1 \text{ regular } \wedge\ A_2 \text{ regular } \rightarrow A_1 \cup A_2 \text{ regular}$$

Notes:

- Unlike the closure property of A^*, this is not a statement about just one regular language, it is a statement about the *class of all* regular languages.
- Informally, just regard *"class"* as another word for *"set"*. For a fixed alphabet Σ, there is no technical difference.
- The theorem holds even if the alphabet is not fixed.

Proof Outline:

u. *Fix* A_1 and A_2

x. A_1 reg. $\wedge$ A_2 reg.

y. *Obtain* M_1 and M_2 where

M_1 recognizes A_1

M_2 recognizes A_2

Define M using M_1 and M_2

$\vdots$ (Core of the proof)

M recognizes $A_1 \cup A_2$

z. $A_1 \cup A_2$ reg.

w. $A_1 \cup A_2$ reg. $\exists$ **E:x, y–z**

v. A_1 reg. $\wedge$ A_2 reg. $\rightarrow$ $A_1 \cup A_2$ reg. $\rightarrow$ **I:x–w**

$\forall A_1\ A_2.\ A_1$ reg. $\wedge$ A_2 reg. $\rightarrow$ $A_1 \cup A_2$ reg. $\wedge$ **I:u–v**

Proof: The Central Idea

We need to take M_1 that recognizes A_1 and M_2 that recognizes A_2 and define M that recognizes $A_1 \cup A_2$.

Key Idea: Run M_1 and M_2 in parallel and accept if and only if either M_1 or M_2 accepts. This can be done by a *product automaton*.

Suppose:

$$M_1 = (\Sigma, Q_1, \delta_1, q_1, F_1) \qquad M_2 = (\Sigma, Q_2, \delta_2, q_2, F_2)$$

Define:

$$M = (\Sigma, Q_1 \times Q_2, \delta, (q_1, q_2), F)$$

where

- $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
- $F = \{(r_1, r_2).\ r_1 \in F_1 \vee r_2 \in F_2\}$.

The transition function δ simulates M_1 and M_2 in parallel.

The accept states are exactly those in which M_1 accepts or M_2 accepts.

Proof that M Recognizes $A_1 \cup A_2$

It is not enough just to define M, we have to prove it correct:

$$\forall w.\ w \in \Sigma^* \to (M \text{ accepts } w \leftrightarrow w \in A_1 \cup A_2).$$

Fix w and *assume* $w \in \Sigma^*$ . . .

(Only if) Suppose M accepts $w = w_1 w_2 \dots w_n$. Then we may *obtain* (using $(\exists\,\mathbf{E})$) a sequence

$$(r_{0,1}, r_{0,2}), (r_{1,1}, r_{1,2}), \dots (r_{n,1}, r_{n,2})$$

such that:

- $(r_{0,1}, r_{0,2}) = (q_1, q_2)$
- $\delta((r_{i,1}, r_{i,2}), w_{i+1}) = (r_{i+1,1}, r_{i+1,2})$, for $i = 0, \dots, n-1$
- $(r_{n,1}, r_{n_2}) \in F$.

Now, $(r_{n,1}, r_{n,2}) \in F$ if and only if either $r_{n,1} \in F_1$ or $r_{n,2} \in F_2$. We prove $w \in A_1 \cup A_2$ in each case $(\vee\,\mathbf{E})$:

- *Assume* $r_{n,1} \in F_1$. Then the sequence $r_{0,1}, r_{1,1}, \dots, r_{n,1}$ shows M_1 accepts w, so $w \in A_1$, hence $w \in A_1 \cup A_2$.
- *Assume* $r_{n,2} \in F_2$. Then the sequence $r_{0,2}, r_{1,2}, \dots, r_{n,2}$ shows M_2 accepts w, so $w \in A_2$, hence $w \in A_1 \cup A_2$.

(If) Suppose $w \in A_1 \cup A_2$. Then either M_1 accepts w or M_2 accepts w. We prove M accepts w in either case (to apply $(\vee\mathbf{E})$).

Assume M_1 accepts w. Then we may *obtain* $r_{0,1}, r_{1,1}, \ldots, r_{n,1}$ such that

- $r_{0,1} = q_1$.
- $\delta_1(r_{i,1}, w_{i+1}) = r_{i+1,1}$, for $i = 0, 1, \ldots, n-1$.
- $r_{n,1} \in F_1$.

Define $r_{0,2}, r_{1,2}, \ldots, r_{n,2}$ (recursively) as follows:

- $r_{0,2} = q_2$.
- $r_{i+1,2} = \delta_2(r_{i,2}, w_{i+1})$, for $i = 0, 1, \ldots, n-1$.

We then *show* that

- $(r_{0,1}, r_{0,2}) = (q_1, q_2)$
- $\delta((r_{i,1}, r_{i,2}), w_{i+1}) = (r_{i+1,1}, r_{i+1,2})$, for $i = 0, \ldots, n-1$
  (this part uses induction)
- $(r_{n,1}, r_{n_2}) \in F$.

This proves that M accepts w.

Assume M_2 accepts w. A symmetric proof (omitted) shows M accepts w.

Since we have shown that M accepts w under either the assumption that M_1 accepts w or that M_2 accepts w, we may conclude (using $(\vee\mathbf{E})$) that M accepts w holds unconditionally.

Closure under Concatenation

Theorem *(Sipser, 1.26):* The class of regular languages is closed under the concatenation operation.

$$\forall A_1\ A_2.\ A_1 \text{ regular} \ \wedge\ A_2 \text{ regular} \ \rightarrow A_1 \circ A_2 \text{ regular}$$

Can we prove it in a similar way?

Recall

$$A_1 \circ A_2 = \{x_1 x_2.\ x_1 \in A_1 \wedge x_2 \in A_2\}$$

- We would need to construct, given M_1 recognizing A_1 and M_2 recognizing A_2, an automaton M that recognizes $A_1 \circ A_2$.
- Maybe we could run M_1 until the end of x_1, then run M_2 on x_2?
- How can we decide where x_1 leaves off and x_2 starts?

So we need some new techniques before we can prove this.

New Approach

There are two ways to think about how to solve this problem:

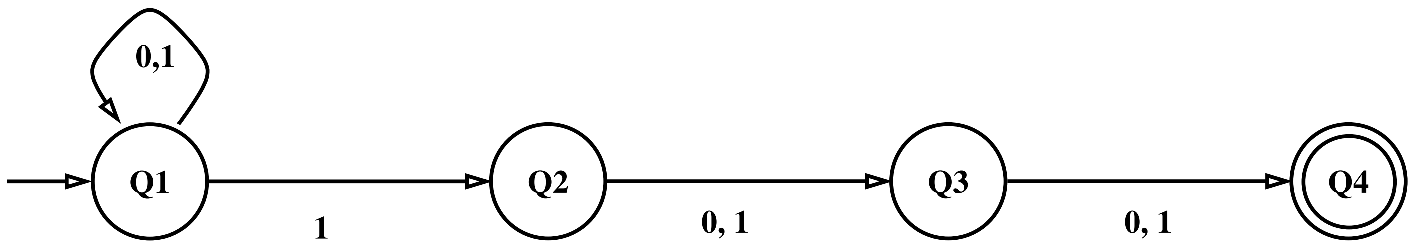
- Try all possible x_1 / x_2 division points in parallel.
  Difficult to understand how to do it.

- Suppose we are able to make a "lucky guess" about where x_1 ends and x_2 starts, and then check whether the guess "works" (*i.e.* leads to acceptance of $x_1 x_2$).
  This is the essence of *nondeterminism*.

Nondeterministic Finite Automata

We allow multiple transitions (even none) for each (q, x). An NFA accepts if *some* (lucky!) choice of moves leads to acceptance.

Example *(Sipser, 1.30):*



Accepts strings over $\{0, 1\}$ with a 1 in the third position from the end. It "guesses" when near the end and then verifies the guess.

Formal Definition of NFA

Def: *(Sipser, Def. 1.5)* A *nondeterministic finite automaton* is a 5-tuple:

$$(Q, \Sigma, \delta, q_0, F),$$

where

- Q is a finite set called the *states*.
- Σ is a finite set called the *alphabet*.
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$ is the *transition function*.
- $q_0 \in Q$ is the *start state*.
- $F \subseteq Q$ is the set of *accept states*.

The function δ gives a *set* of possibilities for the next state, and a transition need not consume an input symbol.

Formal Definition of the Language Recognized by a NFA

Def: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a NFA and suppose $w \in \Sigma^*$. Then M *accepts* w if we can write $w = y_1\, y_2\, \ldots\, y_n$, with each y_i in $\Sigma \cup \{\epsilon\}$, and there exists a sequence of states $r_0, r_1, \ldots, r_n$ in Q such that the following three conditions are satisfied:

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

The set $\{w \in \Sigma^*.\ M \text{ accepts } w\}$ is called the *language recognized by M*.

Equivalence of Automata

Def. Two automata are *equivalent* if they recognize the same language.

Proposition*:* Every DFA has an equivalent NFA.

$\forall M.\ M$ a DFA $\rightarrow$ ($\exists N.\ N$ a NFA $\wedge$ M and N are equivalent).

Proof Idea: Given M, replace δ by δ', where we define $\delta'(r, \epsilon) = \{\}$ (no ϵ-transitions) and $\delta'(r, x) = \{\delta(r, x)\}$ (single-valued).

It is trivial (uninterestingly easy) to verify the equivalence.

Equivalence between NFA and FA

The following is less obvious and much more interesting:

Theorem *(Sipser, 1.39):* Every NFA has an equivalent DFA.

$\forall N.\ N$ a NFA $\rightarrow$ ($\exists M.\ M$ a FA $\wedge$ M and N are equivalent).

Idea of Proof: Given NFA N, we construct FA M so that it keeps track of *all possible states* that N could be in after reading some input.

(Think about using coins to mark states of N and how the markings would have to be updated as each symbol is read.)

“Powerset Automaton” Construction

Given NFA

$$N = (Q, \Sigma, \delta, q_0, F)$$

define DFA

$$M = (Q', \Sigma, \delta', q'_0, F')$$

by . . .

1. $Q' = \mathcal{P}(Q)$
   (all possible markings of states of N)

2. $F' = \{R \in Q'.\ R \cap F \neq \phi\}$
   (all markings in which a state in F is marked)

3. $q'_0 = E(\{q_0\})$
   (mark all states reachable by ϵ-transitions from q_0)

4. $\delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a))$
(update the marking R by marking each state that is reachable from a state $r \in R$ by taking a single a-transition, followed by some sequence of ϵ-transitions)

Def. $E(R) = \{q.\ q$ *can be reached from* R *by following 0 or more* ϵ*-transitions*$\}$.

(the "ϵ-closure" of R)

Correctness of the Construction

The following is the key fact in the correctness proof. From this, it is easy to check that N and M recognize the same language.

Claim: For all $n \geq 0$ and all strings w with $|w| = n$, the state R reached by DFA M on input w is exactly the set of all states r reachable by NFA N on input w.

Proof: By induction.

Basis: If $n = 0$ then $w = \epsilon$ and M reaches $q_0' = E(\{q_0\})$ on input w. But $E(\{q_0\})$ is exactly the set of all states r reachable by NFA N on input ϵ.

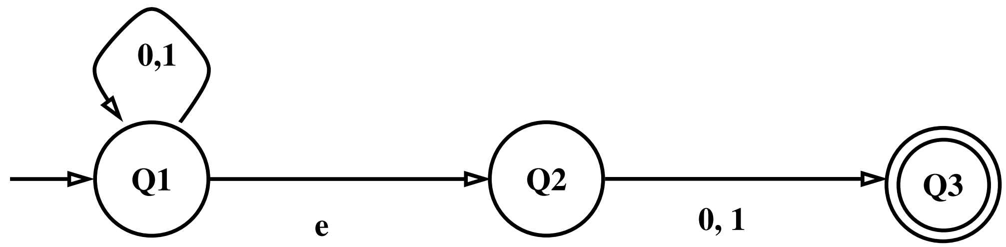
Induction Step: Suppose, for some $n \geq 0$ we have shown that for all strings w with $|w| = n$, the state R reached by DFA M upon reading w is exactly the set of all states r reachable by NFA N on input w.

Let v be an arbitrary string with $|v| = n + 1$. Then $v = wa$ for some w with $|w| = n$ and some $a \in \Sigma$. Let R be the state reached by M on input w; then by the induction hypothesis, R is exactly the set of all states r reachable by NFA N on input w.

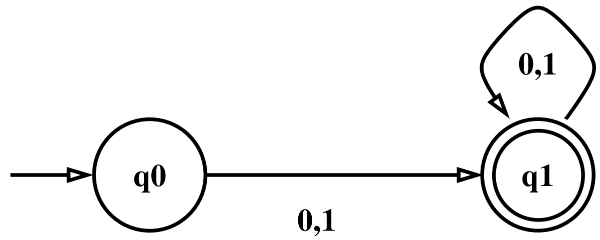
Now, a state s is reachable by N on input wa if and only if there is a state $r \in R$ such that s is reachable from r by a single a transition and a sequence of ϵ-transitions; that is, if and only if $s \in \delta'(R, a)$.

Note: This argument still doesn't deal completely convincingly with the issue of ϵ-transitions, but to do better requires heavier formalism.

Example



- $q'_0 = E(\{Q1\}) = \{Q1, Q2\}$.
- $\delta'(\{Q1, Q2\}, 0) = E(\{Q1, Q3\}) = \{Q1, Q2, Q3\}$
- $\delta'(\{Q1, Q2\}, 1) = E(\{Q1, Q3\}) = \{Q1, Q2, Q3\}$.
- $\delta'(\{Q1, Q2, Q3\}, 0) = E(\{Q1, Q3\}) = \{Q1, Q2, Q3\}$.
- $\delta'(\{Q1, Q2, Q3\}, 1) = E(\{Q1, Q3\}) = \{Q1, Q2, Q3\}$.



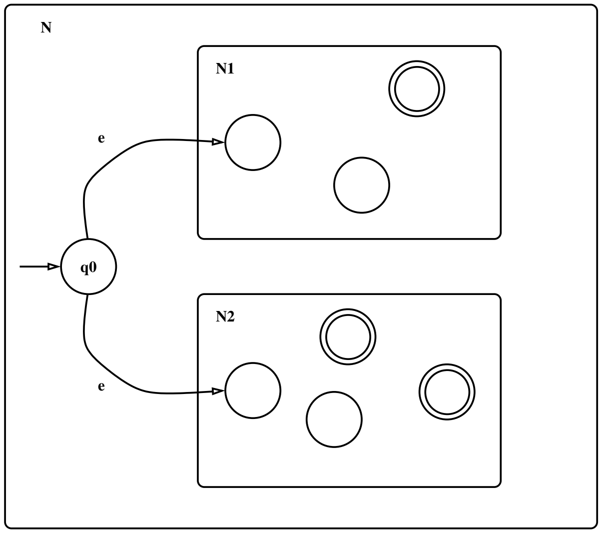
$$q0 = \{Q1, Q2\} \qquad q1 = \{Q1, Q2, Q3\}$$

The full definition of M also includes all six other subsets of $\{Q1, Q2, Q3\}$, but these will not be reachable from $\{Q1, Q2\}$, so are irrelevant to the language recognized by M.

Closure under Union

Theorem *(Sipser, 1.45):* The class of regular languages is closed under the union operation.

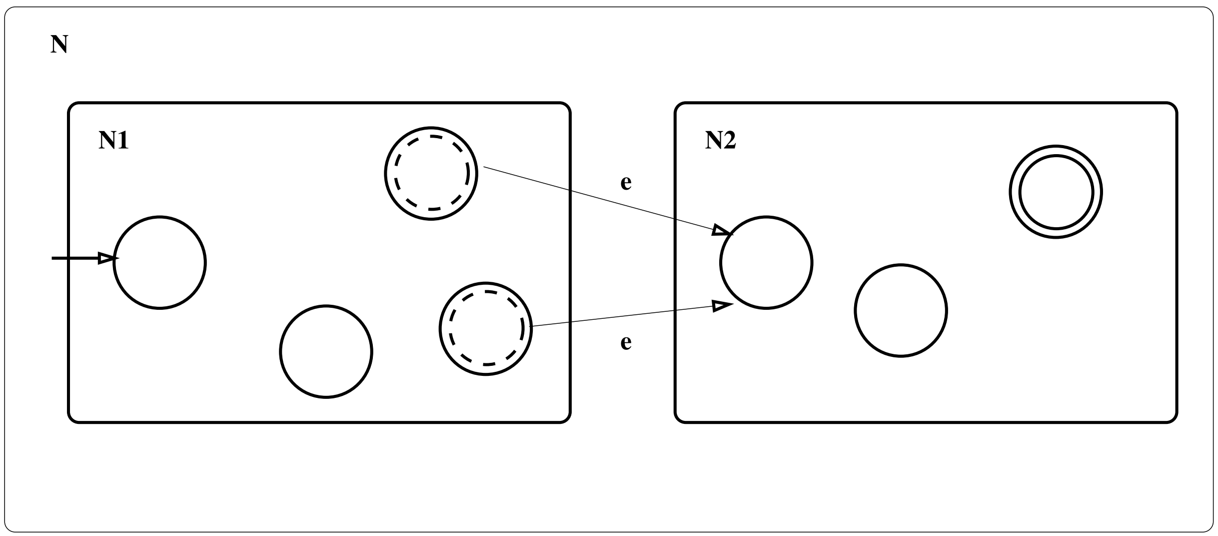
Proof Idea: Given A_1 and A_2, both regular, obtain NFA's N_1 and N_2, where N_1 recognizes A_1 and N_2 recognizes A_2. Then N constructed as follows recognizes $A_1 \cup A_2$:



Closure under Concatenation

Theorem *(Sipser, 1.47):* The class of regular languages is closed under the concatenation operation.

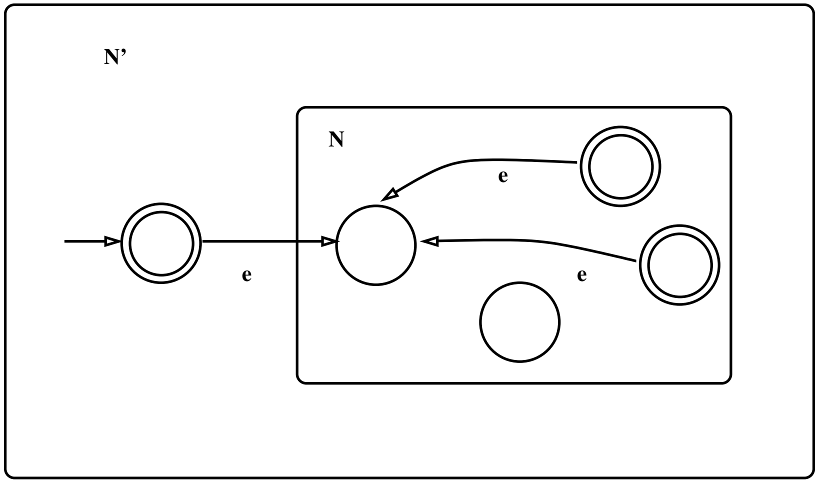
Proof Idea: Given A_1 and A_2, both regular, obtain NFA's N_1 and N_2, where N_1 recognizes A_1 and N_2 recognizes A_2. Then N constructed as follows recognizes $A_1 \circ A_2$:



Closure under Star

Theorem *(Sipser, 1.49):* The class of regular languages is closed under the star operation.

Proof Idea: Given A regular, obtain NFA N where N recognizes A. Then N' constructed as follows recognizes A^*:



(Handling the start state is slightly tricky.)