university of groningen

Languages and Machines

L3: Finite State Machines (Part 1)

Jorge A. Pérez

Bernoulli Institute for Mathematics, Computer Science, and AI
University of Groningen, Groningen, the Netherlands

April 28, 2023

Languages and Their Machines

Regular	$\leftrightarrow$	Finite State Machines (FSMs)
Context-free	$\leftrightarrow$	Pushdown Machines
Context-sensitive	$\leftrightarrow$	Linearly-bounded Machines
Decidable	$\leftrightarrow$	Always-terminating Turing Machines
Semi-decidable	$\leftrightarrow$	Turing Machines

Notation

- Given a relation $\mathcal{R}$, we write $\mathcal{R}^*$ to denote its reflexive, transitive closure.
- Given a set Q, we write $\mathcal{P}(Q)$ to denote the powerset of Q, i.e., the set of all subsets of Q.
  (The reader uses $\mathbb{P}(Q)$ instead of $\mathcal{P}(Q)$.)

Recognizing Strings

Consider the regular expression

$$(\texttt{ca})^{*}\texttt{ab}^{*}$$

Strings we want to recognize:

a, caa, $abbb$, $cacaa$, ...

Recognizing Strings

Consider the regular expression

$$\texttt{(ca)}^*\texttt{ab}^*$$

Strings we want to recognize:

a, caa, $abbb$, $cacaa$, ...

Tasks for a **program** that recognizes strings denoted by $\texttt{(ca)}^*\texttt{ab}^*$:

1. Scan zero OR multiple occurrences of: c followed by a
2. Scan exactly one occurrence of: a
3. Scan zero OR multiple occurrences of: b

Recognizing Strings

Consider the regular expression

$$(\texttt{ca})^*\texttt{ab}^*$$

Strings we want to recognize:
a, caa, $abbb$, $cacaa$, ...

Strings we don't want to recognize:
ca, $cacab$, $aabbb$, $caca$, ...

Tasks for a **program** that recognizes strings denoted by $(\texttt{ca})^*\texttt{ab}^*$:

1. Scan zero OR multiple occurrences of: c followed by a
2. Scan exactly one occurrence of: a
3. Scan zero OR multiple occurrences of: b

The program can get stuck!

Recognizing Strings

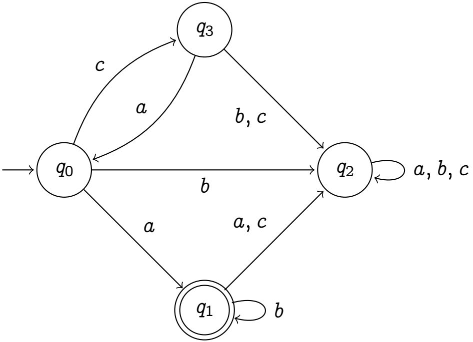
Consider the regular expression

$$\texttt{(ca)}^*\texttt{ab}^*$$

Strings we want to recognize:
a, caa, $abbb$, $cacaa$, ...

Strings we don't want to recognize:
ca, $cacab$, $aabbb$, $caca$, ...

A finite state **machine**:



Three Machines for Regular Languages

Regular Languages

- Built from $\emptyset$, $\{\epsilon\}$, and $\{a_i\}$ (for every $a_i \in \Sigma$) by applications of union, concatenation, and Kleene star operators

The Machines

1. DFSMs: Deterministic finite state machines
2. NFSMs: Nondeterministic finite state machines
3. NϵFSMs: Nondeterministic finite state machines with ϵ-transitions

DFSMs

A **deterministic finite state machine (DFSM)** is a quintuple $M = (Q, \Sigma, \delta, q_0, F)$ where:

- Q is a set of *states*
- Σ is the *input alphabet*
- $\delta : Q \times \Sigma \to Q$ is the *transition function*
- q_0 is the *initial state*
- $F \subseteq Q$ is a set of *accepting* (or *final*) states

Notice:

- When symbol a is read in a state q, the state becomes $\delta(q, a)$.
- NFSMs and NϵFSMs will arise by generalizing/extending δ

DFSMs Process Strings

- A DFSM $M = (Q, \Sigma, \delta, q_0, F)$ *processes* a string $w \in \Sigma^*$ by
  - start in q_0
  - then traverse the graph based on the symbols of w, following δ.
- String w is **accepted** by M if processing w leads to a $q \in F$.
- $L(M)$: the set of the strings that are accepted by M.

DFSMs Process Strings

- A DFSM $M = (Q, \Sigma, \delta, q_0, F)$ *processes* a string $w \in \Sigma^*$ by
  - start in q_0
  - then traverse the graph based on the symbols of w, following δ.
- String w is **accepted** by M if processing w leads to a $q \in F$.
- $L(M)$: the set of the strings that are accepted by M.

More formally, two methods of defining $L(M)$:

- Generalizing $\delta : Q \times \Sigma \to Q$ into $\hat{\delta}$ (recursively defined):

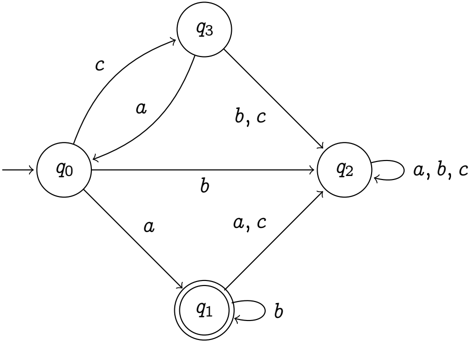
$$\hat{\delta} : Q \times \Sigma^* \to Q$$

  In this case, $L(M) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}$
- A step relation $\vdash_M$ on *configurations*, is defined by

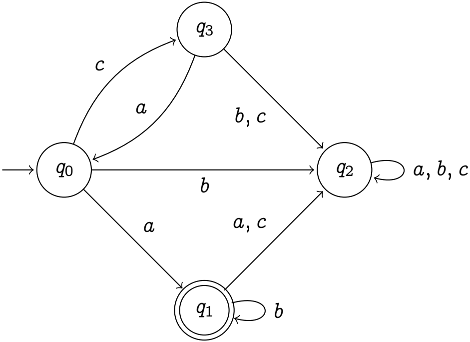
$$(q, aw) \vdash_M (\delta(q, a), w)$$

  In this case, $L(M) = \{w \in \Sigma^* \mid \exists q_i \in F. (q_0, w) \vdash_M^* (q_i, \epsilon)\}$

Example 3.1: State Diagram

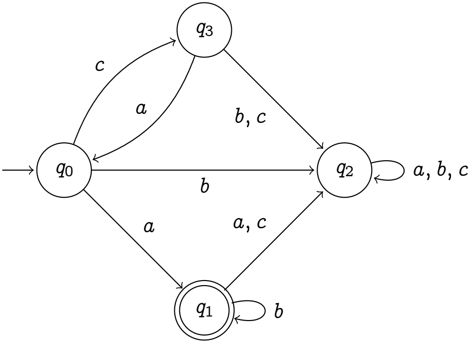


Example 3.1: State Diagram



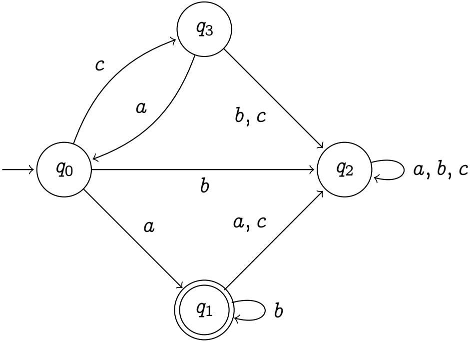
What is Σ? Is it related to Q?

Example 3.1: State Diagram



What is Σ? Is it related to Q? What is δ?

Example 3.1: State Diagram

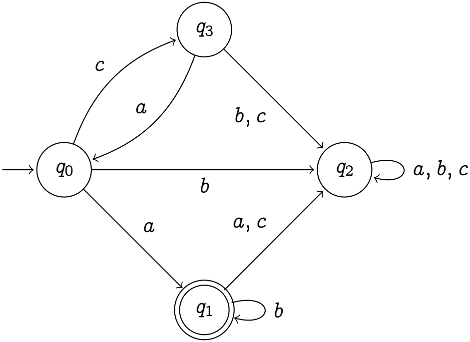


What is Σ? Is it related to Q? What is δ?

The two methods for processing and acceptance:

- $\hat{\delta}(q_0, caabb) = q_1$
- $(q_0, caabb) \vdash_M^* (q_1, \epsilon)$

Example 3.1: State Diagram



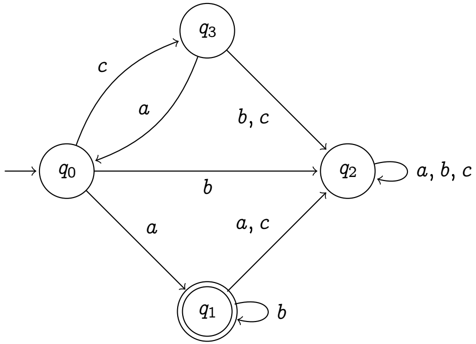
What is Σ? Is it related to Q? What is δ?

The two methods for processing and acceptance:

- $\hat{\delta}(q_0, caabb) = q_1$
- $(q_0, caabb) \vdash_M^* (q_1, \epsilon)$

Because $q_1 \in F$, we have $caabb \in L(M)$

Example 3.1: State Diagram

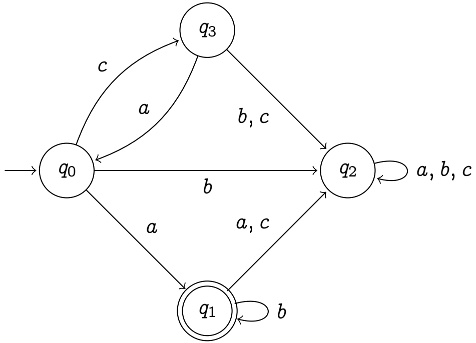


What is Σ? Is it related to Q? What is δ?

The two methods for processing and acceptance:

- $\hat{\delta}(q_0, caaab) = q_2$
- $(q_0, caaab) \vdash_M^* (q_2, \epsilon)$

Example 3.1: State Diagram



What is Σ? Is it related to Q? What is δ?

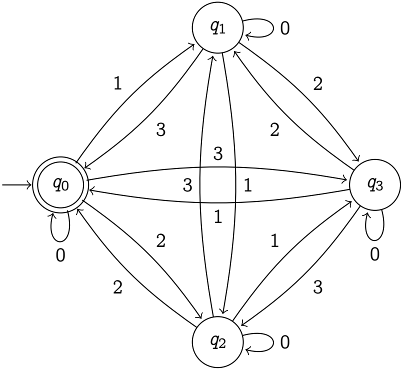
The two methods for processing and acceptance:

- $\hat{\delta}(q_0, caaab) = q_2$
- $(q_0, caaab) \vdash_M^* (q_2, \epsilon)$

Because $q_2 \notin F$, we have $caaab \notin L(M)$

Another Example

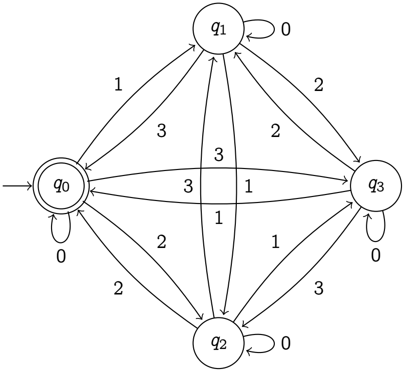
Consider the machine M:



δ	0	1	2	3
$\rightarrow * q_0$				

Another Example

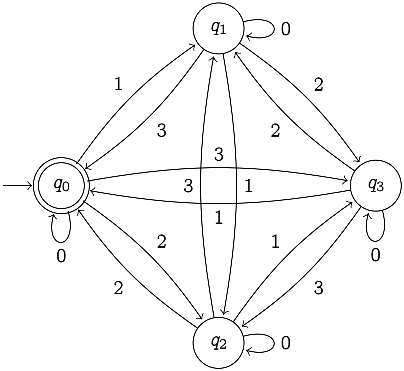
Consider the machine M:



δ	0	1	2	3
$\rightarrow * \; q_0$	q_0	q_1	q_2	q_3
q_1				

Another Example

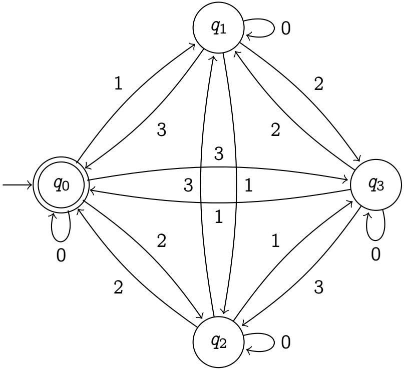
Consider the machine M:



δ	0	1	2	3
$\rightarrow * \ q_0$	q_0	q_1	q_2	q_3
q_1	q_1	q_2	q_3	q_0
q_2				

Another Example

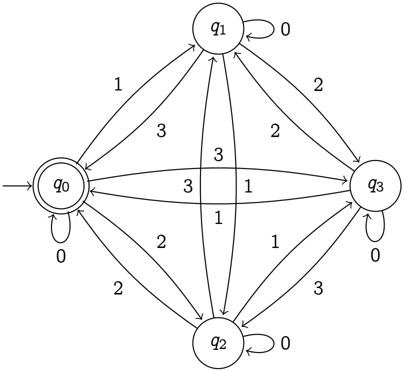
Consider the machine M:



δ	0	1	2	3
$\rightarrow * \; q_0$	q_0	q_1	q_2	q_3
q_1	q_1	q_2	q_3	q_0
q_2	q_2	q_3	q_0	q_1
q_3	q_3	q_0	q_1	q_2

Another Example

Consider the machine M:



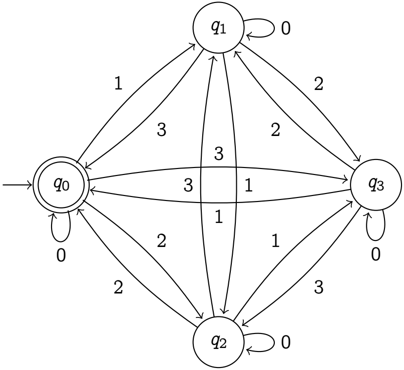
δ	0	1	2	3
$\rightarrow * \; q_0$	q_0	q_1	q_2	q_3
q_1	q_1	q_2	q_3	q_0
q_2	q_2	q_3	q_0	q_1
q_3	q_3	q_0	q_1	q_2

Q: What does this machine determine? Some clues:

- Some accepted strings: 1 2 3 0 2 and 0 1 3 0.
- Some rejected strings: 0 1 1 1 and 1 1 1 2.

Another Example

Consider the machine M:



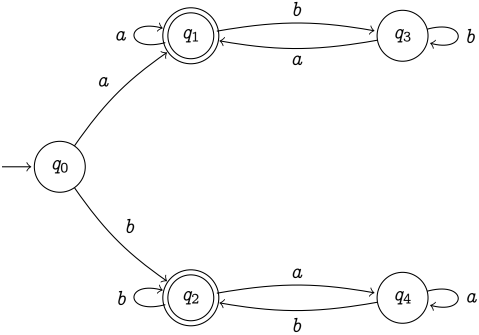
δ	0	1	2	3
$\rightarrow * \; q_0$	q_0	q_1	q_2	q_3
q_1	q_1	q_2	q_3	q_0
q_2	q_2	q_3	q_0	q_1
q_3	q_3	q_0	q_1	q_2

Q: What does this machine determine? Some clues:

- Some accepted strings: $1\,2\,3\,0\,2$ and $0\,1\,3\,0$.
- Some rejected strings: $0\,1\,1\,1$ and $1\,1\,1\,2$.

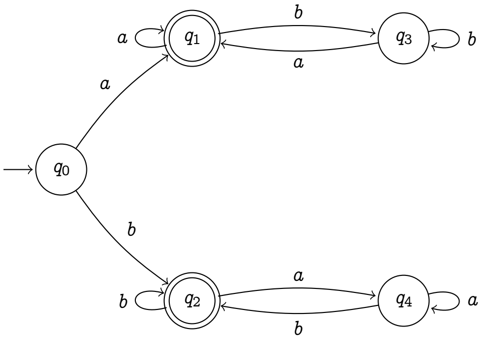
A: M accepts strings whose sum of their elements is divisible by 4.

Yet Another Example



δ	a	b
$\rightarrow q_0$	q_1	q_2
$* q_1$	q_1	q_3
$* q_2$	q_4	q_2
q_3	q_1	q_3
q_4	q_4	q_2

Yet Another Example

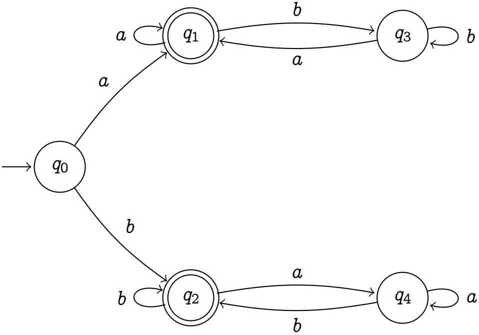


δ	a	b
$\rightarrow q_0$	q_1	q_2
$* \; q_1$	q_1	q_3
$* \; q_2$	q_4	q_2
q_3	q_1	q_3
q_4	q_4	q_2

Q: What does this machine determine? Some clues:

- Some accepted strings: $a\ a\ b\ b\ a$, $a\ a\ a\ a$, and $b\ a\ a\ b$.
- Some rejected strings: $a\ b\ a\ b$ and $b\ a\ a\ a$.

Yet Another Example



δ	a	b
$\rightarrow q_0$	q_1	q_2
$* \; q_1$	q_1	q_3
$* \; q_2$	q_4	q_2
q_3	q_1	q_3
q_4	q_4	q_2

Q: What does this machine determine? Some clues:

- Some accepted strings: $a\ a\ b\ b\ a$, $a\ a\ a\ a$, and $b\ a\ a\ b$.
- Some rejected strings: $a\ b\ a\ b$ and $b\ a\ a\ a$.

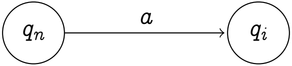
A: It accepts strings that start and end with the same letter.

From Wiktionary

- **Determinism**
  The property of having behavior determined only by initial state and input.
- **Nondeterminism**
  The property of being nondeterministic, involving arbitrary choices; necessitating the choice between various indistinguishable possibilities.
- **Angelic Nondeterminism**
  A notional ability always to choose the most favorable option, in constant time.

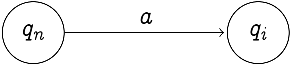
From Deterministic to Nondeterministic

$$\delta(q_n, a) = \quad q_i \ :$$



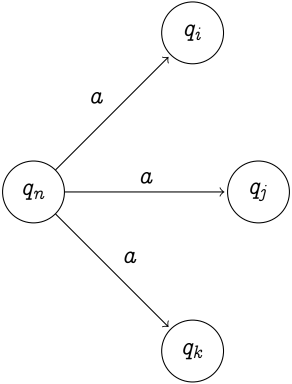
From Deterministic to Nondeterministic

$\delta(q_n, a) = \{q_i\}$:



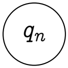
From Deterministic to Nondeterministic

$\delta(q_n, a) = \{q_i, q_j, q_k\}$:



From Deterministic to Nondeterministic

$\delta(q_n, a) = \emptyset$:



NFSMs

A **Nondeterministic finite state machine (NFSM)** is a quintuple $M = (Q, \Sigma, \delta, q_0, F)$ where:

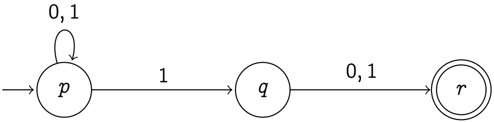
- Q is a set of *states*
- Σ is the *input alphabet*
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is the *transition function*
- q_0 is the *initial state*
- $F \subseteq Q$ is a set of *accepting* (or *final*) states

Notice:

- When symbol a is read in q, the next state is in the **set** $\delta(q, a)$.
- We can consider a set of starting states (rather than just q_0)
- We define $(q, w) \vdash_M (q', w')$ as

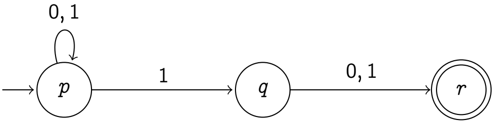
$$(\exists a \in \Sigma : w = aw' \wedge q' \in \delta(q, a))$$

Example 1



- Is this an NFSM? Why?
- Q: What language is recognized?

Example 1



- Is this an NFSM? Why?
- Q: What language is recognized?
  A: $L = \{x \in \{0,1\}^* \mid$ the second symbol from the right is $1\}$.

Example 2

Consider the set:

$$L' = \{x \in \{a\}^* \mid |x| \text{ is divisible by 3 or 5}\}$$

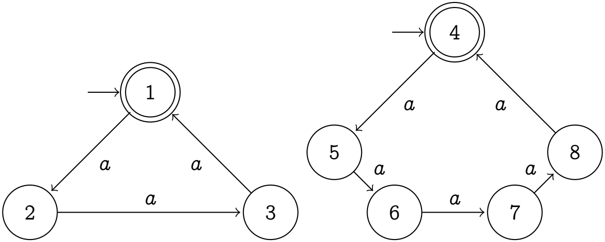
What would be an NFSM for recognizing L'?

Example 2

Consider the set:

$$L' = \{x \in \{a\}^* \mid |x| \text{ is divisible by 3 or 5}\}$$

What would be an NFSM for recognizing L'?



- The only nondeterminism is in the choice of starting state
- Angelic nondeterminism: the NFSM always guesses right

NϵFSMs: NFSMs with ϵ-transitions

An NϵFSMs is a quintuple $M = (Q, \Sigma, \delta, q_0, F)$ where:

- Q is a set of *states*
- Σ is the *input alphabet*
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$ is the *transition function*
- q_0 is the *initial state*
- $F \subseteq Q$ is a set of *accepting* (or *final*) states

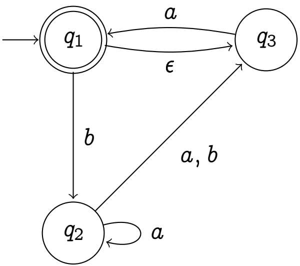
Notice:

- $\delta(q, \epsilon)$: set of the states reachable from q without reading input.
- We define $(q, u) \vdash_M (q', v)$ as

$$\underbrace{(\exists a \in \Sigma : u = av \wedge q' \in \delta(q, a))}_{\text{move by reading } a} \vee \underbrace{(u = v \wedge q' \in \delta(q, \varepsilon))}_{\text{move without reading}}$$

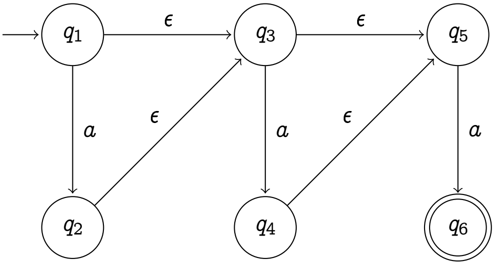
($L(M)$ is defined as before.)

Example 3

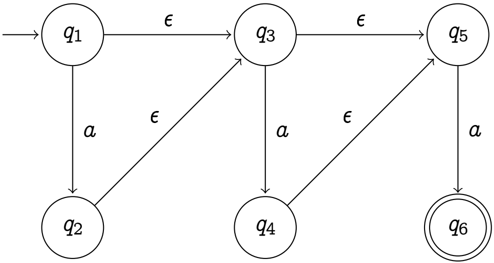


- Some strings that are accepted: a, baa, $baba$.
- Some strings that are not accepted: b, $babba$.

Example 4

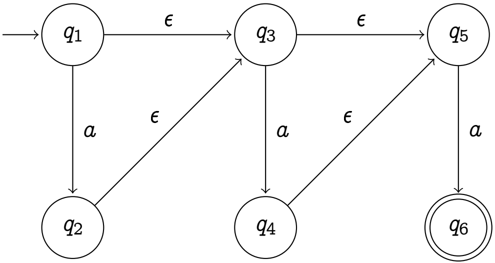


Example 4



Q: What can the machine do in state q_1 with next symbol a?

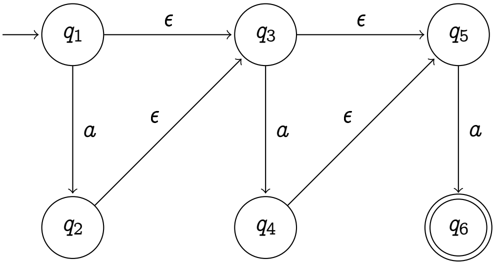
Example 4



Q: What can the machine do in state q_1 with next symbol a?
A: It can nondeterministically do one of three things:

- Read a and move to q_2
- Slide to q_3 without reading input, then read the a and move to q_4
- Slide to q_3 without reading input, then slide to q_5 without reading input, then read the a and move to q_6

Example 4



Q: What can the machine do in state q_1 with next symbol a?
A: It can nondeterministically do one of three things:

- Read a and move to q_2
- Slide to q_3 without reading input, then read the a and move to q_4
- Slide to q_3 without reading input, then slide to q_5 without reading input, then read the a and move to q_6

What set of strings is accepted by this NϵFSM?

Example 2, Revisited

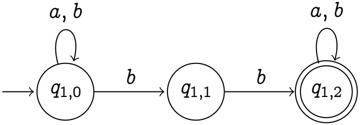
Consider the set:

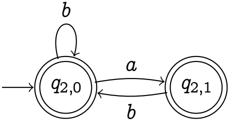
$$L' = \{x \in \{a\}^* \mid |x| \text{ is divisible by 3 or 5}\}$$

What would be an NϵFSM for recognizing L'?

Composing Machines with ϵ-transitions

Consider machines M_1 and M_2:

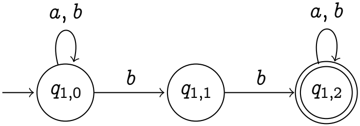


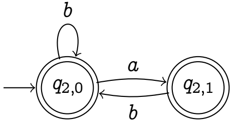


Notice: $L(M_1) = (\mathtt{a}|\mathtt{b})^*\mathtt{bb}(\mathtt{a}|\mathtt{b})^*$ and $L(M_2) = (\mathtt{b}|\mathtt{ab})^*(\mathtt{a}|\epsilon)$.

Composing Machines with ϵ-transitions

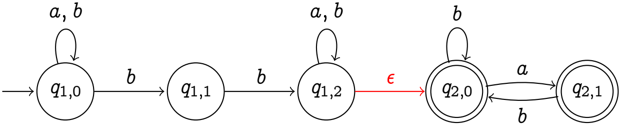
Consider machines M_1 and M_2:





Notice: $L(M_1) = (\mathtt{a}|\mathtt{b})^*\mathtt{bb}(\mathtt{a}|\mathtt{b})^*$ and $L(M_2) = (\mathtt{b}|\mathtt{ab})^*(\mathtt{a}|\epsilon)$.

A composite machine for the concatenation of $L(M_1)$ and $L(M_2)$:



Regular languages & NϵFSMs (Lem. 3.1)

A normal form for NϵFSMs
Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NϵFSM.
Then there is an equivalent NϵFSM M' with:

(i) The new start state q_s has no incoming transitions;

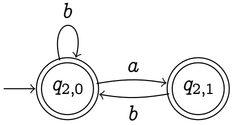
(ii) There is precisely one accepting state q_f, it differs from q_s, and it has no outgoing transitions.

Idea of the proof:

- If needed, add a q_s with an ϵ-transition to q_0
- If needed, add a q_f with an ϵ-transition $q \to q_f$, for all $q \in F$

Example

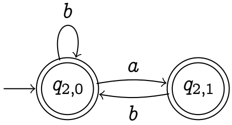
Machine M_2 is *not* in normal form:



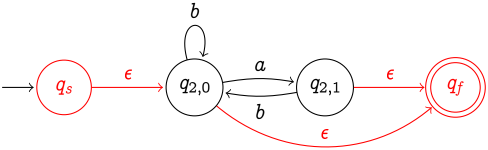
Q: What is its normal form?

Example

Machine M_2 is *not* in normal form:



Q: What is its normal form?
A: We add two states with its transitions:



Regular languages & NϵFSMs (Lem. 3.2)

Concatenation, Union, and Kleene star for NϵFSMs
Let M_1 and M_2 be two NϵFSMs.
Then there are NϵFSMs for each of the three languages:

- $L(M_1)L(M_2)$
- $L(M_1) \cup L(M_2)$
- $L(M_1)^*$

Idea of the proof:

- Assume M_1, M_2 are in normal form (thanks to Lemma 3.1), making sure that their state spaces are disjoint
- Machines for each of the three languages can be built easily

Regular languages & NϵFSMs (Thm. 3.2)

For every regular language L, there is an NϵFSM M with $L(M) = L$.

Idea of the proof:

- Proof method: Induction on the structure of the regular sets
- Three base cases: construct NϵFSMs for $\emptyset$, $\{\epsilon\}$, and $\{a_i\}$ ($a_i \in \Sigma$)
- The induction step uses Lemma 3.2 (previous slide)

Taking Stock

Observe that:

- Every DFSM can be regarded as an equivalent NFSM, and
- Every NFSM can be regarded as an equivalent NεFSM.

Next Lecture

- We will see that every NεFSM gives rise to an equivalent, but much larger, DFSM. (This is the so-called subset construction.)
- Given an N(ε)FSM M, we will determine a regexp for $L(M)$