Finite Automata

Last Updated: August 10th, 2026

1 Languages

Sets of words over some alphabet play a central role in computer science.

Definition 1.1. An **alphabet** $\Sigma = \{s_1, \ldots, s_n\}$ is a set of symbols (also referred to as **letters**). A **word** w over Σ is a sequence of zero or more symbols belonging to Σ. Σ^* denotes the set of all possible words over Σ, including the empty word ε. The **length** of a word w, denoted $|w|$, equals the length of the symbol sequence that comprises w.

$|\text{apple}| = 5$

Example 1.2. If $\Sigma = \{a, b\}$, then aaa, abaabba and bbab are all words over Σ, and $|\text{abaabba}| = 7$. Also $|\varepsilon| = 0$. The expression $\{a, b\}^*$ denotes all words over the alphabet $\{a, b\}$.

The most common operation on words is concatentation. Given words v and w in Σ^*, $v \cdot w$ is the **concatentation** of v with w, and equals the word whose i th letter is v_i if $1 \leq i \leq |v|$, and $w_{i-|v|}$ if $|v| < i \leq |v| + |w|$. Like arithmetic multiplication, we sometimes omit the dot and simply write vw for the concatentation of v with w.

$\text{dog} \cdot \text{house} = \text{doghouse}$

$vw \Leftrightarrow v \cdot w$

words

Definition 1.3. A subset of words over some alphabet Σ is called a **language** over Σ.

Example 1.4. The `Prime` language, is the set of binary words w for which there is a prime number n for which $(n)_2 = w$. The words in `Prime` of length four or less are

$$10, 11, 101, 111, 1011, \text{ and } 1101.$$

2 3 5 7 11 13

Example 1.5. An instance of the `Sort` problem is a sequence of n integers of the form $(a_1\#\ldots\#a_n)$, where each a_i is a nonnegative integer, $i = 1, \ldots, n$. If we think of `Sort` as the language of all words that represent syntactically correct instances of the problem, then provide its alphabet Σ, as well as some words in Σ^* that are in `Sort` and not in `Sort`. Same question, but for the `Prime` decision problem, assuming that valid instances of `Prime` must begin with a leading 1.

$\Sigma = \{0, 1, 2, 3, \ldots, 9, \#, (,)\}$

$0\#((12 \notin Sort$

$(3\#26\#5) \in Sort$

The previous example demonstrates that there is a language associatiated with a computing problem. Conversely, given any language L, there is a decision problem associated with L.

Definition 1.6. Given a language L over some alphabet Σ, an instance of the L-membership problem is a word $w \in \Sigma^*$. The problem is to decide if w is a member of L.

In addition to the above definition of language, the notion of a **formal language** is also central in computer science. Informally, a language is considered formal if there is a set of rules that, when followed, produce words in the language. Such languages can be used to define programming languages, communication protocols, and allow one to search for data in files and databases.

2 Deterministic Finite Automata

In this lecture we begin our study of different automata-based models of computation, starting with the most basic model and working our way up to more complex ones. The determinisitc finite automaton represents the most basic automata-based model of computation. By "automata" based we means that the model is viewed as a machine that computes by transitioning from state to state, starting in an initial state that is usually assocated with the beginning of reading the input data. A deterministic finite automaton consists of a finite number of states which serve as the automaton's only source of memory. Since an automaton is capable of accepting languages having arbitrarily long words, it must read each input word (i.e. problem instance) symbol by symbol with the help of an external tape that stores the input, and a read head that is capable of reading each symbol on the tape until all symbols have been read at which point it either accepts or rejects w.

Finite automata have applications in several areas of computing, including computer architecture, cellular automata, natural language processing and recognition, compiler theory, and text processing to name just a few. Finite automata solve problems that require only a finite amount of memory, regardless of the problem size.

A **deterministic finite automaton (DFA)** consists of a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

Q is a finite set with each member of Q called a **state**

Σ a finite alphabet that is used to represent an input word

δ a transition function that determines the next state of the automaton given the current state and the current input symbol being read. In other words,

$$\delta : Q \times \Sigma \to Q,$$

is a mapping from (current) state-input pairs to (next) states.

$\delta(q, s) = q_{next} \in Q$ (state q, $s \in \Sigma$)

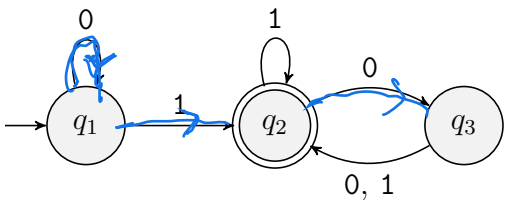
$q_0 \in Q$ the **initial state**.

$F \subseteq Q$ a member of F is called an **accepting state**.

Note that a DFA is capable of reading each input symbol only once, and in the order in which it appears in the input word w.

A graphical way of representing a DFA is to represent the states with graph nodes, with an arrow pointing at the initial state, and accepting states denoted using concentric circles. Furthermore, a directed edge labeled with symbol s starts at node q_i and terminates at node q_j iff $\delta(q_i, s) = q_j$.

Example 2.1. Provide the 5-tuple for the following DFA.



$Q = \{q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$

Initial State: q_1

$F = \{q_2\}$ is the set of accepting

$\delta : Q \times \Sigma \to Q$

current state, current symbol, next state

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

$\delta(q_1, 0) = q_1$

2.1 DFA computation

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA, and $w = w_1 w_2 \cdots w_n$ be a word in Σ^*. Then the **computation of M on input** w is a sequence $q_0, q_1, q_2, \ldots, q_n$ of states recursively defined by

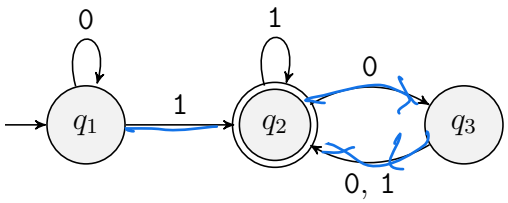
1. q_0 is the initial state of M; and
2. $q_i = \delta(q_{i-1}, w_i)$, for every $1 \leq i \leq n$.

In words, the computation of M on input w is the sequence of states $q_0, q_1, q_2, \ldots, q_n$ that M moves through when successively reading input symbols $w_1, w_2, \ldots, w_n$. The computation of M on input w is said to be **accepting** iff $q_n \in F$. In this case we say M **accepts** w. Otherwise it is called a **rejecting** computation, and M **rejects** w.

Note: a DFA-computation simulator can be found at

`https://www.cs.cmu.edu/~cburch/survey/dfa/index.html`

Example 2.2. For the DFA M shown below, provide the computation of M on input i) $w_1 =$ 01101011 and input ii) $w_2 = 0011000$.



$1000 \notin L$

w_1 0 1 1 0 1 0 1 1

q_1 q_1 q_2 q_2 q_3 q_2 q_3 q_2 $\boxed{q_2}$

$\Downarrow$

accept

2.2 Regular language

Given DFA M, the **language accepted by** M is defined as

$$L(M) = \{w \in \Sigma^* | M \text{ accepts } w\}.$$

If a language $L \subseteq \Sigma^*$ is accepted by some DFA M, then L is called a **regular language**.

Example 2.3. Provide a succinct description of the regular language that is accepted by the DFA shown in Example 1.

At least one 1, and end with a 1 or an even # of 0's

3 Designing DFA's

Given a description of a regular language L, a fundamental programming and problem-solving task is to design a state diagram for a DFA M that accepts L. It is important to become a skilled DFA designer because it is not only the foundation for problem-solving using more advanced automata-based models, but it's also part of the foundation for solving problems with programming languages.

The following are guidelines for designing the necessary DFA.

1. Draw the initial state q_0 including the arrow that should point to it.
2. Write a sentence that describes what must be true (in relation to the description of regular language L) about any word w such that, when w is read by M, the resulting state is q_0. We call this an *annotation for state* q_0.
3. For each state q in your state diagram and for each symbol $s \in \Sigma$ for which there is no arrow leaving q and having label s, create such an arrow and determine to which state the arrow should point.
   (a) To determine this, ask the following question. "If any word w that takes me to q has property X, where X is the annotation of q, then what property would $w \cdot s$ have?
   (b) If your answer is that $w \cdot s$ would have property Y, and Y is the annotation for some state q' in your diagram, then make the arrow point to q'.
   (c) Otherwise, create a new state q', add it to your diagram, give it annotation Y, and make the arrow point to q'.

Example 3.1. Provide the state diagram for a DFA that accepts the language of all words w over the alphabet $\{-1, +1\}$ for which, for all $k = 1, \ldots, |w|$,

$$\left| \sum_{i=1}^{k} w_i \right| \leq 3.$$

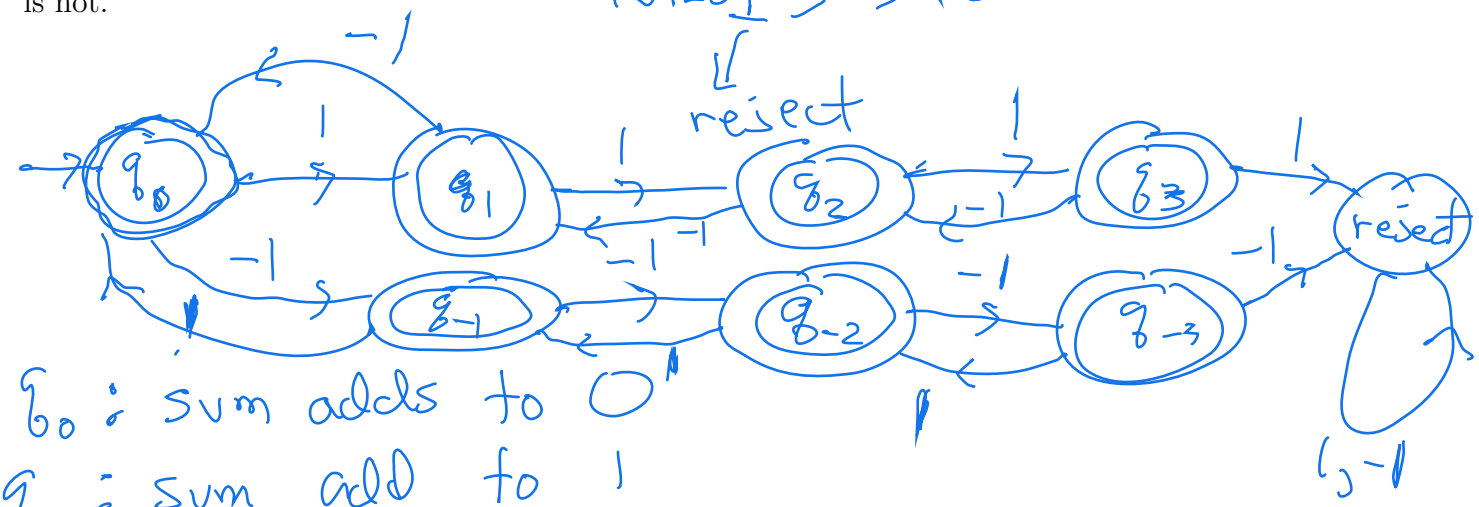
For example,

$1\ 1\ 1\ -1\ 1$
$1\ 2\ 3\ 2\ 3 \Rightarrow$ accept

is in the language, but

$-1\ 1\ 1\ 1\ 1\ 1\ -1$
$-1\ 0\ 1\ 2\ 3\ 4\ 3 \Rightarrow$ reject

is not.



q_0 : sum adds to 0
q_1 : sum add to 1
q_{-1} : " " -1
q_2 : " " 2
q_{-2} : " " -2

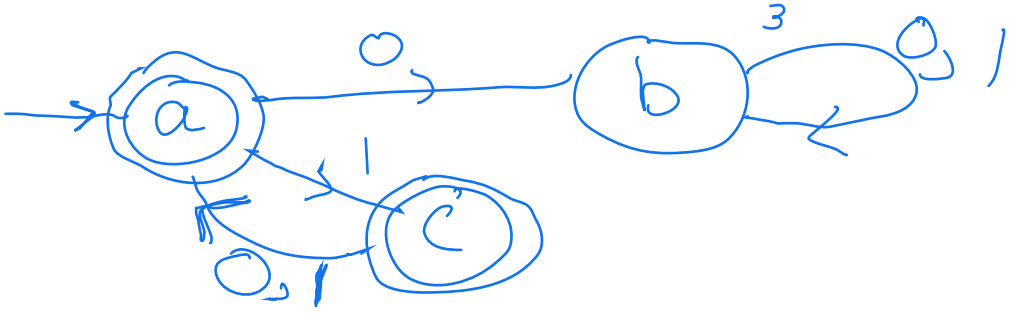
General statement:
q_i : sum adds to i.

reject: word is "out of bounds"

Example 3.2. Let L be the set of binary words w such that every odd bit of w equals 1. Show that L is a regular language.

$w = 1\,1\,1\,0\,1 \in L$ (positions 1 2 3 4 5)

$u = 10001 \notin L$ (position 3)



If i is odd, then $w_i = 1$

$\forall j \geq 1 \left[(i \text{ is odd}) \rightarrow (w_i = 1) \right]$

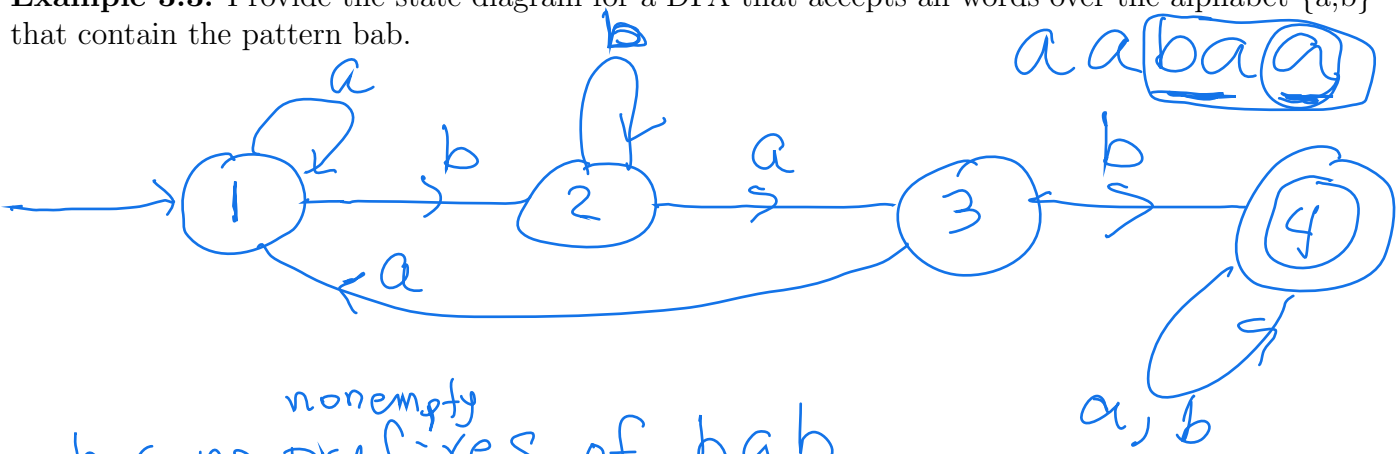
p	Q	P → Q
0	0	1
0	1	1
1	0	0
1	1	1

a: all odd bits equal 1 and an even # of bits have been read

b: reject since there is an odd bit equal to 0

c: all odd bits = 1 and an odd # of bits have been read

Example 3.3. Provide the state diagram for a DFA that accepts all words over the alphabet {a,b} that contain the pattern bab.



aabaa

nonempty

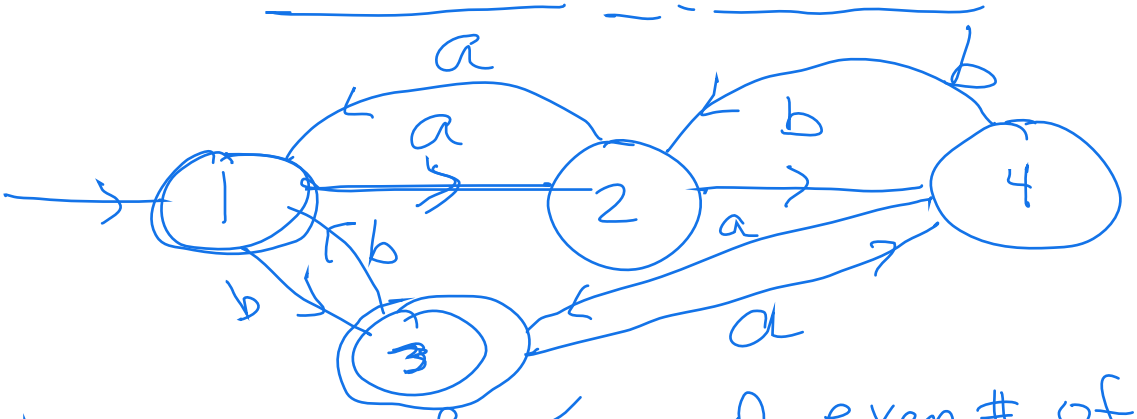
1: has no prefixes of bab

2: has prefix b

3: has prefix ba

4: has bab pattern

Example 3.4. Provide the state diagram for a DFA that accepts all words over the alphabet {a,b} for which there are an even number of a's and and odd number of b's.



1: even # of a's and even # of b's

2: odd # of a's and even # of b's

3: even # of a's and odd # of b's

4: odd # of a's and odd # of b's

w: a b a b b

1 2 4 3 [3] ⇒ accept

Example 3.5. Consider the alphabet

$$\Sigma = \left\{ \begin{matrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{matrix} \right\}$$

where each symbol is a triple of bits.

1. Provide the δ-transition table for a DFA that accepts all words w over Σ for which the top layer of w minus the middle layer of w equals the bottom layer of w, where each layer is viewed as a binary number whose left most bit is the least significant bit, and whose rightmost bit is the most significant bit. For example, consider the two words

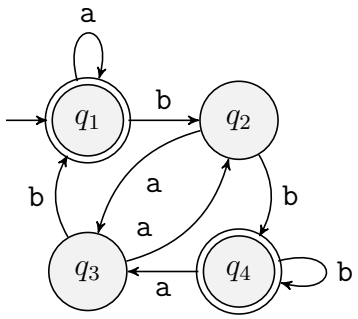
$$w_1 = \begin{matrix} 0001 \\ 1010 \\ 1100 \end{matrix}, \qquad \text{and} \qquad w_2 = \begin{matrix} 11001 \\ 10100 \\ 01010 \end{matrix}.$$

   The DFA should accept w_1 since the top layer represents the number 8, the middle layer 5, the bottom layer 3, and $8 - 5 = 3$. However, it should not accept w_2 since the top layer represents the number 19, the middle layer 5, the bottom layer 25, and $19 - 5 = 14 \neq 10$.

2. Show the computation of the DFA for input w_1 defined above.

DFA Core Exercises

1. Consider the following state diagram for a DFA M.



   Provide the following for M.

   (a) start state
   (b) set of accept states
   (c) the sequence of states that M goes through on input `aabb`
   (d) Does M accept `aabb`? ε?

2. Provide a formal definition for the DFA M from the previous exercise.

3. Consider the formal DFA description

$$M = (\{q_1, q_2, q_3, q_4, q_5\}, \{u, d\}, \delta, q_3, \{q_3\}),$$

   where δ is given by the following table.

$q_i \backslash s$	u	d
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_2	q_4
q_4	q_3	q_5
q_5	q_4	q_5

   Draw the state diagram for this DFA.

4. For the machine M provided in the previous exercise, provide the computation of M on input ududduddu. Is this input accepted or rejected?

5. Provide the state diagram for a DFA that accepts all words from $\{a, b\}^*$ that have an even number of a's.

6. Repeat the previous exercise, but now assume the DFA accepts all words from $\{a, b\}^*$ that have an odd number of a's and end with a b.

7. Provide a DFA that accepts all binary words that contain the substring 0101.

8. Provide a DFA that accepts all binary words that do *not* contain 0101.

9. Provide a DFA that only accepts ε and 10.

10. Provide a DFA that accepts all words over the alphabet {a,b} that have an even number of a's or (inclusive) an odd number of b's.

Solutions to DFA Core Exercises

1. We have the following.

   (a) q_1

   (b) $F = \{q_1, q_4\}$

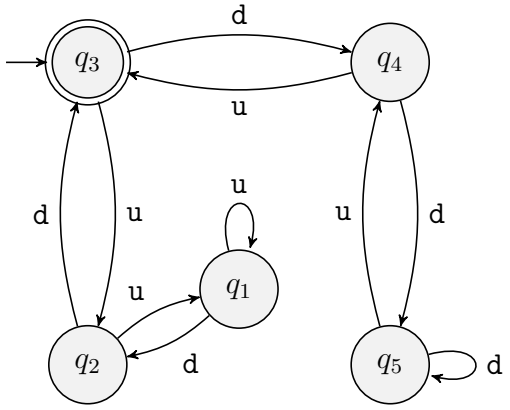
   (c) q_1, q_1, q_1, q_2, q_4

   (d) M accepts `aabb` since the final state of the computation $M(\texttt{aabb})$ is $q_4 \in F$. M accepts ε since the final state of the computation $M(\varepsilon)$ is $q_1 \in F$.

2. $M = (\{q_1, q_2, q_3, q_4\}, \{a, b\}, \delta, q_1, \{q_1, q_4\})$ where δ is defined by the following table.

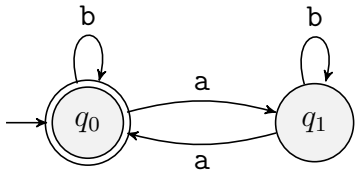
$q_i \backslash s$	a	b
q_1	q_1	q_2
q_2	q_3	q_4
q_3	q_2	q_1
q_4	q_3	q_4

3. We have the following state diagram.

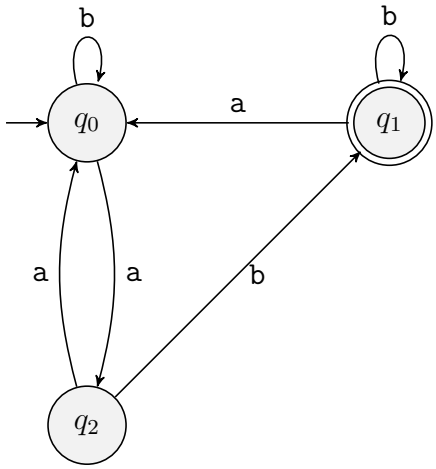


4. $q_3, q_2, q_3, q_2, q_3, q_4, q_3, q_4, q_5, q_4$ is a rejecting computation.

5. We have the following state diagram.

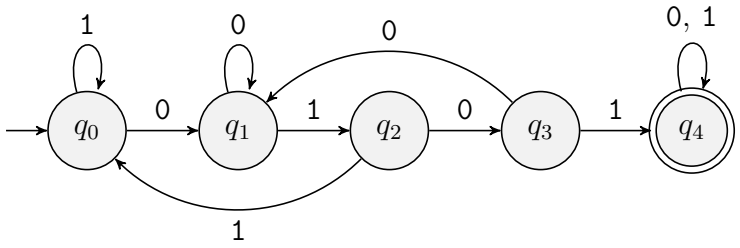


6. We have the following state diagram.

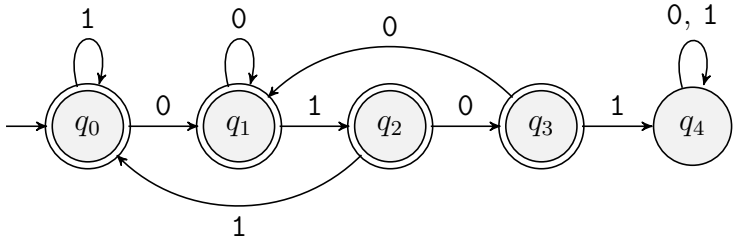


q_0: even number of a's; q_1: odd number of a's and ending with b; q_2: odd number of a's and ending with a;

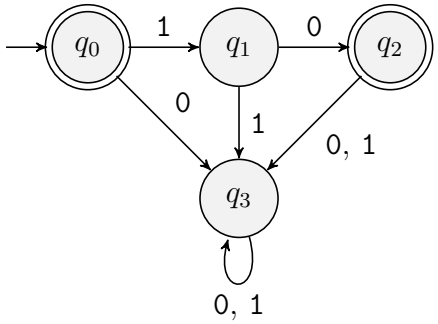
7. We have the following state diagram.



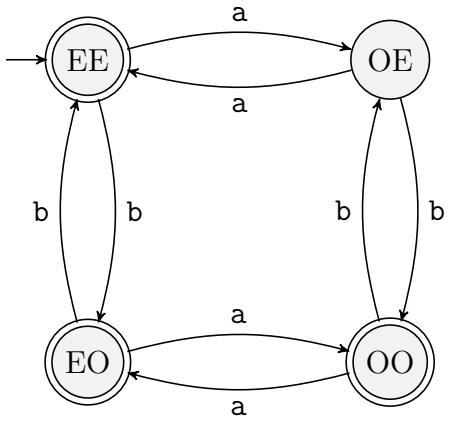
8. We have the following state diagram.



9. We have the following state diagram.

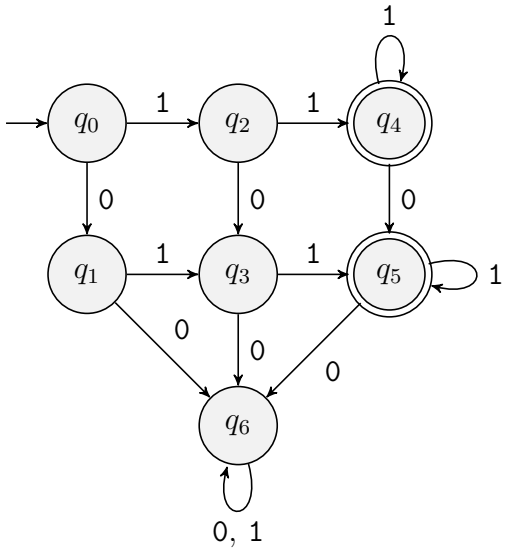


10. We have the following state diagram. For example, state EO means that an even number of a's and an odd number of b's have been read up to this point.


EE
OE
EO
OO
a
a
b
b
b
b
a
a

Additional Exercises

1. Describe the set of binary words that are accepted by the DFA given below. Prove your answer! Hint: consider the structure of the graph, in terms of the paths from initial state to accepting states.



2. Consider the alphabet

$$\Sigma = \left\{ \begin{matrix} 0 \\ 0 \end{matrix}, \begin{matrix} 0 \\ 1 \end{matrix}, \begin{matrix} 1 \\ 0 \end{matrix}, \begin{matrix} 1 \\ 1 \end{matrix} \right\}$$

where each symbol is a column vector of two bits. Provide the state diagram for a DFA that accepts all words w over Σ for which the top layer of w represents a binary number that is greater than the binary number represented by the bottom layer. For example, the DFA should accept

$$w_1 = \begin{matrix} 01011 \\ 00110 \end{matrix}$$

since the top layer represents the number 11, and the bottom layer represents the number 6. However, it should reject

$$w_2 = \begin{matrix} 10001 \\ 10011 \end{matrix}$$

since the top layer represents the number 17, the bottom layer 19, and $17 < 19$.

Solutions to Additional Exercises

1. The DFA accepts all words that have at least two 1's and at most one 0. **Proof.** Notice that accepting-state q_4 can only first be reached via the input prefix 11, while accepting-state q_5 can only first be reached either by input prefex 011 or 101. Thus, upon first reaching either accepting state with an input word, that word must have a prefix with at least two 1's and at most one 0. Moreover, any additional number of 1's keeps the computation in either q_4 or q_5. The only way to (permanently) escape both of these states is when a second 0 gets read.

2. Let $M = (\{\text{a,b,c}\}, \Sigma, \delta, \text{a}, \{\text{b}\})$, where Σ is defined by the problem statement. State a: the numbers read so far are equal, b: the top number is greater than the bottom, c: the bottom number is greater than the top. The following table provides δ.

$Q\backslash\Sigma$	$\begin{matrix}0\\0\end{matrix}$	$\begin{matrix}0\\1\end{matrix}$	$\begin{matrix}1\\0\end{matrix}$	$\begin{matrix}1\\1\end{matrix}$
a	a	c	b	a
b	b	b	b	b
c	c	c	c	c

   Once the top number has been determined to be greater (respectively, smaller) than the bottom number, it enters the permanent state b (respectively, c).