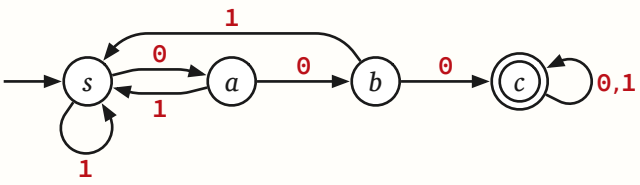
1 DFAs

Describe deterministic finite-state automata that accept each of the following languages over the alphabet $\Sigma = \{\mathtt{0}, \mathtt{1}\}$. Describe briefly what each state in your DFAs *means*.

1. All strings containing the substring **000**.

Solution:

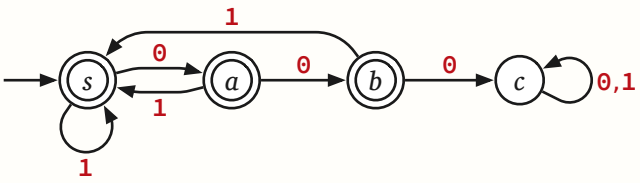


- s: We didn't just read a **0**
- a: We've read one **0** since the last **1** or the start of the string.
- b: We've read two **0**s since the last **1** or the start of the string.
- c: We've read the substring **000**.

■

2. All strings *not* containing the substring **000**.

Solution:

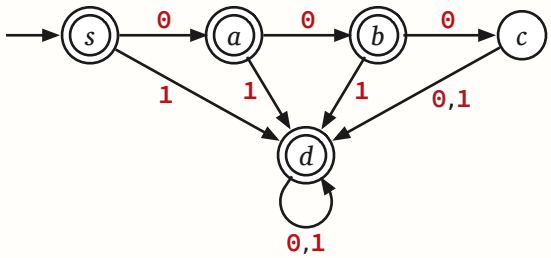


- s: We didn't just read a **0**
- a: We've read one **0** since the last **1** or the start of the string.
- b: We've read two **0**s since the last **1** or the start of the string.
- c: We've read the substring **000**.

(Yes, these are the same states as in problem 1.) ■

3. Every string except **000**. *[Hint: Don't try to be clever.]*

Solution:



- s: We haven't read anything yet
- a: Input so far is **0**.
- b: Input so far is **00**.
- c: Input so far is **000**.
- d: Input is not **000**; accept.

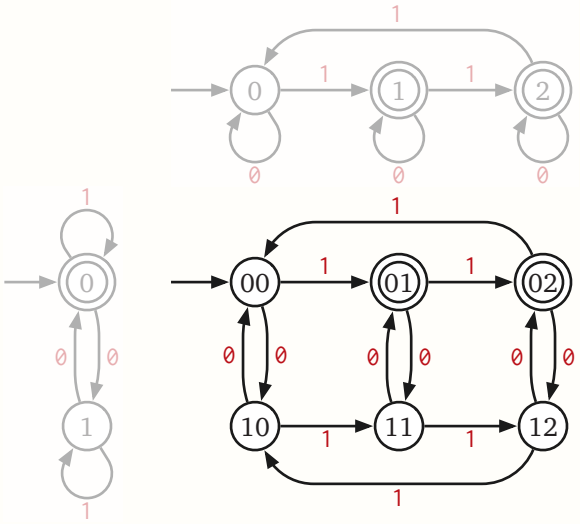
■

4. All strings in which the number of **0**s is even **and** the number of **1**s is *not* divisible by 3.

Solution: We use a standard product construction of two DFAs, one accepting strings with an even number of **0**s, and the other accepting strings where the number of **1**s is not a multiple of 3. The product DFA has six states, each labeled with a pair of integers, one indicating the number **0**s read modulo 2, the other indicating the number of **1**s read modulo 3.

$$Q := \{0,1\} \times \{0,1,2\}$$
$$s := (0,0)$$
$$A := \{(0,1),(0,2)\}$$
$$\delta((q,r),\mathbf{0}) := (q+1 \bmod 2,\ r)$$
$$\delta((q,r),\mathbf{1}) := (q,\ r+1 \bmod 3)$$

In this case, the product DFA is simple enough that we can just draw it out in full. I've drawn the two factor DFAs (in gray) to the left and above for reference.



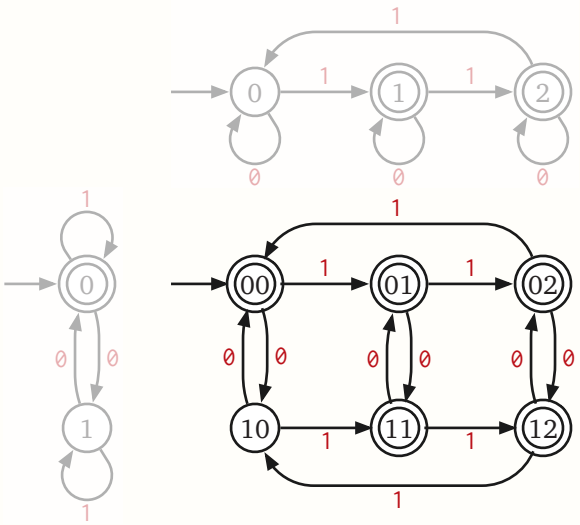
■

5. All strings in which the number of **0**s is even **or** the number of **1**s is *not* divisible by 3.

Solution: We use a standard product construction of two DFAs, one accepting strings with an even number of **0**s, and the other accepting strings where the number of **1**s is not a multiple of 3. The product DFA has six states, each labeled with a pair of integers, one indicating the number **0**s read modulo 2, the other indicating the number of **1**s read modulo 3.

$$Q := \{0,1\} \times \{0,1,2\}$$
$$s := (0,0)$$
$$A := \{(0,0),(0,1),(0,2),(1,1),(1,2)\}$$
$$\delta((q,r),\mathbf{0}) := (q+1 \bmod 2,\ r)$$
$$\delta((q,r),\mathbf{1}) := (q,\ r+1 \bmod 3)$$

In this case, the product DFA is simple enough that we can just draw it out in full. I've drawn the two factor DFAs (in gray) to the left and above for reference.



This is exactly the same DFA as problem 1, except for the accepting states. Similar standard product constructions yield DFAs for any other boolean combination of these two factor languages, including the following:

- $\{w \mid \#(\mathbf{0},w)$ is even and $\#(\mathbf{1},w)$ is divisible by $3\}$
- $\{w \mid \#(\mathbf{0},w)$ is odd or $\#(\mathbf{1},w)$ is divisible by $3\}$
- $\{w \mid$ if $\#(\mathbf{0},w)$ is even, then $\#(\mathbf{1},w)$ is not divisible by $3\}$
- $\{w \mid$ if $\#(\mathbf{1},w)$ is not divisible by 3, then $\#(\mathbf{0},w)$ is even$\}$
- $\{w \mid \#(\mathbf{0},w)$ is even if and only if $\#(\mathbf{1},w)$ is not divisible by $3\}$
- $\{w \mid \#(\mathbf{0},w)$ is odd if and only if $\#(\mathbf{1},w)$ is divisible by $3\}$

This is why, whenever you describe a product construction, especially for a homework or exam problem, you ***must*** specify the set of accepting states. ■

6. Given DFAs M_1 and M_2, all strings in $\overline{L(M_1)} \oplus L(M_2)$.

Solution: We use a standard product construction of two DFAs, $M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$. As observed in the solution the problem 2, the major difference is the set of accepting states. Here

$$\begin{aligned} w \in \overline{L(M_1)} \oplus L(M_2) &\iff w \in \overline{L(M_1)} \textbf{ xor } w \in L(M_2) \\ &\iff w \notin L(M_1) \textbf{ xor } w \in L(M_2). \end{aligned}$$

so a state (q, r) should be accepting if and only if $q \notin A_1$ *xor* $r \in A_2$.

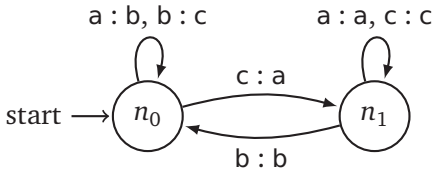
$$\begin{aligned} Q &:= Q_1 \times Q_2 \\ s &:= (s_1, s_2) \\ A &:= \{(q, r) \in Q \mid q \notin A_1 \textbf{ xor } r \in A_2\} \end{aligned}$$

$$\delta((q, r), a) := (\delta_1(q, a), \delta_2(r, a))$$

Since M_1 and M_2 were given abstractly, the product DFA can only be specified mathematically. ■

2 Other types of automata

1. A *finite-state transducer* (FST) is a type of deterministic finite automaton whose output is a string instead of just *accept* or *reject*. The following is the state diagram of finite state transducer FST_0.



Each transition of an FST is labeled at least an input symbol and an output symbol, separated by a colon (:). There can also be multiple input-output pairs for each transitions, separated by a comma (,). For instance, the transition from n_0 to itself can either take a or b as an input, and outputs b or c respectively.

When an FST computes on an input string $s := \overline{s_0 s_1 \dots s_{n-1}}$ of length n, it takes the input symbols $s_0, s_1, \dots, s_{n-1}$ one by one, starting from the starting state, and produces corresponding output symbols. For instance, the input string abccba produces the output string bcacbb, while cbaabc produces abbbca.

(a) Each of the following strings is the input of FST_0. Give the sequence of states entered and the output produced.

- aaca

Solution: The states: $n_0 \to n_0 \to n_0 \to n_1 \to n_1$
The output: `bbaa` ■

- `cbbc`

Solution: The states: $n_0 \to n_1 \to n_0 \to n_0 \to n_1$
The output: `abca` ■

- `bcba`

Solution: The states: $n_0 \to n_0 \to n_1 \to n_0 \to n_0$
The output: `cabb` ■

- `acbbca`

Solution: The states: $n_0 \to n_0 \to n_1 \to n_0 \to n_0 \to n_1 \to n_1$
The output: `babcaa` ■

(b) Assume that FST_1 has an input alphabet Σ_1 and an output alphabet Γ_1, give a formal definition of this model and its computation. (Hint: An FST is a 5-tuple with no accepting states. Its transition function is of the form $\delta : Q \times \Sigma \to Q \times \Gamma$.)

Solution: HW problem. ■

(c) Give a formal description of FST_0.

Solution: HW problem. ■

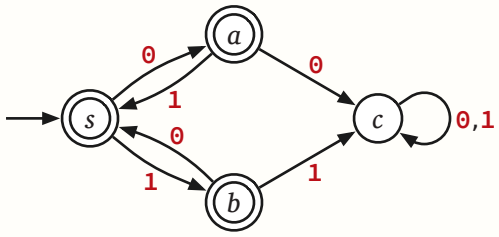
(d) Give a state diagram of an FST with the following behavior. Its input and output alphabets are $\{\mathsf{T}, \mathsf{F}\}$. Its output string is inverted on the positions with indices divisible by 3 and is identical on all the other positions. For instance, on an input TFTTFTFT it should output FFTFFTTT.

Solution: HW problem. ■

Work on these later:

7. All strings w such that *in every prefix of* w, the number of **0**s and **1**s differ by at most 1.

Solution: This is the same as the set of strings that alternate between **0**s and **1**s.

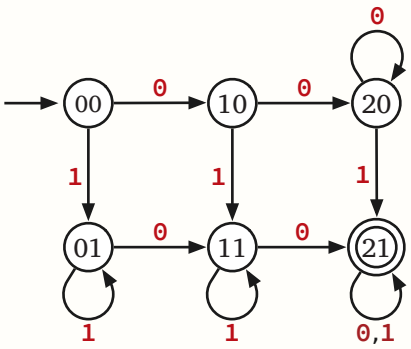


- s: We haven't read anything yet
- a: Input so far is an alternating string ending in **0**.
- b: Input so far is an alternating string ending in **1**.
- c: We've seen the substring **00** or **11**; reject.

■

8. All strings containing at least two **0**s and at least one **1**.

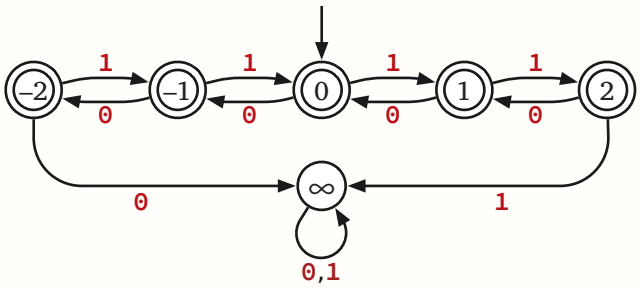
Solution:



Each state is labeled with a pair of integers. The first integer indicates the number of **0**s read so far (up to 2), and the second indicates the number of **1**s read so far (up to 1). ■

9. All strings w such that *in every prefix of* w, the number of **0**s and **1**s differ by at most 2.

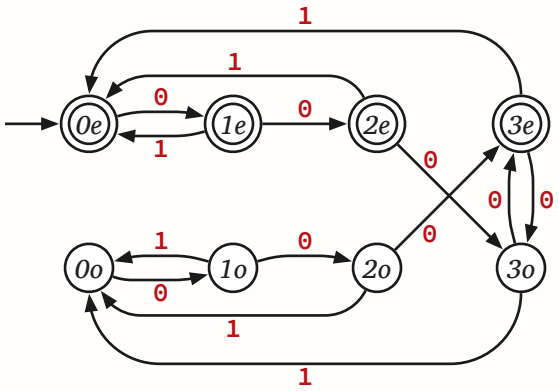
Solution:



The fail state ∞ indicates that we have read some prefix where the number of **0**s and **1**s differ by more than 2. Each of the other states states $-2, -1, 0, 1, 2$ indicates the number of **1**s minus the number of **0**s of the prefix read so far. ■

*10. All strings in which the substring **000** appears an even number of times.
(For example, **0001000** and **0000** are in this language, but **00000** is not.)

Solution:



Each state is labeled with an integer from 0 to 3, indicating how many consecutive **0**s have just been read, and a letter e or o, indicating whether we have read an even or odd number of **000** substrings. ■

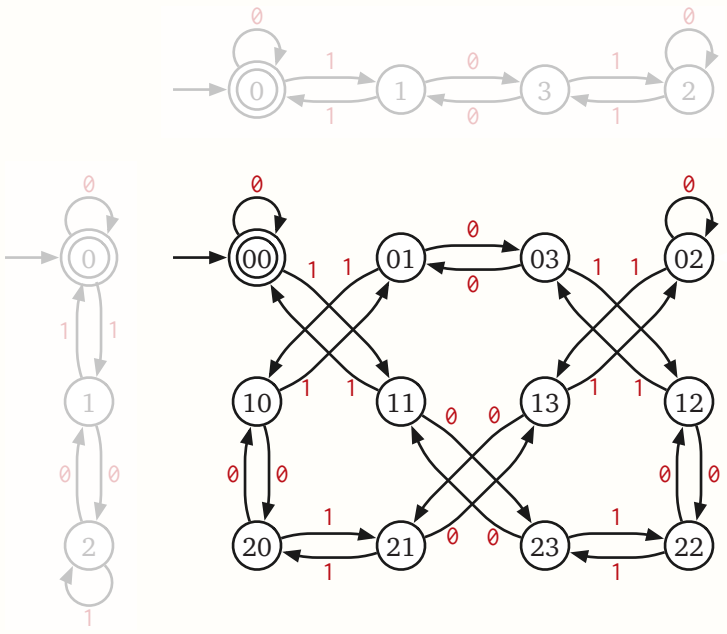
11. All strings that are **both** the binary representation of an integer divisible by 3 **and** the ternary (base-3) representation of an integer divisible by 4. For example, the string **1100** is an element of this language, because it represents $2^3 + 2^2 = 12$ in binary and $3^3 + 3^2 = 36$ in ternary.

Solution: Again, we use a standard product construction of two DFAs, one accepting binary strings divisible by 3, the other accepting ternary strings divisible by 4. The product DFA has twelve states, each labeled with a pair of integers: The binary value read so far modulo 3, and the ternary value read so far modulo 4.

$$Q := \{0,1,2\} \times \{0,1,2,3\}$$
$$s := (0,0)$$
$$A := \{(0,0)\}$$

$$\delta((q,r),\mathbf{0}) := (2q \bmod 3, \quad 3r \bmod 4)$$
$$\delta((q,r),\mathbf{1}) := (2q+1 \bmod 3, \; 3r+1 \bmod 4)$$

For reference, here is a drawing of the DFA, with the two factor DFAs (in gray) to the left and above. We wouldn't expect you to draw this, especially on exams. Or more accurately: We would expect you ***not*** to draw this, ***especially*** on exams. The states of the factor DFA that maintains ternary-value-mod-4 are deliberately "out of order" to simplify the drawing.



■

*12. All strings w such that $F_{\#(\mathtt{10},w)} \bmod 10 = 4$, where $\#(\mathtt{10}, w)$ denotes the number of times **10** appears as a substring of w, and F_n is the nth Fibonacci number:

$$F_n = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F_{n-1} + F_{n-2} & \text{otherwise} \end{cases}$$

Solution: Our DFA has 200 states, each labeled with three values:

- $F_k \bmod 10$, where k is the number of times we have seen the substring **10**.
- $F_{k+1} \bmod 10$, where k is the number of times we have seen the substring **10**.
- The last symbol read (or **0** if we have read nothing yet)

Here is the formal description:

$$\begin{aligned} Q &:= \{0,1,2,3,4,5,6,7,8,9\} \times \{0,1,2,3,4,5,6,7,8,9\} \times \{\mathtt{0},\mathtt{1}\} \\ s &:= (0,1,\mathtt{0}) \\ A &:= \{(4,r,a) \mid r \in \{0,1,2,3,4,5,6,7,8,9\} \text{ and } a \in \{\mathtt{0},\mathtt{1}\}\} \end{aligned}$$

$$\begin{aligned} \delta((q,r,\mathtt{0}),\mathtt{0}) &:= (q,r,\mathtt{0}) \\ \delta((q,r,\mathtt{1}),\mathtt{0}) &:= (r, q+r \bmod 10, \mathtt{0}) \\ \delta((q,r,\mathtt{0}),\mathtt{1}) &:= (q,r,\mathtt{1}) \\ \delta((q,r,\mathtt{1}),\mathtt{1}) &:= (q,r,\mathtt{1}) \end{aligned}$$

The transition function exploits the recursive definition $F_{k+1} = F_k + F_{k-1}$. ■

Solution (sketch): The Fibonacci numbers modulo 10 define a repeating sequence with period 60. So this language can be accepted by a DFA with "only" 120 states. ■