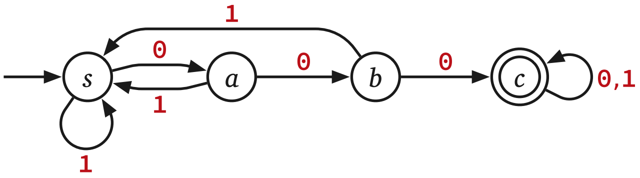
Describe deterministic finite-state automata that accept each of the following languages over the alphabet $\Sigma = \{\texttt{0}, \texttt{1}\}$. Describe briefly what each state in your DFAs *means*.

1. All strings containing the substring `000`.

Solution:

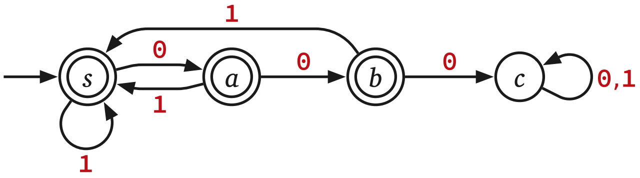


- s: We didn't just read a `0`
- a: We've read one `0` since the last `1` or the start of the string.
- b: We've read two `0`s since the last `1` or the start of the string.
- c: We've read the substring `000`.

■

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

Solution:

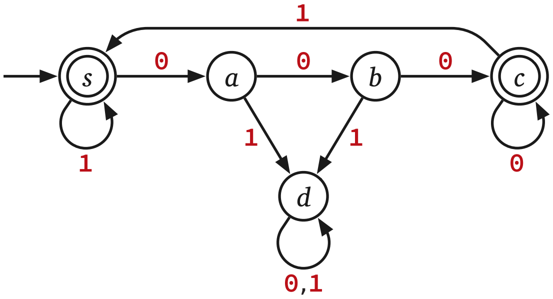


- s: We didn't just read a `0`
- a: We've read one `0` since the last `1` or the start of the string.
- b: We've read two `0`s since the last `1` or the start of the string.
- c: We've read the substring `000`.

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

3. All strings in which every run of 0s has length at least 3.

Solution:

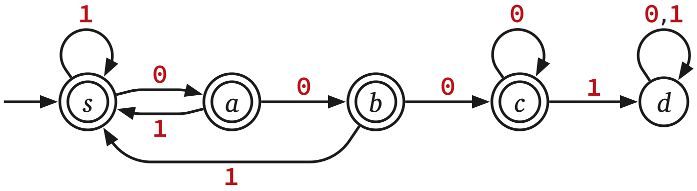


- *s*: We did not just read a 0
- *a*: We've read one 0 since the last 1 or the start of the string.
- *b*: We've read two 0s since the last 1 or the start of the string.
- *c*: We've read at least three 0s since the last 1 or the start of the string.
- *d*: We've read the substring 01 or 001; reject.

■

4. All strings in which no substring 000 appears before a 1.
(Equivalently: All strings in which every substring 000 appears after every 1.)

Solution: A string is in this language if and only if it does not contain the substring 0001.

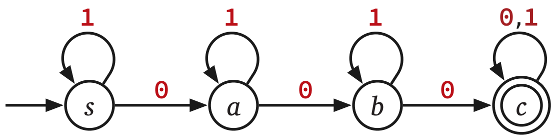


- *s*: We did not just read a 0
- *a*: We've read one 0 since the last 1 or the start of the string.
- *b*: We've read two 0s since the last 1 or the start of the string.
- *c*: We've read at least three 0s since the last 1 or the start of the string
- *d*: We've read the substring 0001; reject.

■

5. All strings containing at least three 0s.

Solution:

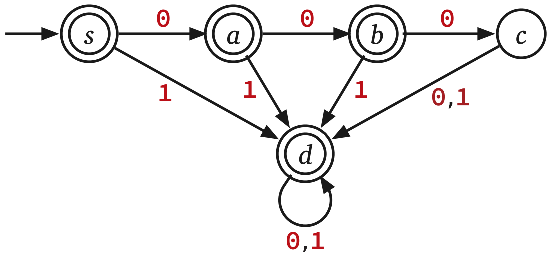


- s: We've read no 0s.
- a: We've read one 0.
- b: We've read two 0s.
- c: We've read at least three 0s; accept.

■

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

Solution:



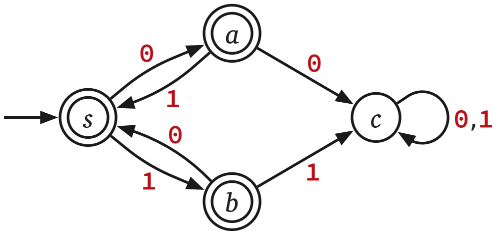
- s: We haven't read anything yet
- a: Input so far is 0.
- b: Input so far is 00.
- c: Input so far is 000.
- d: Input is not 000; accept.

■

Work on these later:

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

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

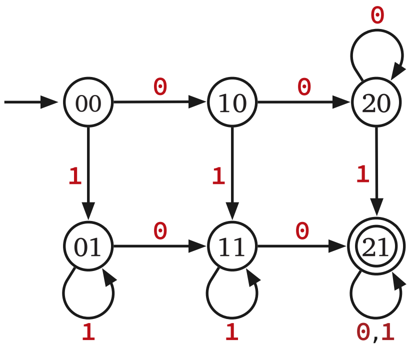


- s: We haven't read anything yet
- a: Input so far is an alternating string ending in 0.
- b: Input so far is an alternating string ending in 1.
- c: We've seen the substring 00 or 11; reject.

■

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

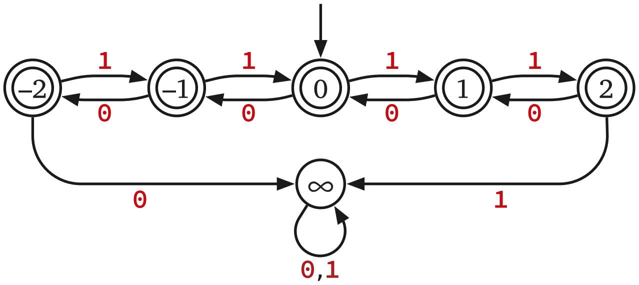
   Solution:



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

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

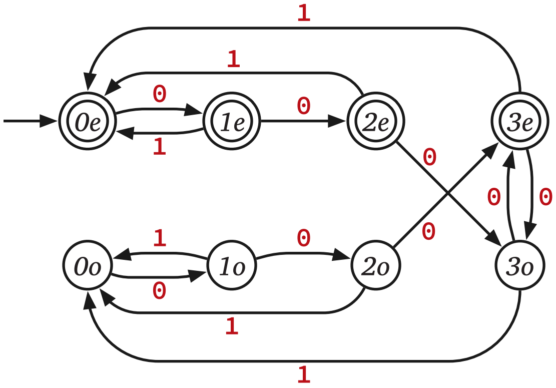
Solution:



The fail state ∞ indicates that we have read some prefix where the number of 0s and 1s differ by more than 2. Each of the other states states $-2,-1,0,1,2$ indicates the number of 1s minus the number of 0s 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 0s have just been read, and a letter e or o, indicating whether we have read an even or odd number of 000 substrings. ■