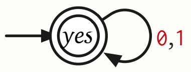
Describe deterministic finite-state automata that accept each of the following languages over the alphabet $\Sigma = \{0, 1\}$. Give the states of your DFAs mnemonic names, and describe briefly *in English* the meaning or purpose of each state.

0. All strings.

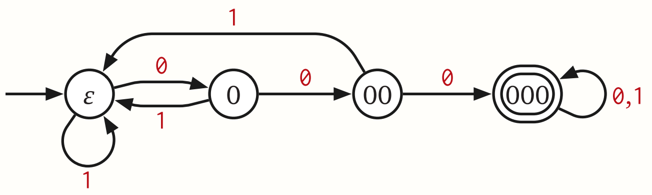
Solution:



The purpose of the only state is to accept. ■

1. All strings containing the substring 000.

Solution:

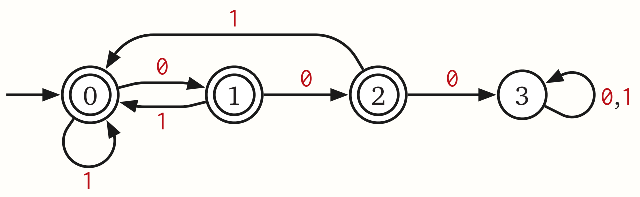


- ε: We didn't just read a 0.
- 0: We've read one 0 since the last 1 or the start of the string.
- 00: We've read two 0s since the last 1 or the start of the string.
- 000: We've read the substring 000; accept!

■

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

Solution:

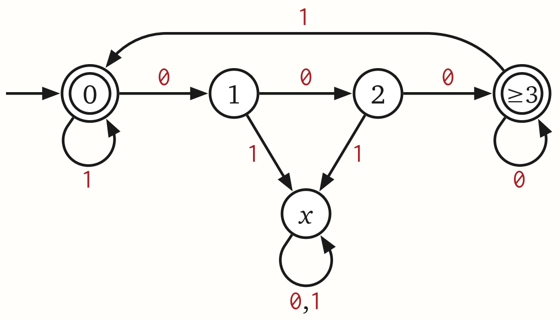


- 0: We didn't just read a 0
- 1: We've read one 0 since the last 1 or the start of the string.
- 2: We've read two 0s since the last 1 or the start of the string.
- 3: We've read the substring 000; reject!

(Yes, these are exactly the same states as in problem 1, with different names.) ■

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

Solution:

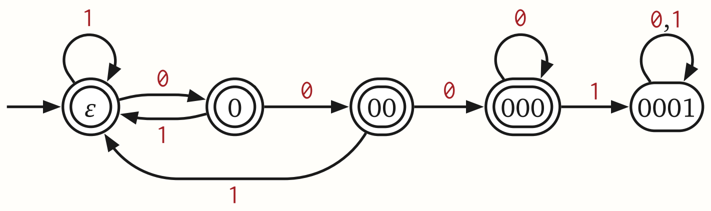


- 0: We did not just read a 0.
- 1: We've read one 0 in the current run of 0s.
- 2: We've read two 0s in the current run of 0s.
- ≥3: We've read at least three 0s in the current run of 0s.
- x: We've already seen a run of 0s of length less than 3; reject.

(It's okay to omit the dump state x, as long as you include the magic sentence "All unspecified transitions lead to a dump state.") ■

4. All strings in which the last 1 (if any) appears before the first substring 000 (if any).

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

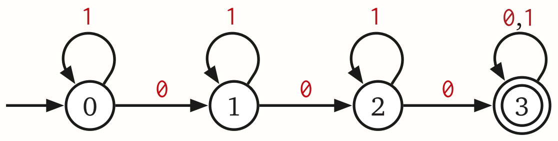


- ε: We did not just read a 0
- 0: We've read one 0 since the last 1 or the start of the string.
- 00: We've read two 0s since the last 1 or the start of the string.
- 000: We've read at least three 0s since the last 1 or the start of the string
- 0001: We've read the substring 0001; reject.

(Again, it's okay to omit the dump state 0001, as long as you include the magic sentence "All unspecified transitions lead to a dump state.") ■

5. All strings containing at least three 0s.

Solution:

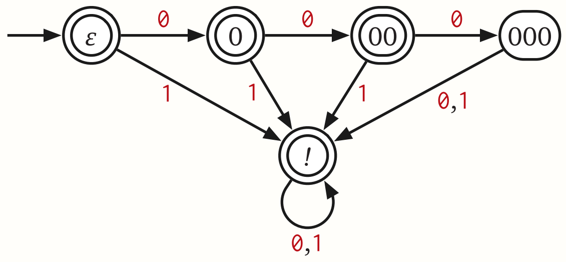


- 0: We've read no 0s.
- 1: We've read one 0.
- 2: We've read two 0s.
- 3: We've read at least three 0s; accept.

■

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

Solution:



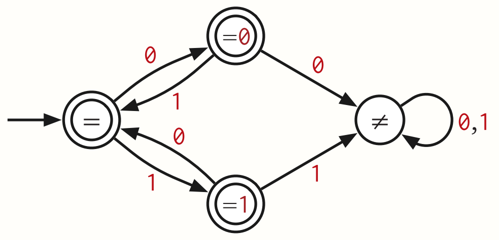
- ε: We haven't read anything yet.
- 0: Input so far is 0.
- 00: Input so far is 00.
- 000: Input so far is 000.
- !: Input string is not 000; accept!

■

Harder problems to think about later:

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

Solution: Equivalently, strings in which every even-length prefix has the same number of 0s and 1s.

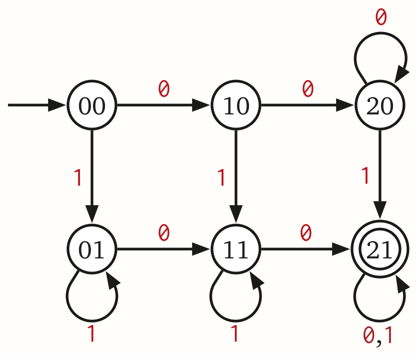


- $=$: We've read an even-length prefix with the same number of 0s and 1s
- $=$0: We've read an even-length prefix with the same number of 0s and 1s, followed by a 0.
- $=$1: We've read an even-length prefix with the same number of 0s and 1s, followed by a 1.
- $\neq$: We've read an even-length prefix with different numbers of 0s and 1s; reject.

■

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

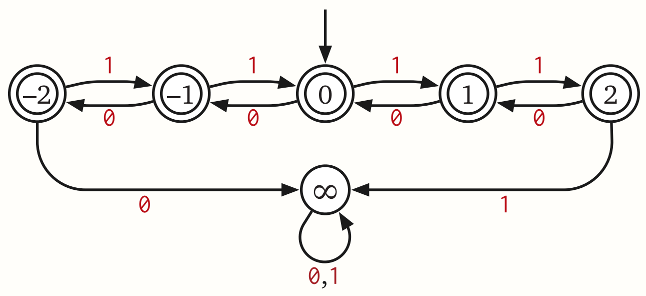
Solution:



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). This DFA is the result of a standard product construction. ■

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

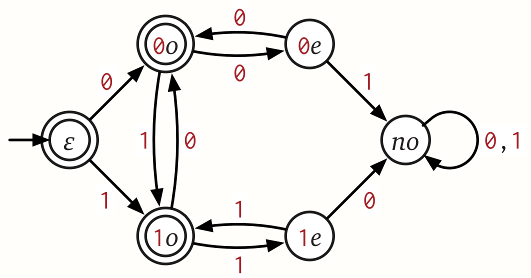
Solution:



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 every run has odd length. (For example, `0001` and `100000111` and the empty string ε are in this language, but `000000` and `001000` are not.)

Solution:

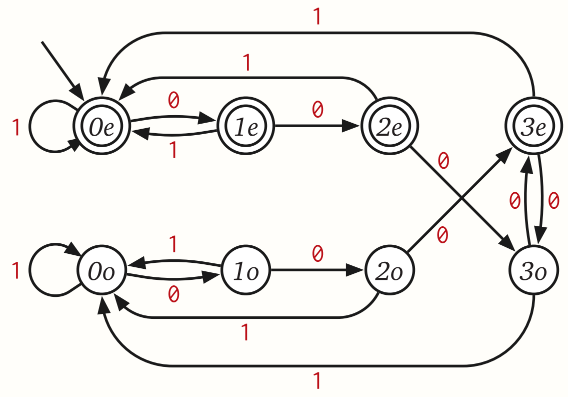


- ε: We haven't read anything yet.
- `0`o: The prefix we have read so far ends with an odd-length run of `0`s, and all previous runs have odd length.
- `0`e: The prefix we have read so far ends with an even-length run of `0`s, and all previous runs have odd length.
- `1`o: The prefix we have read so far ends with an odd-length run of `1`s, and all previous runs have odd length.
- `1`e: The prefix we have read so far ends with an even-length run of `1`s, and all previous runs have odd length.
- *no*: We have read a complete run with even length; reject.

(Again, it's okay to omit the dump state *no*, as long as you include the magic sentence "All unspecified transitions lead to a dump state.") ■

*11. All strings in which the substring `000` appears an even number of times. (For example, `01100` and `000000` and the empty string ε are in this language, but `00000` and `001000` are not.)

Solution:



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