Finite-State Machines

Computer Science 111
Boston University

Vahid Azadeh-Ranjbar, Ph.D.

CS 111: A Breadth-Based Introduction

- Five major units:
  - ✓ computational problem solving and "functional" programming
  - ✓ a look "under the hood" (computer organization)
  - ✓ imperative programming
  - ✓ object-oriented programming
  - • **topics from CS theory**

Finite State Machine (FSM)

- FSM is a **mathematical model of computation**. It is an abstract machine that can be in exactly one of a finite number of states at any given time.
- Let me explain FSM with an example

  ➢ Given a long sequence of 0's and 1's, how machines determine if there is an odd number of 1's in the sequence?

    ❖For example for '10011', a machine's output is a **yes** and for '1001', it is a **no**.

Finite State Machine (FSM)

- In **state-based** reasoning,
  - machines identify different situations as ***states***
  - how one state changes to another is called ***state transition***
- In our problem, we want to read a text file that is filled with a sequence of 0's and 1's one character at a time and keep track whether we have currently seen an even number of 1's or an odd number.

Is there an odd number
of 1's in a sequence?

Finite State Machine (FSM)

- In our problem, we want to read a text file that is filled with a sequence of 0's and 1's one character at a time and keep track whether we have currently seen an even number of 1's or an odd number.

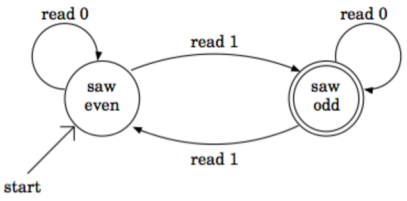


Figure1: FSM for parity checking

Is there an odd number of 1's in a sequence?

- This diagram shows a FSM with **two states** (circles labeled **saw odd** and **saw even**)
- **Four transitions** (arrows drawn from one state to another and labeled either "read 1" or "read 0").

Finite State Machine (FSM)

- The idea is we enter the FSM via the arrow labeled "start"
- and then when we are in a state, we read a single character
- and follow the appropriate transition based on the character we read.
- If the input is '10011' then the following list shows the sequence of states that will be visited:

  even odd odd odd even odd

  '' '1' '10' '100' '1001' '10011'
- The double circle on the state "saw odd" indicates it is the ***accept* state** for our FSM.
- If we end up in the accept state at the end of execution then we have positively answered our question.

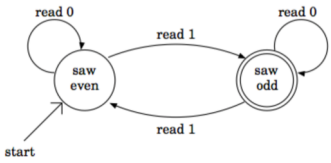
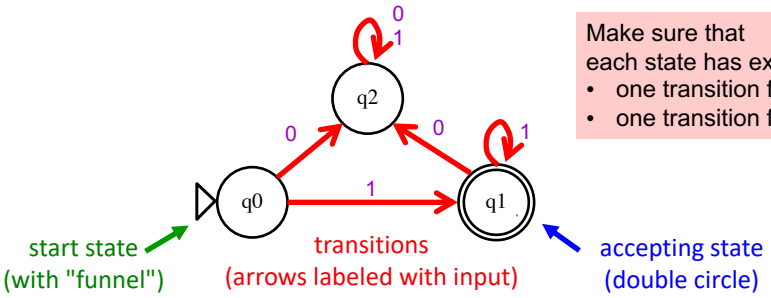


Figure1: FSM for parity checking

Is there an odd number of 1's in a sequence?

Finite State Machine (FSM)

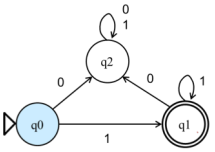
- An abstract model of computation
- Consists of:
  - one or more states (the circles)
    - *exactly one* of them is the *start / initial state*
    - *zero or more* of them can be an *accepting state*
  - a set of possible input characters (we're using {0, 1})
  - *transitions* between states, based on the inputs



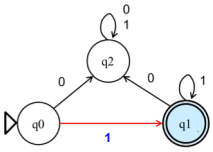
Make sure that each state has exactly:
- one transition for 0
- one transition for 1

Accepting an Input Sequence

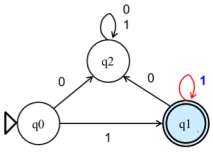
- We can use an FSM to test if an input meets some criteria.
- An FSM *accepts* an input if the transitions produced by the input leave the FSM in an accepting state.
- Example: input **111** on the FSM from the last slide



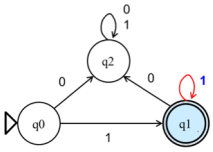
initial state



transition caused by first **1**



transition caused by next **1**

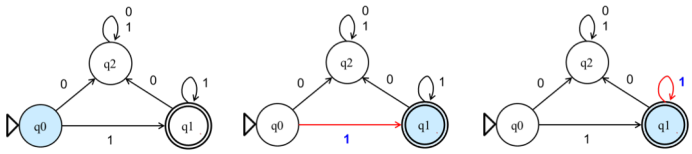


transition caused by final **1**

111 *is* accepted!

Rejecting an Input Sequence

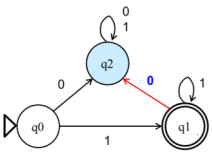
- An FSM *rejects* an input if the transitions produced by the input do *not* leave the FSM in an accepting state.
- Example: input **1101** on the FSM from the last slide



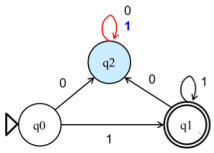
initial state

transition from first **1**

transition from next **1**



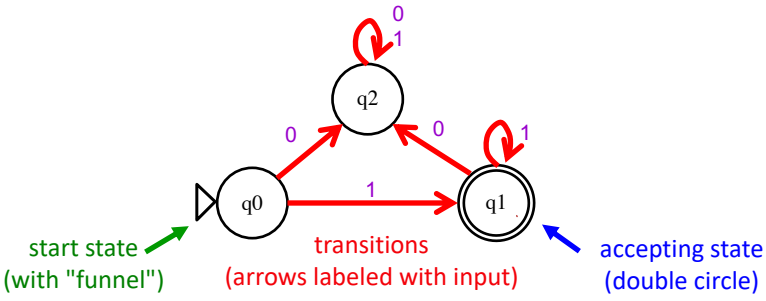
transition from **0**



transition from final **1**

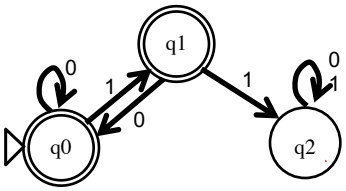
1101 is *not* accepted

Which Bit Strings Does This FSM Accept?



Bit strings containing only 1s

Which of these inputs is accepted?

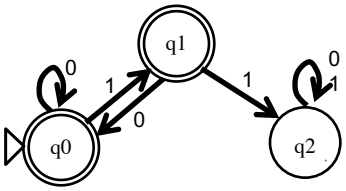


A. 0000
B. 10010001
C. 00111
D. A and B
E. A and C

In general, what English phrase describes the ***inputs accepted*** by this FSM?

What does each state say about the ***input seen thus far****?*
q0:
q1:
q2:

Which of these inputs is accepted?

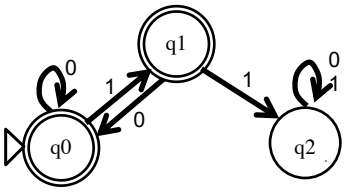


A. 0000
B. 10010001
C. 00111
D. **A and B**
E. A and C

In general, what English phrase describes the ***inputs accepted*** by this FSM?

What does each state say about the ***input seen thus far****?*
q0:
q1:
q2:

Which of these inputs is accepted?

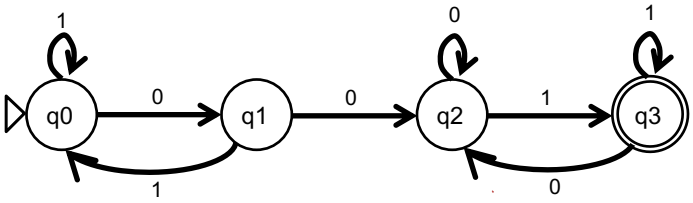


A. 0000
B. 10010001
C. 00111
D. **A and B**
E. A and C

In general, what English phrase describes the ***inputs accepted*** by this FSM?
inputs that don't include two or more 1s in a row

What does each state say about the ***input seen thus far****?*
q0: empty string *or*
at most one 1 in a row; last digit == 0
q1: at most one 1 in a row; last digit == 1
q2: two or more 1s in a row

Which of these inputs is accepted?

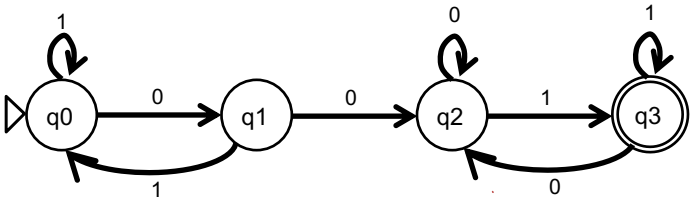


A. 0101
B. 10010
C. 0011101
D. two or more
E. none of them

In general, what English phrase describes the ***inputs accepted*** by this FSM?

What does each state say about the ***input seen thus far****?*
q0:
q1:
q2:
q3:

Which of these inputs is accepted?

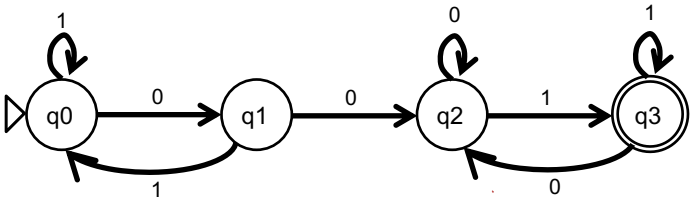


A. 0101
B. 10010
C. **0011101**
D. two or more
E. none of them

In general, what English phrase describes the ***inputs accepted*** by this FSM?

What does each state say about the ***input seen thus far****?*
q0:
q1:
q2:
q3:

Which of these inputs is accepted?



A. 0101
B. 10010
C. **0011101**
D. two or more
E. none of them

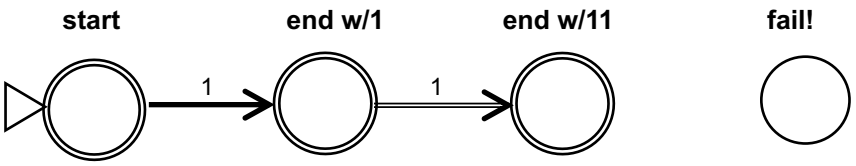
In general, what English phrase describes the ***inputs accepted*** by this FSM?
inputs that include at least two 0s in a row and that end with a 1

What does each state say about the ***input seen thus far****?*
q0: empty or one 0 in a row, ends with 1
q1: one 0 in a row, ends with 0
q2: seen two 0s in a row, ends with 0
q3: seen two 0s in a row, ends with 1

No Occurrences of 110

Construct a FSM accepting strings that
do ***NOT*** contain the pattern **110.**

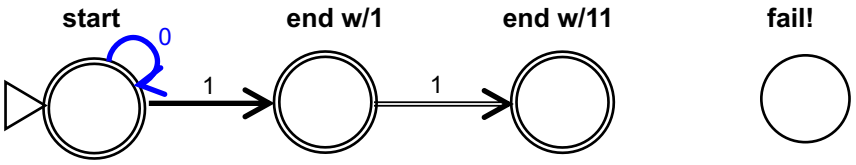
Accepted: 1010001, 00111, ... **Rejected:** 10100**110**0, 00**110**101, 1**110**, ...



No Occurrences of 110

Construct a FSM accepting strings that
do ***NOT*** contain the pattern **110.**

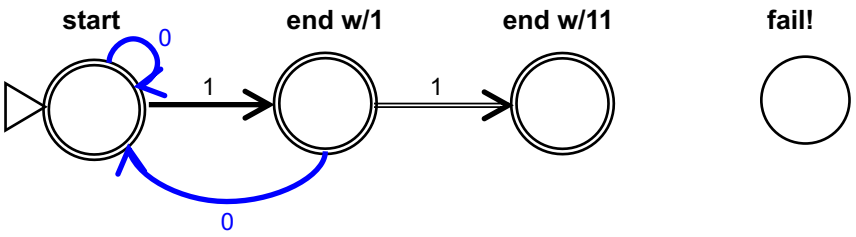
Accepted: 1010001, 00111, ... **Rejected:** 10100**110**0, 00**110**101, 1**110**, ...



No Occurrences of 110

Construct a FSM accepting strings that
do ***NOT*** contain the pattern **110.**

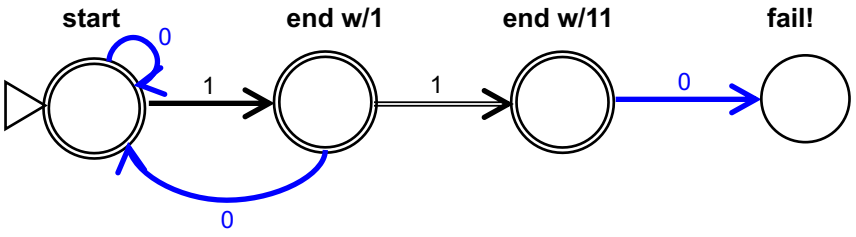
Accepted: 1010001, 00111, ... **Rejected:** 10100**110**0, 00**110**101, 1**110**, ...



No Occurrences of 110

Construct a FSM accepting strings that
do ***NOT*** contain the pattern **110.**

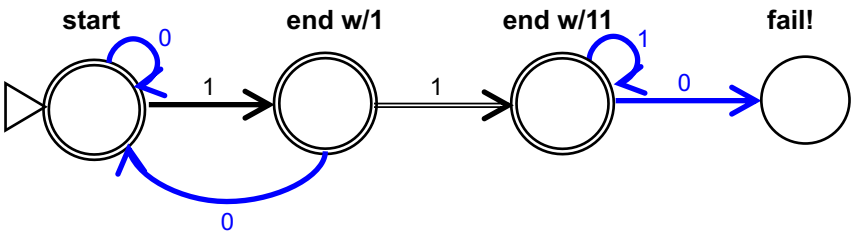
Accepted: 1010001, 00111, ... **Rejected:** 10100**110**0, 00**110**101, 1**110**, ...



No Occurrences of 110

Construct a FSM accepting strings that
do ***NOT*** contain the pattern **110.**

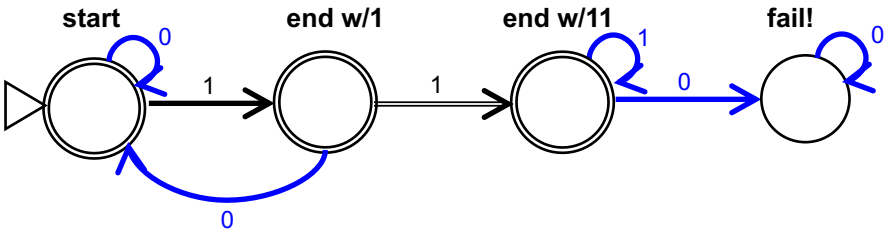
Accepted: 1010001, 00111, ... **Rejected:** 10100**110**0, 00**110**101, 1**110**, ...



No Occurrences of 110

Construct a FSM accepting strings that
do ***NOT*** contain the pattern **110.**

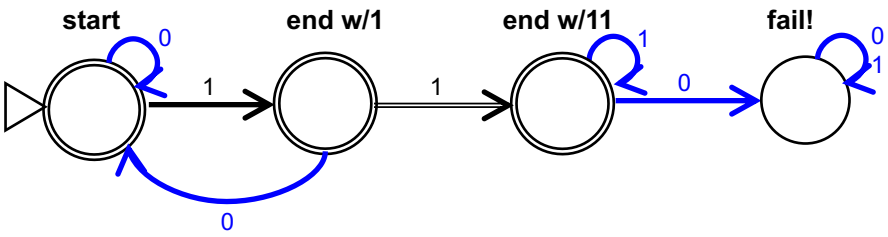
Accepted: 1010001, 00111, ... **Rejected:** 10100**110**0, 00**110**101, 1**110**, ...



No Occurrences of 110

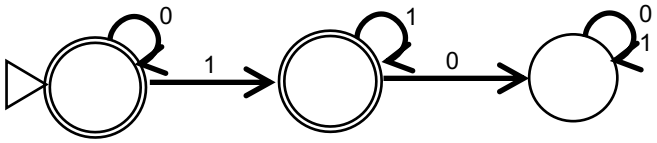
Construct a FSM accepting strings that do ***NOT*** contain the pattern **110.**

Accepted: 1010001, 00111, ... **Rejected:** 10100**110**0, 00**110**101, 1**110**, ...



No Occurrences of 110?

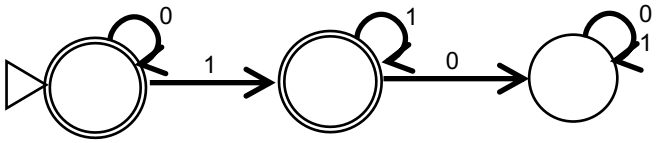
Does this alternative work?



No Occurrences of 110?

Does this alternative work?

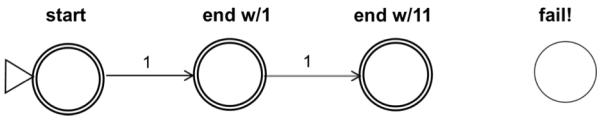
No. It also rejects the input if there's an occurrence of 10.



Build Your Own!

Modify the FSM below to create one that accepts strings that ***don't*** contain the pattern **110**.

Accepted: 1010001, 00111, ... **Rejected:** 10100**110**0, 00**110**101, 1**110**, ...



Construct a FSM that accepts strings in which:

- the number of **0**s is a multiple of 3 – i.e., strings with 0, 3, 6, … zeros
- the number of **1**s doesn't matter

Accepted: 110101110, 11, 0000010, ... **Rejected:** 101, 0000, 111011101111, ...