Deterministic Finite State Machine Problems

Use: Preparation

Problem

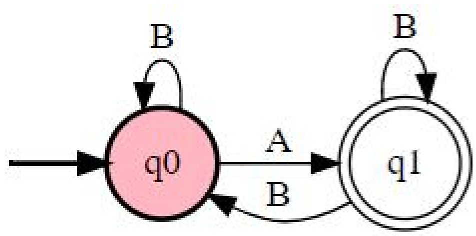
Consider these five terms: *alphabet*, *states*, *start state*, *accepting state*, *rules*. For each definition provided below, match it with the term from the above list that is most aligned with the definition.

- The first state of the machine when no input symbols have been read
- Specifies set of possible input symbols
- Specifies transitions between states as input symbols are read

Problem

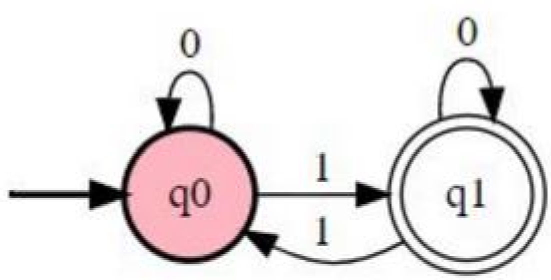
Given the finite state machine shown below, which of these three descriptions best describes the finite state machine? Choose the one best answer.

- This machine does NOT violate the constraint of *for each state, at most one transition on a given input symbol*
- This machine does violate the constraint of *for each state, at most one transition on a given input symbol*, and state `q0` is where the violation occurs
- This machine does violate the constraint of *for each state, at most one transition on a given input symbol*, and state `q1` is where the violation occurs



Problem

True or False: Given the finite state machine shown below, the trace that results from inputting the word 101 into the machine will be `q0`,`q1`,`q1`,`q0`.



Use: Warmup

Problem
Consider the alphabet (set of symbols) $\{a, b\}$. How many total one-letter and two-letter words can be generated from that alphabet?

Setup
Consider the finite state machine below in answering the following questions. You can assume that the alphabet for this machine is the set of symbols $\{a, b\}$.

Note that `a, b` on an arrow does **not** mean *an `a` symbol followed by a `b` symbol*; instead, it means *an `a` symbol or, independently, a `b` symbol.* It would be written as two separate transitions if written out in text:

- `so:a>s1`
- `s0:b>s1`

The machine allows checking for the following structural pattern: *For words made up of the symbols `a` and `b`, the second symbol in the word is a `b` symbol.*

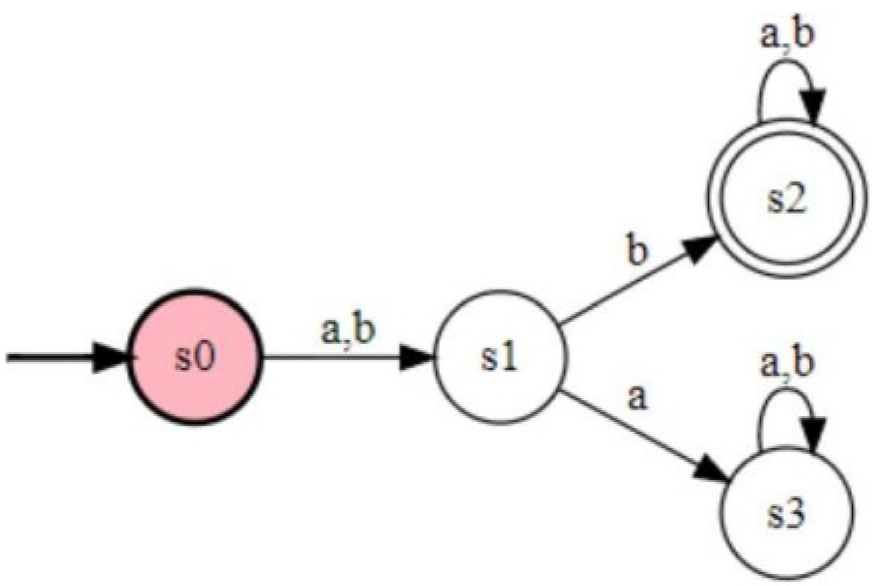


Figure 1: A finite state machine that, for the alphabet {*a*,*b*}, represents the language of strings with a `b` symbol as the second symbol.

Problem
How many states are there in the machine?

Problem
True or *False*: There is more than one accepting state in the machine.

Problem
Which of the following is not a transition encoded by the machine? *Choose all that apply.*

- `s0:a>s1`
- `s1:a>s1`
- `s1:b>s2`
- `s2:a>s2`
- `s2:b>s2`

Problem

How many transitions are there in the diagram of the machine?

Problem

How many of these four words are accepted by the machine: `b`, `ab`, `bab`, `bba`?

Problem

The trace `s0, s1, s3, s3` is a trace of which of the following words? *Choose all that apply* (more than one word could have the same trace).

- `a`
- `ab`
- `abb`
- `baa`
- `bbb`

Problem

True or *False*: All words *rejected* (not accepted) by this machine end in state `s3` when traced.

Problem

Which, if any, states in the machine are *not on a path to an accepting state* and could be removed from the drawing?

Use: Advanced

Problem

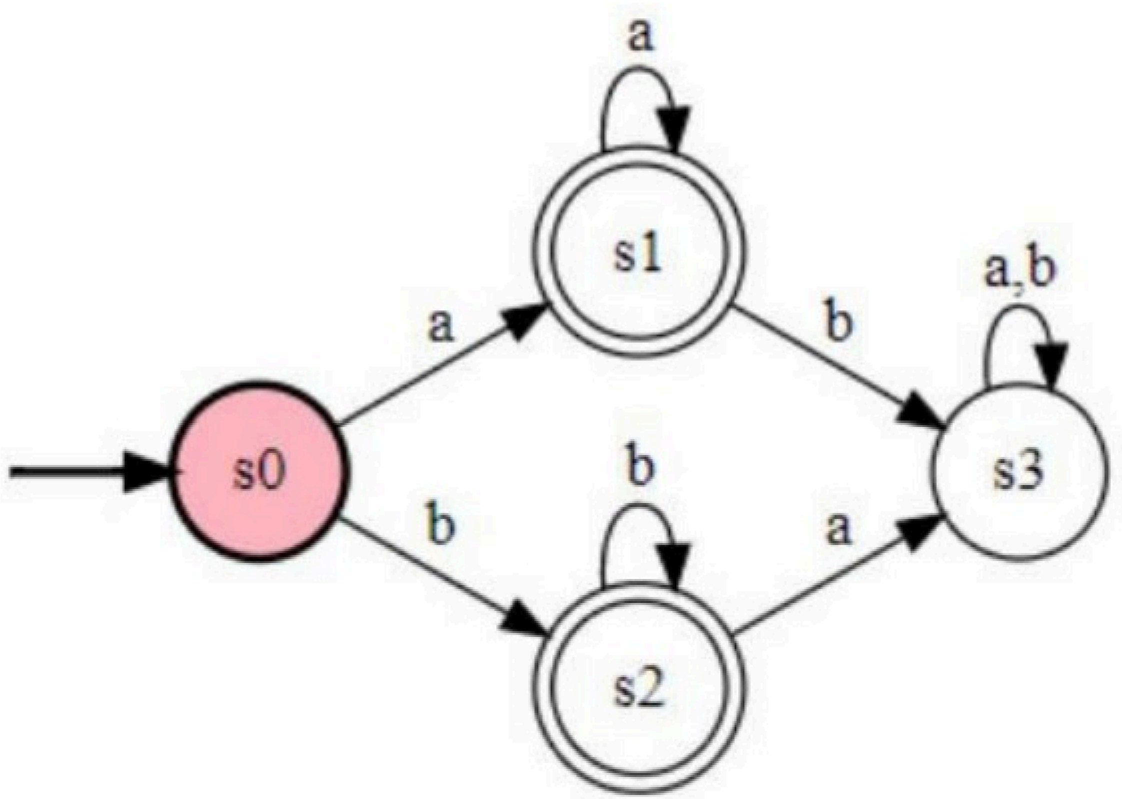
Consider the alphabet (set of symbols) $\{x, y, z\}$. How many three-letter words can be generated from that alphabet?

Problem

Consider the alphabet (set of symbols) $\{x, y, z\}$. If there must be at least two (2) consecutive x symbols in a word, how many three letter words with that property can be generated from that alphabet?

Setup

Consider the finite state machine shown below in answering the following questions.



You can assume that the alphabet for this machine is the set of symbols $\{a, b\}$. Note that a, b on an arrow from state X to state Y does **not** mean *an* `a` *symbol followed by a* `b` *symbol.* Instead, that represents two *separate* transitions when written out in text:

- `X:a>Y`
- `X:b>Y`

Problem
How many states are there in the machine referenced in the most recent **Setup** section? Of those, how many are *accepting*?

Problem
Given the machine referenced in the most recent **Setup** section, which of the following six words are *accepted*: $\{a, b, aa, ab, ba, bb\}$?

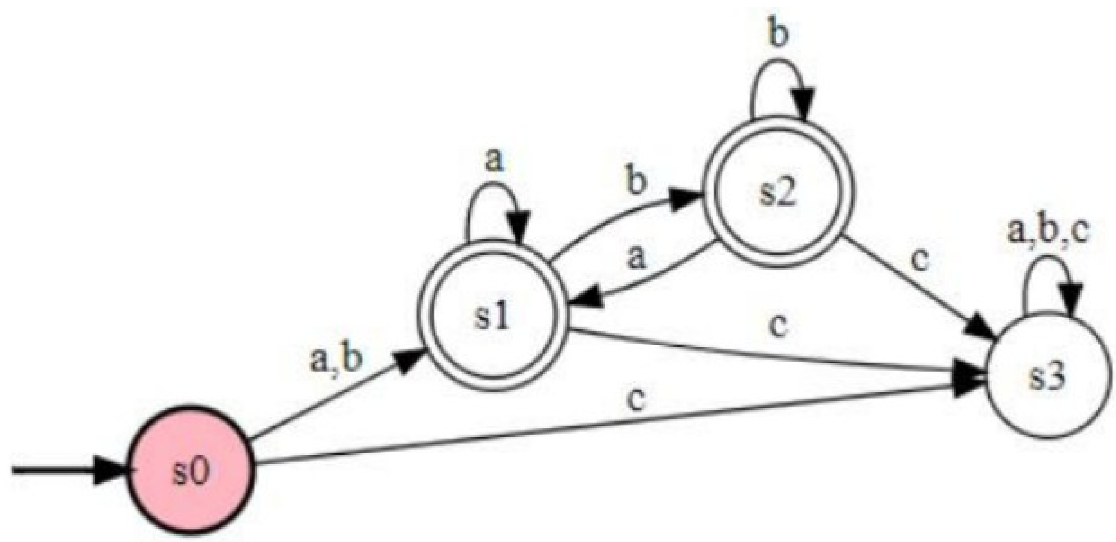
Problem
Given the machine referenced in the most recent **Setup** section, the trace `s0,s2,s2,s3,s3` is a trace of what word?

Problem
What is the structural property of the words that the machine in the most recent **Setup** section is checking for?

Problem
Which, if any, of the states in the machine in the most recent **Setup** section are *not on a path to an accepting state*? If they all are, write your answer as **NA**.

Problem
Is the language of words (the set of words) accepted by the machine in the most recent **Setup** section *finite* or *infinite* in size?

Setup
Consider the finite state machine shown below in answering the following questions.



You can assume that the alphabet for this machine is the set of symbols $\{a, b, c\}$.

Problem

Given the machine referenced in the most recent **Setup** section, how many words of length 2 are accepted?

Problem

True or *False* and Explain: Given the machine referenced in the most recent **Setup** section, all words that contain a `c` symbol are *rejected* by this machine.

Problem

True or *False* and Explain: Given the machine referenced in the most recent **Setup** section, only words that contain a `c` symbol are *rejected* by this machine.

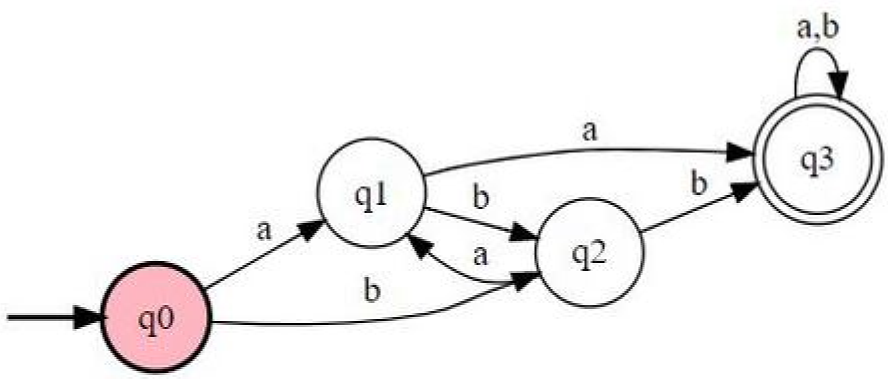
Problem

The machine referenced in the most recent **Setup** section has at least one state that is *not on a path to an accepting state*. The following text is the start of the *textual description* for the machine referenced in the most recent **Setup** section. The part that is left out is the data for the *transitions* section. Copy the text below and paste it in the large white textbox on the *Input automaton* tab of this website: https://ivanzuzak.info/noam/webapps/fsm_simulator/. Then add transitions in the appropriate format (`currentState:symbol>nextState`) to recreate the machine for all states *except* states that are not on a path to an accepting state. You can press the *Create automaton* button to visualize your machine. When you are comfortable with your answer, take a screenshot of the machine and then upload your screenshot as an image in the answer box below.

```
#states
s0
s1
s2
s3
#initial
s0
#accepting
s1
s2
#alphabet
a
b
c
#transitions
```

Use: Assess Yourself

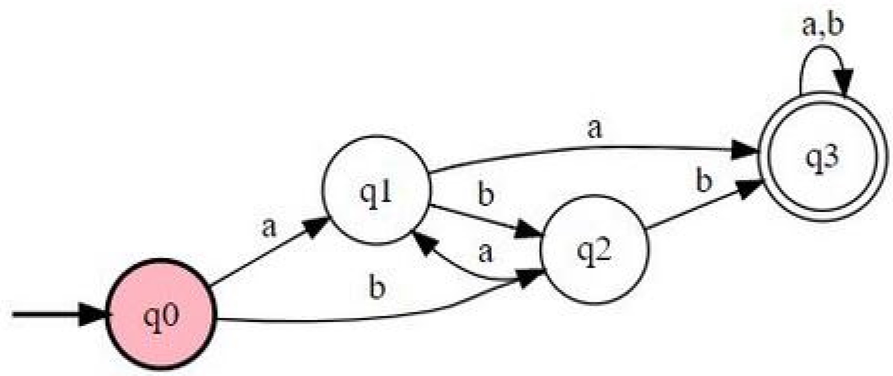
Problem
Consider the finite state machine shown in the image associated with this question. The associated alphabet is {*a*,*b*}, the machine is fully described (all transitions and states are shown), and it obeys the constraint of "for a given state, at most one transition on a given symbol". How many transitions are in the machine (how many *rules* are encoded)?



Choose the one best answer.

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- None of the above

Problem
For the same finite state machine, which of the following is/are NOT transitions in the machine? Check all that apply.

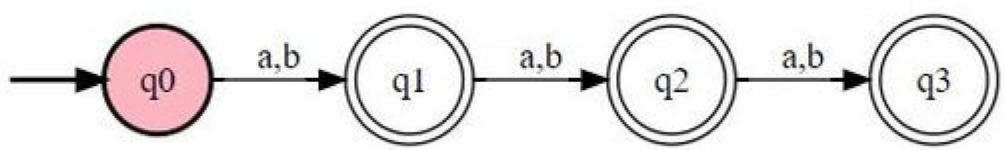


Choose all that apply.

- q1:a>q3
- q2:b>q1
- q2:a>q1
- q3:b>q3
- q3:a>q4

Problem

For this finite state machine, which of the following is/are the accepting states? You can assume the alphabet is {a,b}.

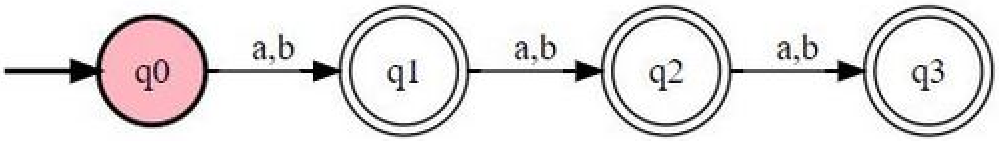


Choose all that apply.

- q0
- q1
- q2
- q3
- None of the above

Problem

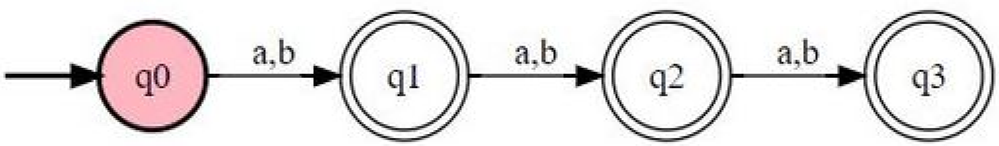
For the same finite state machine, the representation makes use of implicit transitions, not explicitly drawing the trash state or moves to the trash state. How many transitions are NOT present in the drawn image? You can assume that the alphabet is {a,b} and that it obeys the constraint of "for a given state, at most one transition on a given symbol".



Choose the one best answer.

- 1
- 2
- 4
- 6
- None of the above

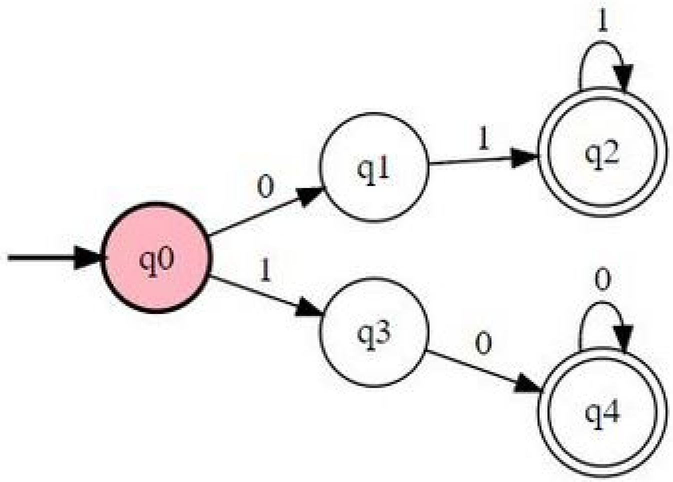
Problem
Given the same finite state machine, which of the following words are accepted?



Choose all that apply.

- a
- b
- aa
- aaa
- aba
- aaaa
- abab
- the empty string (with no symbols)

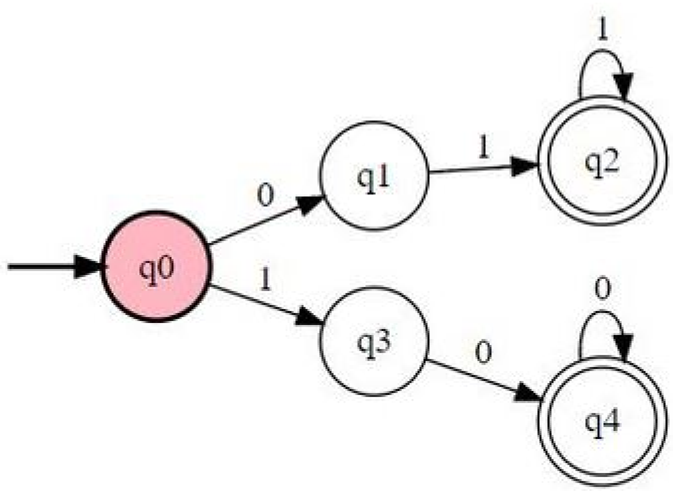
Problem
Consider the finite state machine shown in the image associated with this question. Which of the following words are REJECTED? Note that the machine is not explicitly drawing the trash state and moves to the trash state. You can assume the alphabet is {0,1} and that it obeys the constraint of "for a given state, at most one transition on a given symbol". Note I am asking about REJECTED words for this machine.



Choose all that apply.

- 0
- 1
- 01
- 10
- 010
- 011
- 101
- 111

Problem
Given the same finite state machine, which of these accurately represents the trace of the accepted word 100?



Choose the one best answer.

- q0,q1
- q0,q1,q2
- q0,q1,q2,q2
- q0,q3,q4,q4
- q0,q1,q3,q4,q4
- q3,q4

Build: Preparation

Problem

When designing a finite state machine, if one wants to accept one specific word of length k, how many states should be required?

- k
- $k - 1$
- $k + 1$
- $2 * k$
- None of the above

Problem

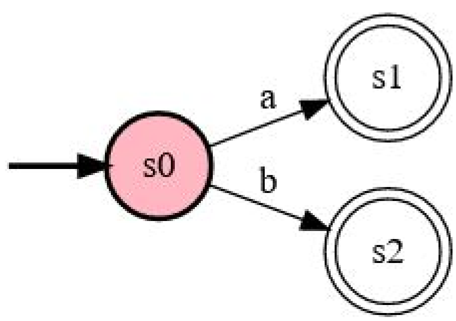
When designing a finite state machine, what feature of a finite state machine allows for the notion of a symbol or series of symbols being repeated?

- Linear ordering of transitions
- Looping
- Branching

Problem

The image below is most representative of which of the following aspects of designing a finite state machine?

- Linear ordering of transitions
- Looping
- Branching



Build: Warmup

Setup

In designing finite state machines to recognize the languages described below:

- Build a machine that obeys the constraint: *For any given state, at most one transition on (for) any given input symbol*
  - That is, no state should have two or more arrows out from it labeled with the same symbol
- It is OK to leave off (not draw) any states that are *not on a path to an accepting state*, as well as any transitions into those states.

Problem

Design a deterministic finite state machine over the alphabet $\{0, 1\}$ that accepts the words: `1`, `10`, `100`, `1000`, `10000`, … (a `1` symbol followed by zero or more `0` symbols).

Problem

Design a deterministic finite state machine over the alphabet $\{0, 1\}$ that accepts the words: `1`, `10`, `100`, `1000`, `10000`, … and the words `11`, `111`, `1111`, … (a `1` symbol followed by zero or more of just `0` symbols or followed by zero or more of just `1` symbols).

Problem

Design a deterministic finite state machine over the alphabet $\{a, b\}$ that accepts words of one or more `b` symbols followed by one or more `a` symbols. The number of `a` and `b` symbols do not have to be equal - there just has to be at least one of each AND all the `b` symbols have to appear before all of the `a` symbols.

Examples of words that should be *accepted* are:

- `ba`, `bba`, `baa`, `bbbaa`

Examples of words that should be *rejected* are:

- `a`, `b`, `ab`, `bab`, `aba`

Build: Advanced

Setup

For the *digits* alphabet {0, 1, 2, 3,4, 5, 6, 7, 8, 9}, the language of interest is *hours* in *military time*, which are all the two-digit values between and including 00 and 23. Ignore minutes and just focus on the language of *two-digit* hours.

Problem

Only using the defined alphabet, write down three words that should be *accepted* by this finite state machine and, separately, three words that should be *rejected* by this finite state machine. Try to make your rejected words be rejected for various reasons (though remember to only use the defined alphabet).

Problem

Design a *deterministic* finite state machine for this language on paper or on the computer, and then, using this website - https://ivanzuzak.info/noam/webapps/fsm_simulator/, encode your designed machine on the *Input automaton* tab so it can be visualized and words can be traced.

Problem

Using the website, trace your six words from two problems back through the machine designed in the immediately previous problem. Indicate for each word the following:

- The word
- The *trace* of the word
- Whether the word is actually *accepted* or *rejected* by the designed machine

Problem

Critique the designed machine:

- Is it correct (as far as you can tell with your testing words)?
- Are there states that are not on the path to an accepting state and could be left out of the drawing?
- Do there appear to be any ways to *simplify* the machine, such as removing states that look "redundant"?

Setup

For the alphabet `<`, `!`, `=`, the language of interest is the following comparison operators: `<`, `<=`, `!=`, `==`.

Problem

Only using the defined alphabet, write down three words that should be *accepted* by this finite state machine and, separately, three words that should be *rejected* by this finite state machine. Try to make your rejected words be rejected for various reasons (though remember to only use the defined alphabet).

Problem

Design a *deterministic* finite state machine for this language on paper or on the computer, and then, using this website - https://ivanzuzak.info/noam/webapps/fsm_simulator/, encode your designed machine on the *Input automaton* tab so it can be visualized and words can be traced.

Problem

Using the website, trace your six words from two problems back through the machine designed in the immediately previous problem. Indicate for each word the following:

- The word
- The *trace* of the word
- Whether the word is actually *accepted* or *rejected* by the designed machine

Problem

Critique the designed machine:

- Is it correct (as far as you can tell with your testing words)?
- Are there states that are not on the path to an accepting state and could be left out of the drawing?
- Do there appear to be any ways to *simplify* the machine, such as removing states that look "redundant"?

Setup

For the alphabet `0`, `1`, the language of interest is the words that have an *odd length* and *odd count of `1` symbols*. When referring to *lengths* and *counts of `1` symbols*, *odd* is 1, 3, 5, 7, … and *even* is 0, 2, 4, 6, … .

Problem

Only using the defined alphabet, write down three words that should be *accepted* by this finite state machine and, separately, three words that should be *rejected* by this finite state machine. Try to make your rejected words be rejected for various reasons (though remember to only use the defined alphabet).

Problem

Design a *deterministic* finite state machine for this language on paper or on the computer, and then, using this website - https://ivanzuzak.info/noam/webapps/fsm_simulator/, encode your designed machine on the *Input automaton* tab so it can be visualized and words can be traced.

Problem

Using the website, trace your six words from two problems back through the machine designed in the immediately previous problem. Indicate for each word the following:

- The word
- The *trace* of the word
- Whether the word is actually *accepted* or *rejected* by the designed machine

Problem

Critique the designed machine:

- Is it correct (as far as you can tell with your testing words)?
- Are there states that are not on the path to an accepting state and could be left out of the drawing?
- Do there appear to be any ways to *simplify* the machine, such as removing states that look "redundant"?

Build: Assess Yourself

No Assess Yourself problems have been created for this content.