29 Deterministic Finite Automaton

19 points

Introduction

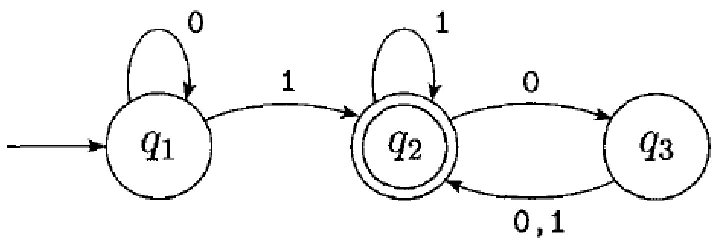
A Deterministic Finite Automaton (DFA) is a simple computational model that accepts or rejects a given string of symbols by running through a state sequence uniquely determined by the string.

A DFA can be formally defined as a 5-tuple **(Q, Σ, δ, q_0, F)**, where,

1. **Q** is the finite set of available states,
2. **Σ** is the finite set of accepted symbols, also called the alphabet,
3. **δ: Q X Σ → Q** is the transition function,
4. **q_0 ∈ Q** is the start state, and
5. **F ⊆ Q** is the set of accepted states.

As said, DFAs are capable of processing strings whose characters are part of the DFA's alphabet. Starting in state **q_0**, the automaton takes one symbol of the input string at a time and performs the corresponding state transition defined by its **δ**(q, c) (being **q ∈ Q** the current state and **c ∈ Σ** the symbol being processed). The process finishes when the last symbol has been processed. At this point, we say that the DFA accepts the input string if it is in a state **q ∈ F**. Otherwise, the input is rejected.

As an example, in the following figure it is represented a three-state DFA that we call **M**.



Formally described:

M = (Q, Σ, δ, q_0, F), where

1. **Q = {q_1, q_2, q_3}**,

2. $\Sigma = \{0, 1\}$,

3. δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

4. q_1 is the start state, and

5. $F = \{q_2\}$

Notice that **M** will accept strings such as **1**, **1111**, **0010010000** or **0101000111001**. Contrary, strings like **0**, **0000**, **111000** or **1010101010** aren't accepted. Indeed, **M** recognizes the language of strings that contain at least one 1 and an even number of 0s follow the last 1.

Let's try to build a program that allows us to determine if a DFA formally defined would accept or not a given string.

Input

The input will consist in several rows:

- 1st will come the set of states of the DFA.

- 2nd the alphabet supported by the DFA.

- 3rd the start state.

- 4th the set of accepted states (note that there could exist more than one accepted state).

- N rows, one per state and in order of appearance, defining the transition function for such state using pairs of symbol-destination.

- Finally, the input string.

In sake of clarity, the input for trying to discern if **M** (our previous example DFA) would accept 100010 would be:

q1 q2 q3

0 1

q1

q2

0 q1 1 q2

0 q3 1 q2

0 q2 1 q2

100010

Output

The output will be a single sentence depending on the evaluation's outcome:

- If the input string has symbols that aren't from the DFA's alphabet: "Invalid input string!"

- If the input string is accepted: "Input string accepted"

- If the input string is rejected: "Input string rejected"

- If the definition does not correspond with a DFA: "This is not a deterministic finite automaton!"

Note that the later situation may arise when:

- A transition involving a symbol not existing in the DFA's alphabet is defined.

- A transition involving a state not existing in the set of states of the DFA is defined.

- There isn't exactly a single transition defined for each pair of state-symbol. If that rule is broken we would be in front of a Nondeterministic Finite Automaton (NFA), so we better leave that topic for another year :)

Example 1

Input

```
q1 q2 q3 q4
0 1
q1
q4
0 q2 1 q1
0 q3 1 q1
```

```
0 q3 1 q4
0 q4 1 q4
011001100
```

Output

```
Input string accepted
```

Example 2

Input

```
s q1 q2 r1 r2
a b
s
q1 r1
a q1 b r1
a q1 b q2
a q1 b q2
a r2 b r1
a r2 b r1
baba
```

Output

```
Input string rejected
```

Example 3

Input

```
q1 q2
a b c
q1
q1
a q2 b q1 c q2
a q1 b q1
abc
```

Output

```
This is not a deterministic finite automaton!
```