Chapter 4. Computation

4.2 Finite state machines
4.3 Turing machines

4.2 Finite state machines

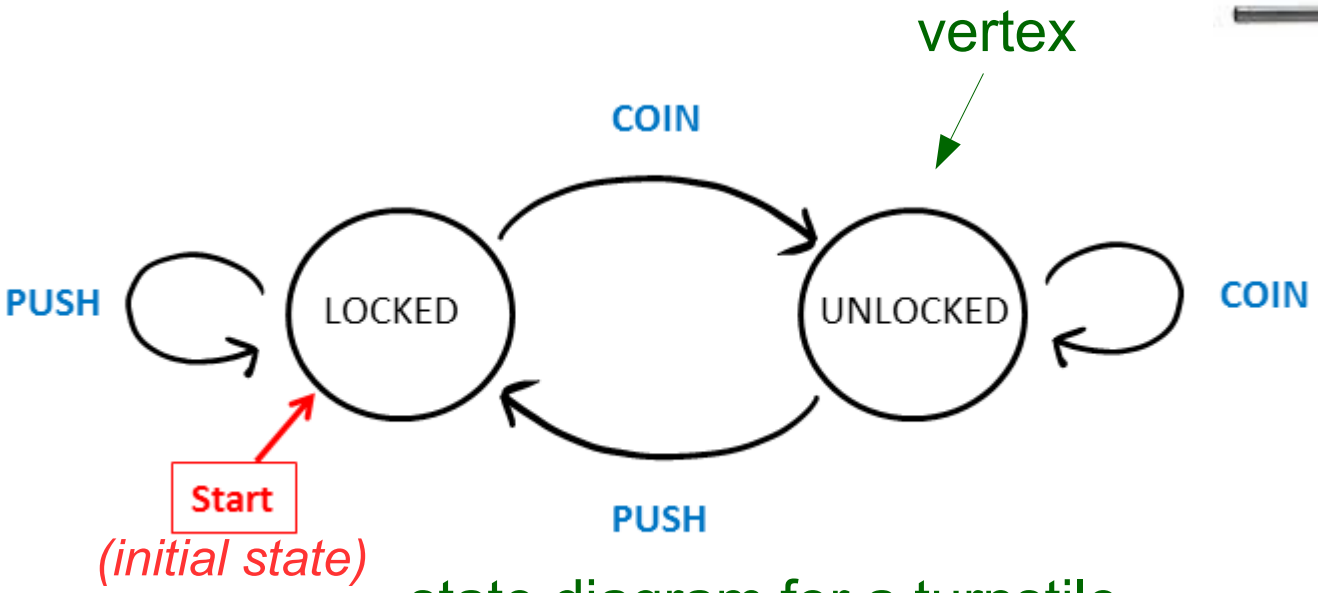
A finite-state machine (FSM) is a mathematical model of computation used to design both *computer programs* and *sequential logic circuits*.

Note: search for *sequential logic circuits* online.
For example check out here:
https://en.wikipedia.org/wiki/Sequential_logic

Other names:
finite-state automaton (FSA, plural: automata)
state machine

FSM consists of:
- a finite set of *states* with
- *transitions* between the states triggered by different
- *input* actions

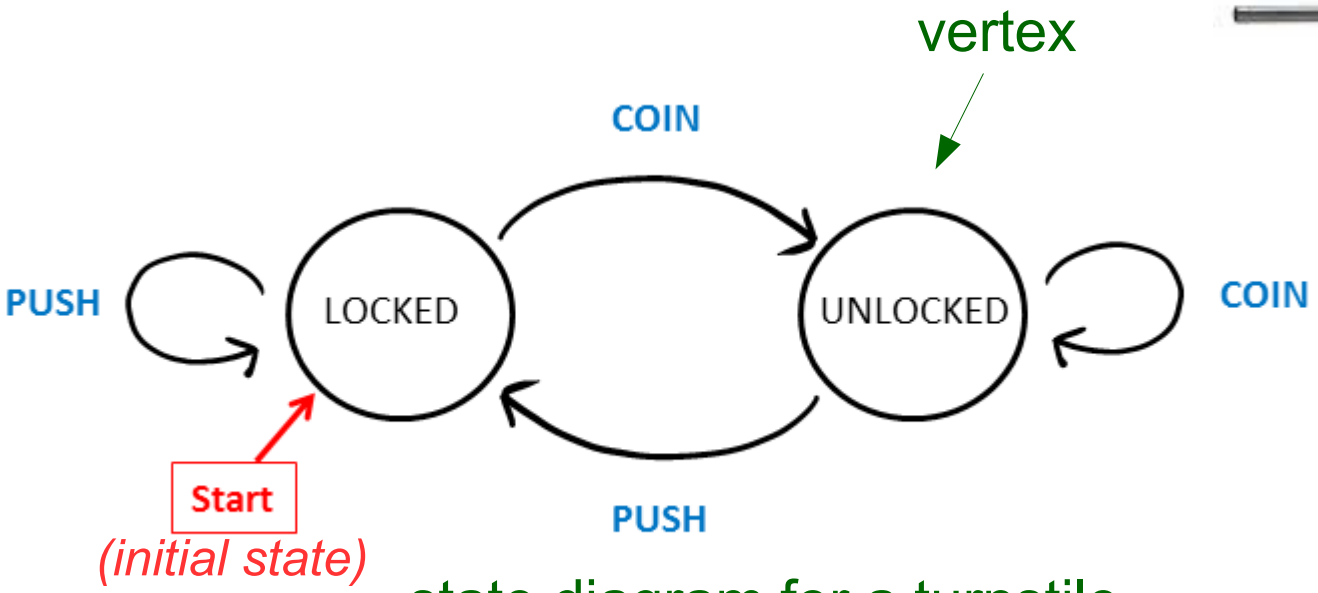
Example: turnstile



state diagram for a turnstile

4.2 Finite state machines

Example: turnstile



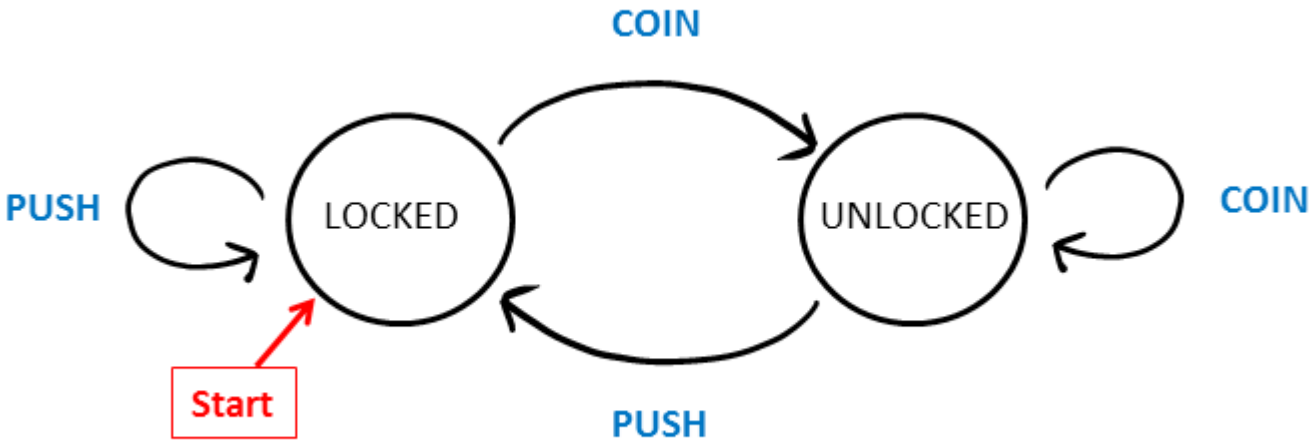
state diagram for a turnstile

Set of states S = {LOCKED,UNLOCKED}
Input set I = {COIN, PUSH} (input alphabet)
Transition function δ:

δ(LOCKED,PUSH) = LOCKED
δ(LOCKED,COIN) = UNLOCKED
δ(UNLOCKED,COIN) = UNLOCKED
δ(UNLOCKED,PUSH) = LOCKED

good practice:
give states descriptive names.

Example: turnstile

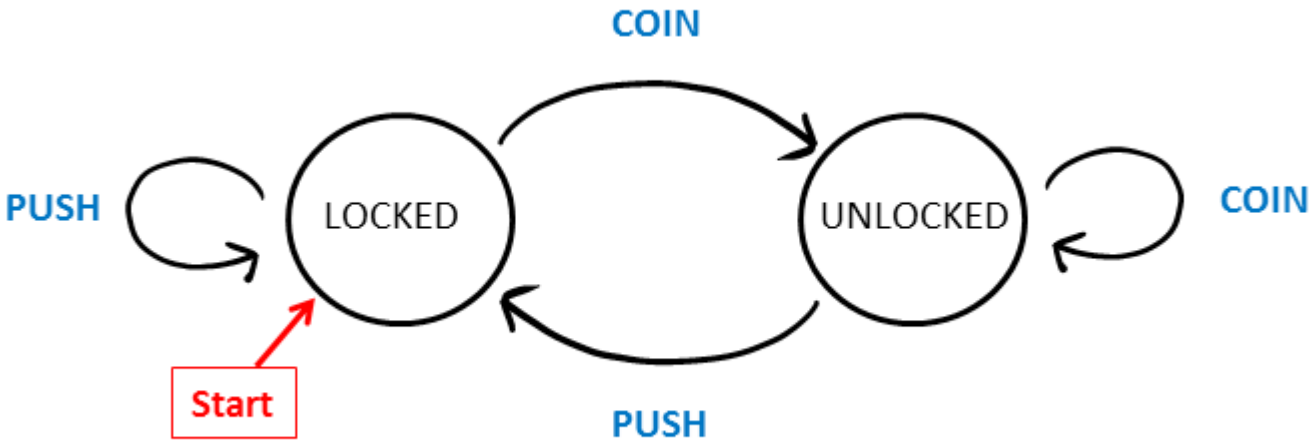


state diagram for a turnstile

Set of states S = {LOCKED,UNLOCKED}
Input set I = {COIN, PUSH}
State transition table for the turnstile finite state machine:

state/input	COIN	PUSH
LOCKED	UNLOCKED	LOCKED
UNLOCKED	UNLOCKED	LOCKED

Example: turnstile

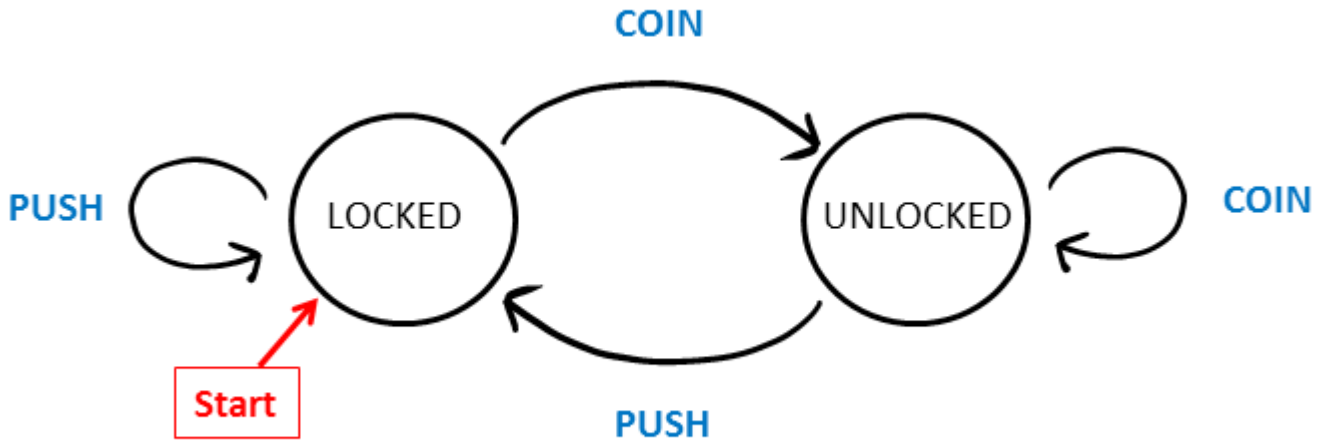


state diagram for a turnstile

The input to the FSM is a sequence of actions from the input set
A possible input: PUSH COIN COIN PUSH
The resulting state is: LOCKED

4.2 Finite state machines

Example: turnstile



state diagram for a turnstile

Set of states S = {LOCKED,UNLOCKED}
Input set I = {COIN, PUSH}
Transition function δ:

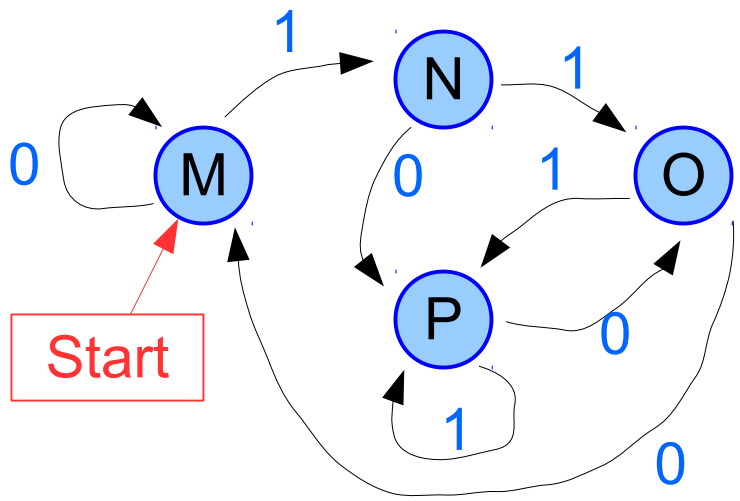
δ(LOCKED,PUSH) = LOCKED
δ(LOCKED,COIN) = UNLOCKED
δ(UNLOCKED,COIN) = UNLOCKED
δ(UNLOCKED,PUSH) = LOCKED

Components of a FSM:

Notation	Description
S	Finite set of states
$s_0 \in S$	s_0 is the start state
I	Finite set of input actions (alphabet)
$\delta: S \times I \rightarrow S$	Transition function

4.2 Finite state machines - practice

Consider the FSM defined in the state diagram below.
The set of states S = {M, N, O, P} and the set of input actions I = {0,1}.



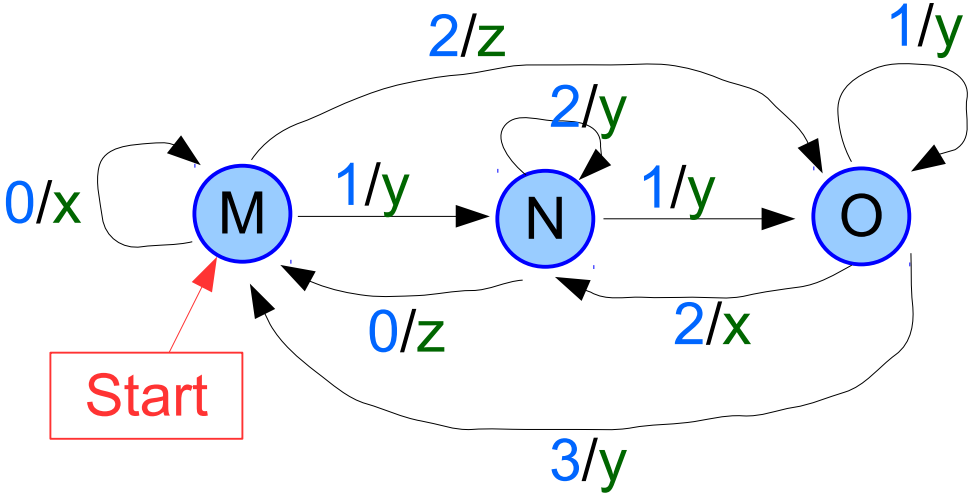
1. What is the current state after the FSM has processed the input sequence 01011?

2. What is the current state after the FSM has processed the input sequence 1111?

3. What input sequence required to get from state M to state M and changing to at least one other state?

4.2 Finite state machines with output

FSM with output produces a response that depends on the current state as well as the most recently received input



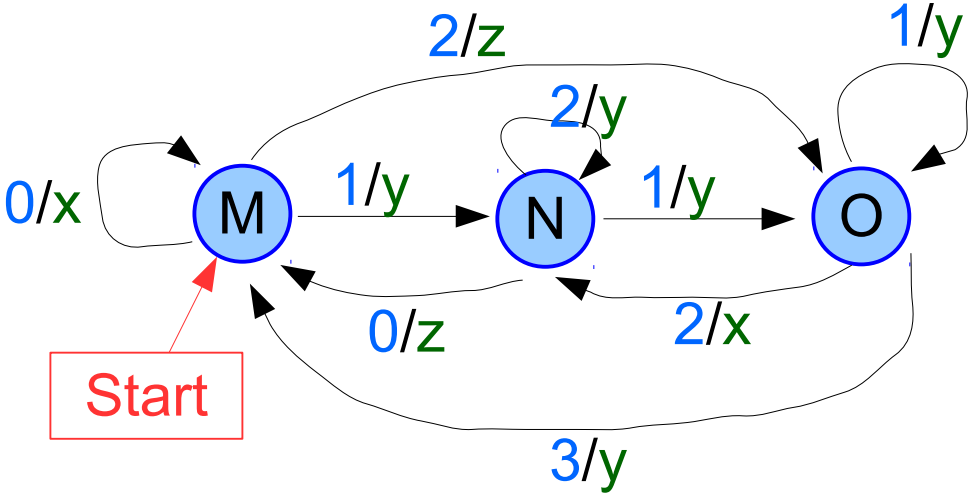
state diagram

Notation	Description
S = {M,N,O}	Finite set of states
I = {0,1,2}	Finite set of input actions (input alphabet)
O = {x,y,z}	Finite set of output responses

Let's find the output corresponding to the input string 0 1 2 1 1 2

4.2 Finite state machines with output

FSM with output produces a response that depends on the current state as well as the most recently received input



state diagram

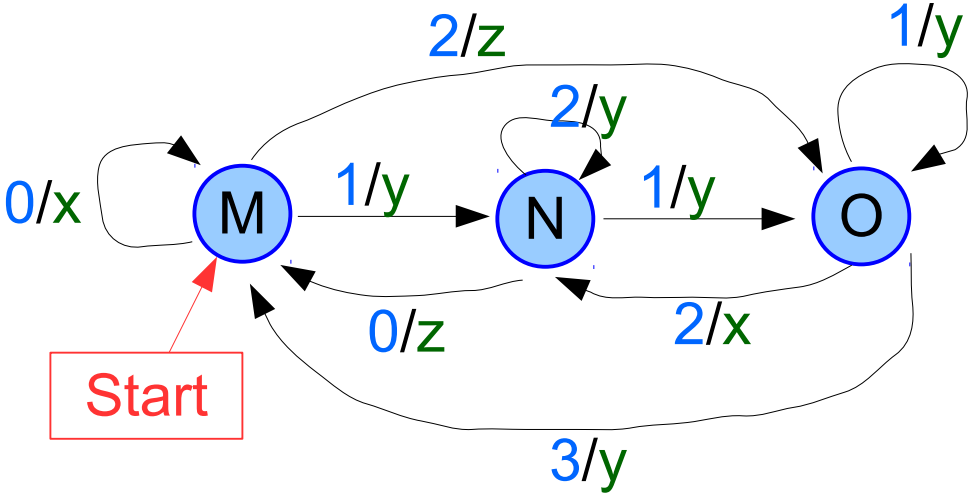
Notation	Description
S = {M,N,O}	Finite set of states
I = {0,1,2}	Finite set of input actions (input alphabet)
O = {x,y,z}	Finite set of output responses

Let's find the output corresponding to the input string 0 1 2 1 1 2

x y y y y x

4.2 Finite state machines with output

FSM with output produces a response that depends on the current state as well as the most recently received input



state diagram

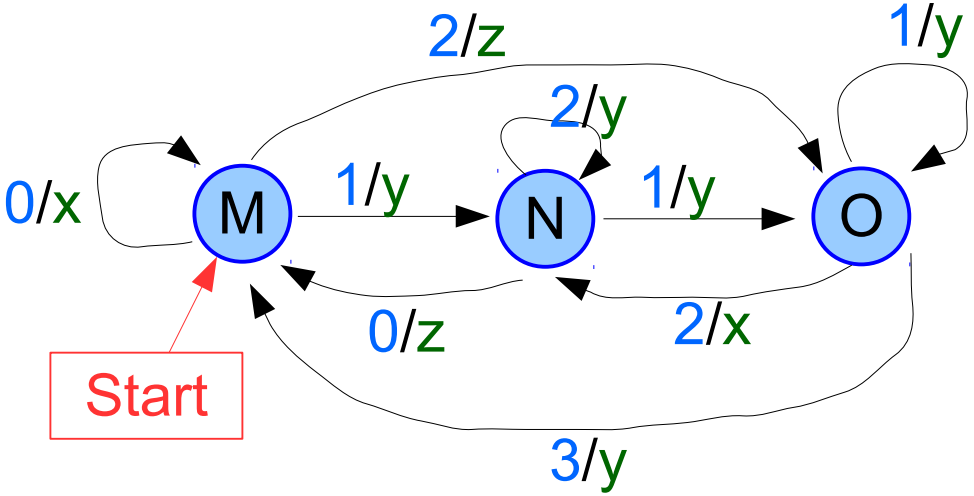
Notation	Description
S = {M,N,O}	Finite set of states
I = {0,1,2}	Finite set of input actions (input alphabet)
O = {x,y,z}	Finite set of output responses

Let's find the output corresponding to the input string 0 1 2 1 1 2
x y y y y x

Now let's find the output corresponding to the input string 2 1 3 1 2

4.2 Finite state machines with output

FSM with output produces a response that depends on the current state as well as the most recently received input



state diagram

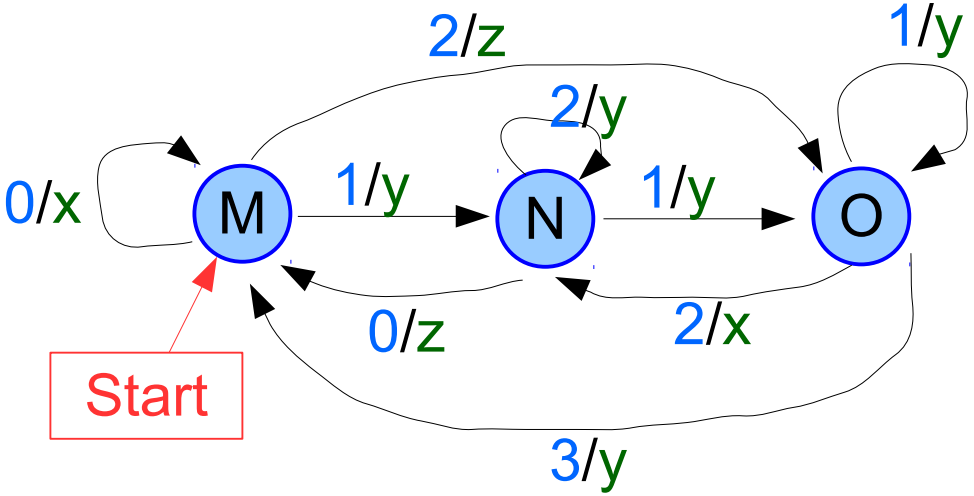
Notation	Description
S = {M,N,O}	Finite set of states
I = {0,1,2}	Finite set of input actions (input alphabet)
O = {x,y,z}	Finite set of output responses

Let's find the output corresponding to the input string 0 1 2 1 1 2
x y y y y x

Now let's find the output corresponding to the input string 2 1 3 1 2
z y y y y

4.2 Finite state machines with output

FSM with output produces a response that depends on the current state as well as the most recently received input



state diagram

Notation	Description
S = {M,N,O}	Finite set of states
I = {0,1,2}	Finite set of input actions (input alphabet)
O = {x,y,z}	Finite set of output responses

Let's find the output corresponding to the input string 0 1 2 1 1 2

x y y y y x

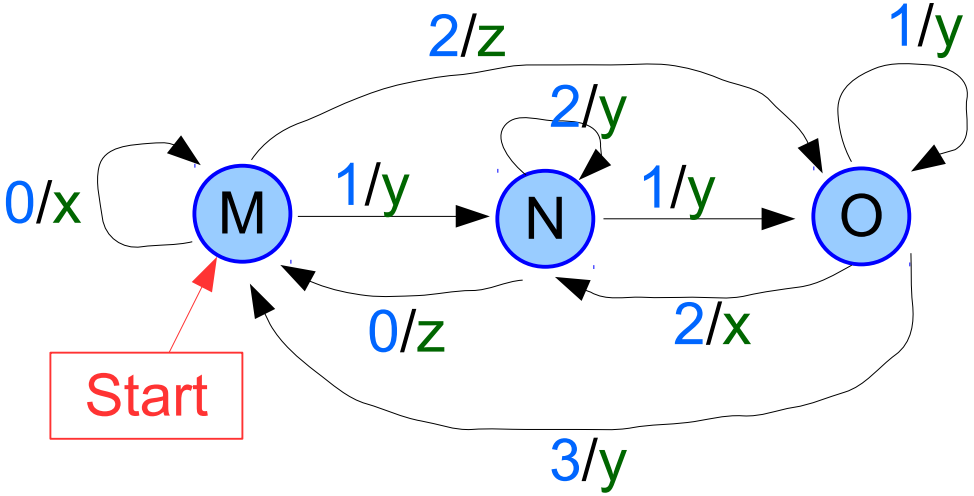
Now let's find the output corresponding to the input string 2 1 3 1 2

z y y y y

What state did we end up at?

4.2 Finite state machines with output

FSM with output produces a response that depends on the current state as well as the most recently received input



state diagram

Notation	Description
S = {M,N,O}	Finite set of states
I = {0,1,2}	Finite set of input actions (input alphabet)
O = {x,y,z}	Finite set of output responses

Let's find the output corresponding to the input string 0 1 2 1 1 2

x y y y y x

Now let's find the output corresponding to the input string 2 1 3 1 2

z y y y y

What state did we end up at?

N

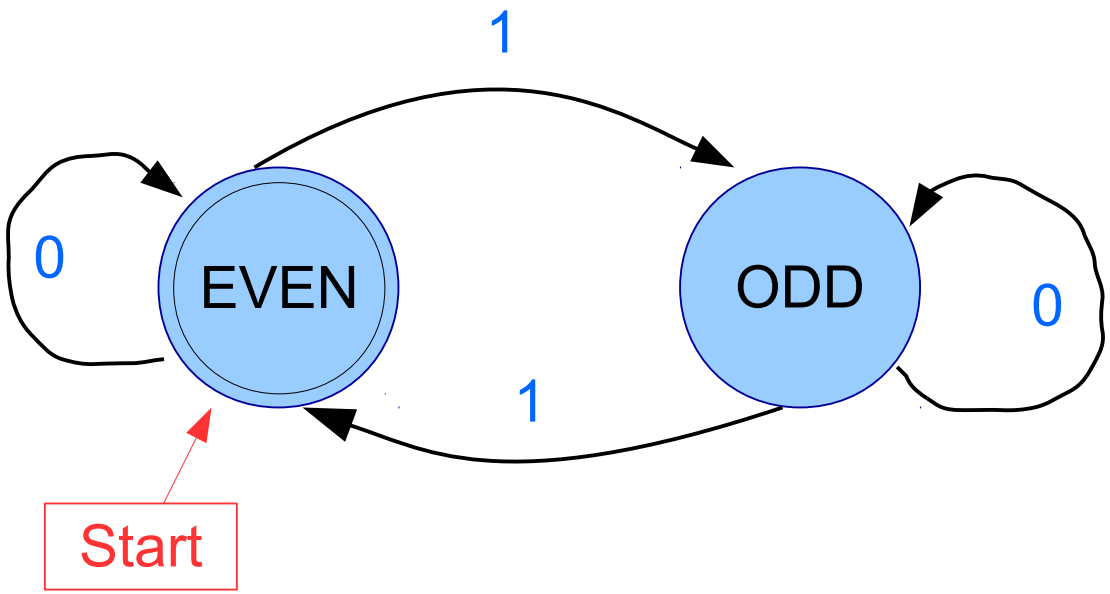
4.2 Finite state machines that recognize properties

In addition to what we already have for FSM so far, let's consider set A of *accepting states*.

$A \subseteq S$

An input string is accepted if the FSM ends up in an accepting state after each character in the string is processed.

Example: Let's process a bit string and if the number of 1's is even or no 1's are in the string, the finite state machine will accept it.



1) What happens on the input 1001101?

2) What happens on the input 01011011?

4.2 Finite state machines with output

Finite-state machines can model a large number of problems, among which are electronic design automation, communication protocol design, language parsing and other engineering applications. In biology and artificial intelligence research, state machines or hierarchies of state machines have been used to describe neurological systems. In linguistics, they are used to describe simple parts of the grammars of natural languages.

Considered as an abstract model of computation, the finite state machine has less computational power than some other models of computation such as the Turing machine. There are tasks that no FSM can do, but some Turing machines can. This is because an FSM's memory is limited by the number of states it has.
Example: it is impossible to design an FSM that takes as input any binary string and accepts if and only if the majority of the bits are 1

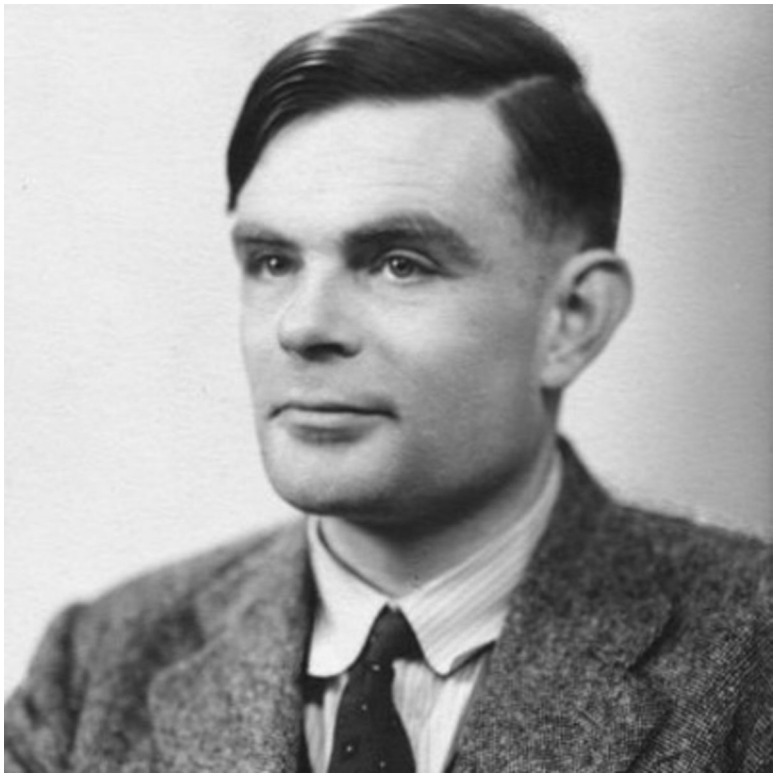
FSMs are studied in the more general field of automata theory.

4.3 Turing machines

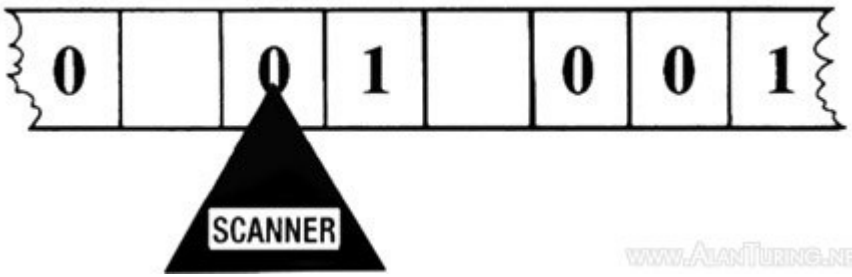
Mathematician Alan Turing developed a model of a computer, called a Turing machine, in order to reason about general purpose computers.

It is an abstract machine that manipulates symbols on a strip of tape according to a table of rules; to be more exact, it is a mathematical model of computation that defines such a device.

Despite the model's simplicity, given any computer algorithm, a Turing machine can be constructed that is capable of simulating that algorithm's logic.

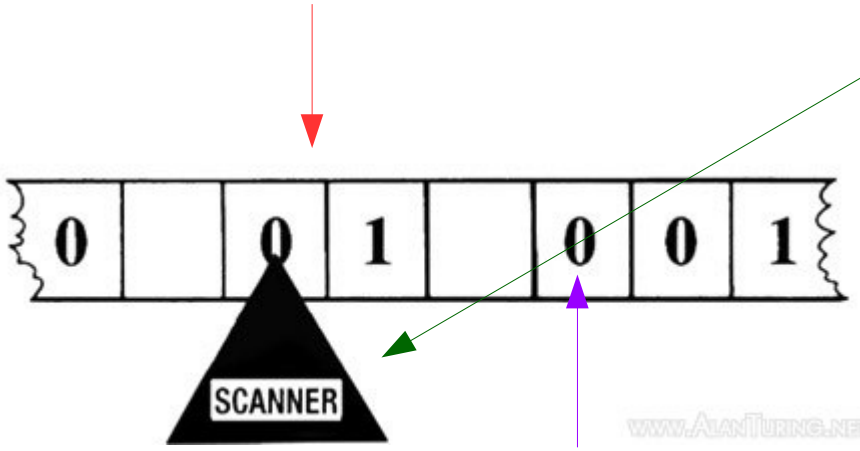


Alan Mathison Turing
23 June 1912 – 7 June 1954



4.3 Turing machines

Memory: a one-dimensional tape divided into individual cells



head can read and write symbols on the tape and move the tape/along the tape left and right one only one cell at a time

each cell can hold one symbol from a finite tape alphabet denoted by Γ

In general the tape is infinite in both directions, but without any limitation we can assume that it is infinite to the right, and has a leftmost cell.

The tape alphabet Γ must contain a special symbol called the *blank* symbol (represented by a * symbol) and at least one other symbol.

4.3 Turing machines

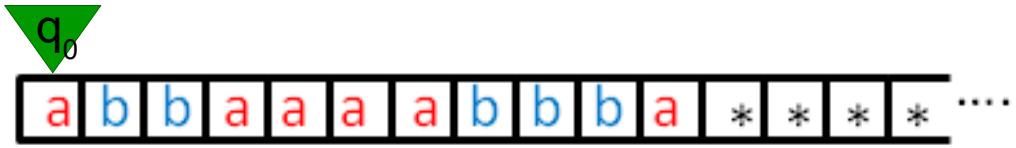
Turing machine has:

- tape alphabet Γ , blank symbol (*) $\in \Gamma$
- finite set of states S
- q_0 (start state) $\in$ S, q_{acc} (accept state) $\in$ S, q_{rej} (reject state) $\in$ S
- head pointing to a cell (current symbol)
- transition function δ

δ(current state,current symbol) =

(state, tape symbol, direction for the head to move)

- direction for the tape head to move: L,R
- δ is defined for every (state, tape symbol) pair, except for the states q_{acc} and q_{rej}. If the Turing machine ever reaches one of these two states, it stops.



- input alphabet Σ
- $\Sigma \subset \Gamma$, blank symbol (*) $\notin \Sigma$
- input is as a string over the input alphabet Σ
- Turing machine starts out an execution in state q_0 with the head in the leftmost position
- The input to the problem to be solved is written on the tape as a string of input symbols starting in the leftmost cell. The rest of the tape is all blank symbols that extend infinitely to the right.

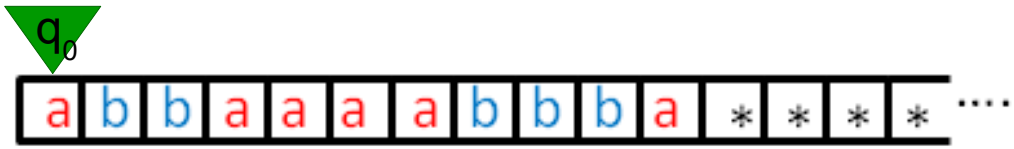
4.3 Turing machines

The Turing machine proceeds according to the rules designated by the transition function.

If the Turing machine reaches the *accept state* after starting with a particular input string x, then the Turing machine accepts x.

If the Turing machine reaches the *reject state* after starting with a particular input string x, then the Turing machine rejects x.

If a Turing machine reaches either the accept or reject state on input x, we say that the Turing machine halts on input x.
As with other computer programs, the Turing machine may never halt on a particular input in which case the Turing machine neither accepts or rejects the input.



4.3 Turing machines

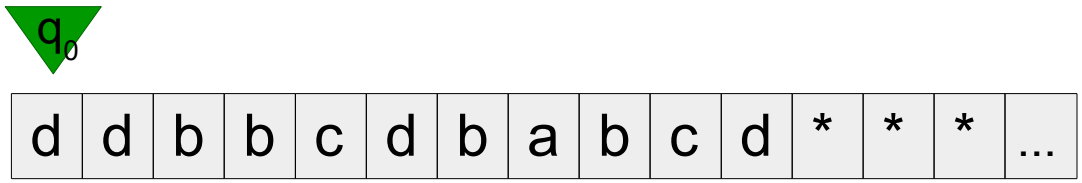
Example: the following Turing machine looks for the first occurrence of letter a in the input string. If it is found, the input is accepted, otherwise it is rejected.

Γ = {a,b,c,d,*}

input = ddbbcdbabcd

S = $\{q_0, q_{acc}, q_{rej}\}$

	a	b	c	d	*
q_0	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	(q_0,d,R)	(q_{rej},b,R)



4.3 Turing machines

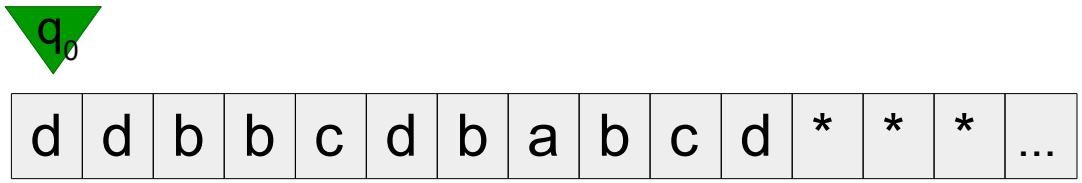
Example: the following Turing machine looks for the first occurrence of letter a in the input string. If it is found, the input is accepted, otherwise it is rejected.

Γ = {a,b,c,d,*}

input = ddbbcdbabcd

S = $\{q_0, q_{acc}, q_{rej}\}$

	a	b	c	d	*
q_0	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	(q_0,d,R)	(q_{rej},b,R)



4.3 Turing machines

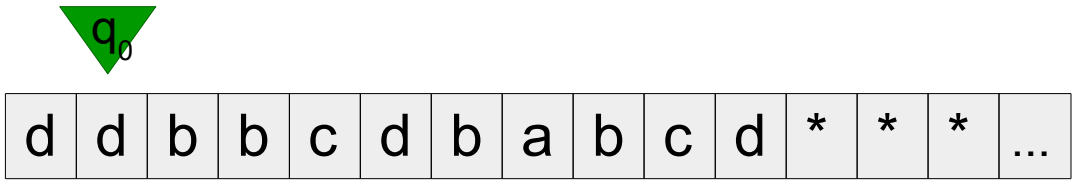
Example: the following Turing machine looks for the first occurrence of letter a in the input string. If it is found, the input is accepted, otherwise it is rejected.

Γ = {a,b,c,d,*}

input = ddbbcdbabcd

S = $\{q_0, q_{acc}, q_{rej}\}$

	a	b	c	d	*
q_0	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	(q_0,d,R)	(q_{rej},b,R)



4.3 Turing machines

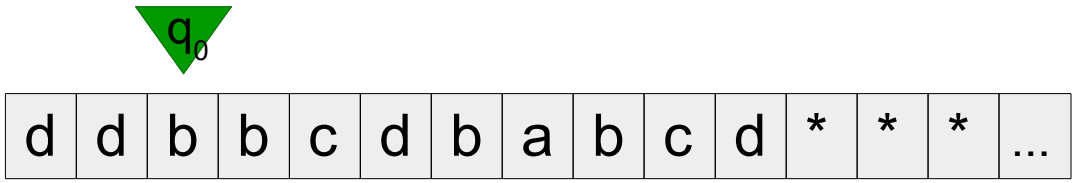
Example: the following Turing machine looks for the first occurrence of letter a in the input string. If it is found, the input is accepted, otherwise it is rejected.

Γ = {a,b,c,d,*}

input = ddbbcdbabcd

S = $\{q_0, q_{acc}, q_{rej}\}$

	a	b	c	d	*
q_0	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	(q_0,d,R)	(q_{rej},b,R)



4.3 Turing machines

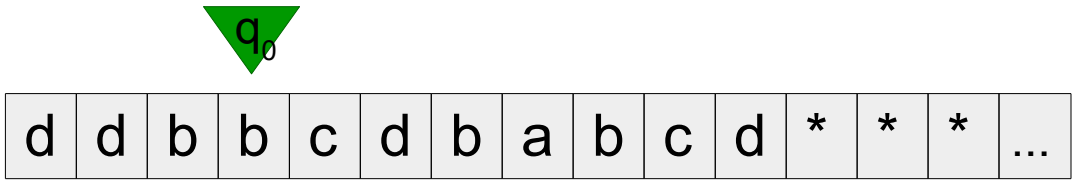
Example: the following Turing machine looks for the first occurrence of letter a in the input string. If it is found, the input is accepted, otherwise it is rejected.

Γ = {a,b,c,d,*}

input = ddbbcdbabcd

S = $\{q_0, q_{acc}, q_{rej}\}$

	a	b	c	d	*
q_0	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	(q_0,d,R)	(q_{rej},b,R)



4.3 Turing machines

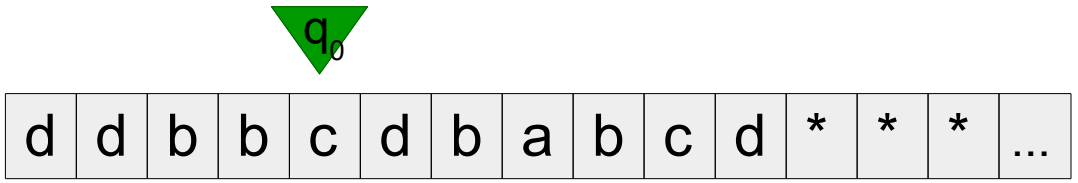
Example: the following Turing machine looks for the first occurrence of letter a in the input string. If it is found, the input is accepted, otherwise it is rejected.

Γ = {a,b,c,d,*}

input = ddbbcdbabcd

S = $\{q_0, q_{acc}, q_{rej}\}$

	a	b	c	d	*
q_0	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	(q_0,d,R)	(q_{rej},b,R)



4.3 Turing machines

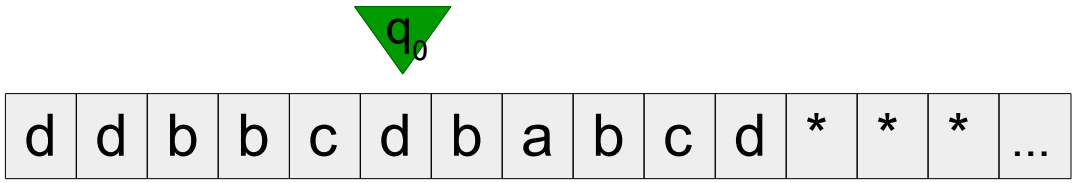
Example: the following Turing machine looks for the first occurrence of letter a in the input string. If it is found, the input is accepted, otherwise it is rejected.

Γ = {a,b,c,d,*}

input = ddbbcdbabcd

S = $\{q_0, q_{acc}, q_{rej}\}$

	a	b	c	d	*
q_0	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	(q_0,d,R)	(q_{rej},b,R)



4.3 Turing machines

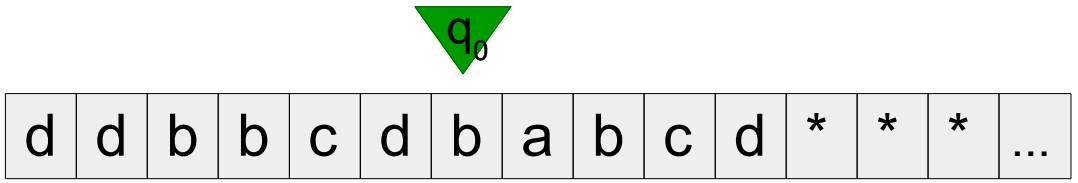
Example: the following Turing machine looks for the first occurrence of letter a in the input string. If it is found, the input is accepted, otherwise it is rejected.

Γ = {a,b,c,d,*}

input = ddbbcdbabcd

S = $\{q_0, q_{acc}, q_{rej}\}$

	a	b	c	d	*
q_0	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	(q_0,d,R)	(q_{rej},b,R)



4.3 Turing machines

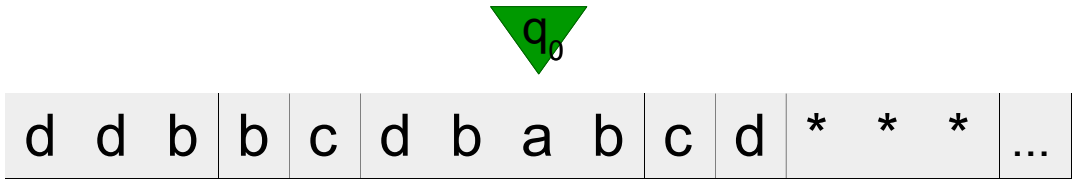
Example: the following Turing machine looks for the first occurrence of letter a in the input string. If it is found, the input is accepted, otherwise it is rejected.

Γ = {a,b,c,d,*}

input = ddbbcdbabcd

S = $\{q_0, q_{acc}, q_{rej}\}$

	a	b	c	d	*
q_0	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	(q_0,d,R)	(q_{rej},b,R)



4.3 Turing machines

Example: the following Turing machine looks for the three consequtive occurrences of letter **a** in the input string. If it is found, the input is accepted, otherwise it is rejected.

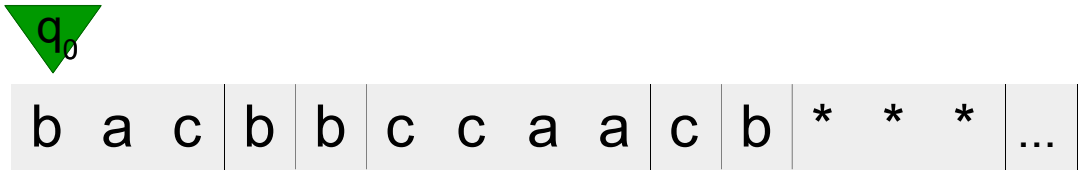
$\Gamma = \{a,b,c,*\}$

input = bacbbccaacb

$S = \{q_0, q_{acc}, q_{rej}\}$

Transition function:

	a	b	c	*
q_0	(q_1,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_1	(q_2,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_2	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$



4.3 Turing machines

Example: the following Turing machine looks for the three consequtive occurrences of letter **a** in the input string. If it is found, the input is accepted, otherwise it is rejected.

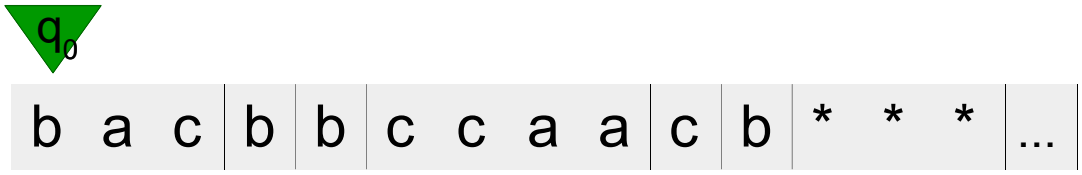
$\Gamma = \{a,b,c,*\}$

input = bacbbccaacb

$S = \{q_0, q_{acc}, q_{rej}\}$

Transition function:

	a	b	c	*
q_0	(q_1,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_1	(q_2,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_2	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$



4.3 Turing machines

Example: the following Turing machine looks for the three consequtive occurrences of letter **a** in the input string. If it is found, the input is accepted, otherwise it is rejected.

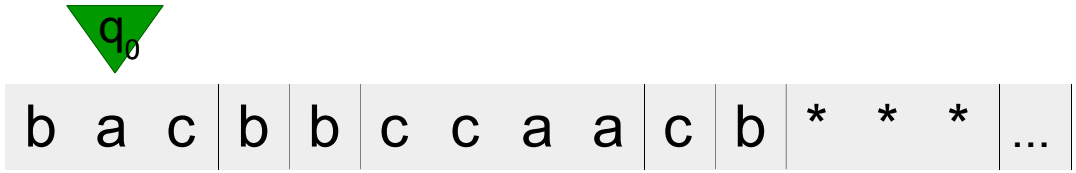
Γ = {a,b,c,*}

input = bacbbccaacb

S = $\{q_0, q_{acc}, q_{rej}\}$

Transition function:

	a	b	c	*
q_0	(q_1,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_1	(q_2,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_2	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$



4.3 Turing machines

Example: the following Turing machine looks for the three consequtive occurrences of letter **a** in the input string. If it is found, the input is accepted, otherwise it is rejected.

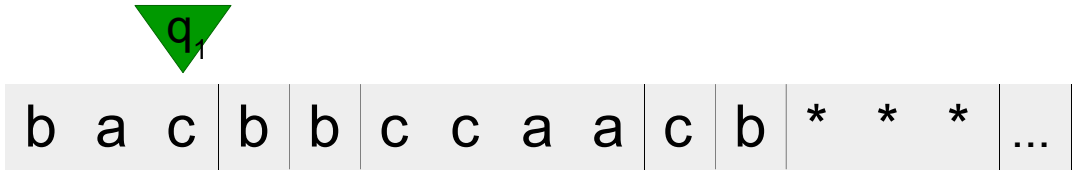
Γ = {a,b,c,*}

input = bacbbccaacb

S = $\{q_0, q_{acc}, q_{rej}\}$

Transition function:

	a	b	c	*
q_0	(q_1,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_1	(q_2,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_2	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$



4.3 Turing machines

Example: the following Turing machine looks for the three consequtive occurrences of letter **a** in the input string. If it is found, the input is accepted, otherwise it is rejected.

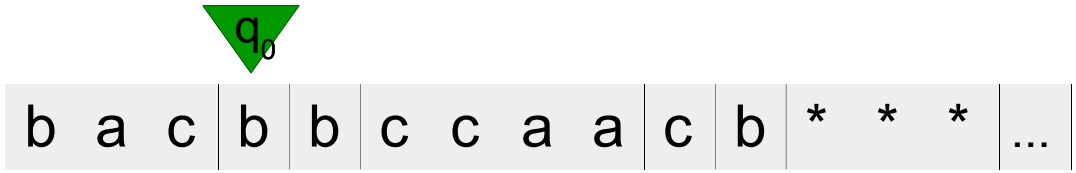
Γ = {a,b,c,*}

input = bacbbccaacb

S = $\{q_0, q_{acc}, q_{rej}\}$

Transition function:

	a	b	c	*
q_0	(q_1,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_1	(q_2,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_2	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$



4.3 Turing machines

Example: the following Turing machine looks for the three consequtive occurrences of letter **a** in the input string. If it is found, the input is accepted, otherwise it is rejected.

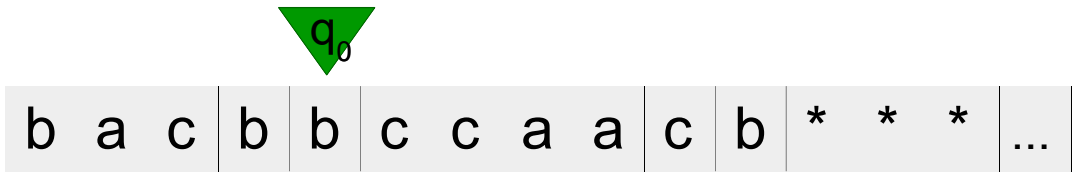
Γ = {a,b,c,*}

input = bacbbccaacb

S = $\{q_0, q_{acc}, q_{rej}\}$

Transition function:

	a	b	c	*
q_0	(q_1,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_1	(q_2,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_2	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$



4.3 Turing machines

Example: the following Turing machine looks for the three consequtive occurrences of letter **a** in the input string. If it is found, the input is accepted, otherwise it is rejected.

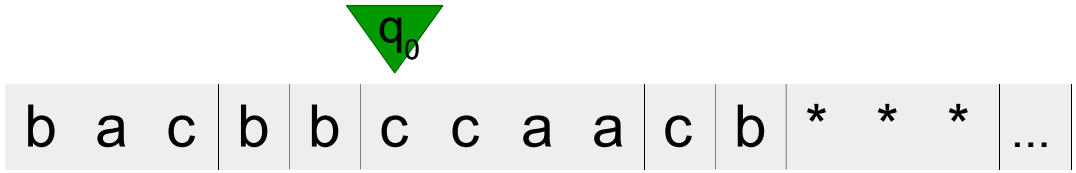
Γ = {a,b,c,*}

input = bacbbccaacb

S = $\{q_0, q_{acc}, q_{rej}\}$

Transition function:

	a	b	c	*
q_0	(q_1,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_1	(q_2,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_2	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$



4.3 Turing machines

Example: the following Turing machine looks for the three consequtive occurrences of letter **a** in the input string. If it is found, the input is accepted, otherwise it is rejected.

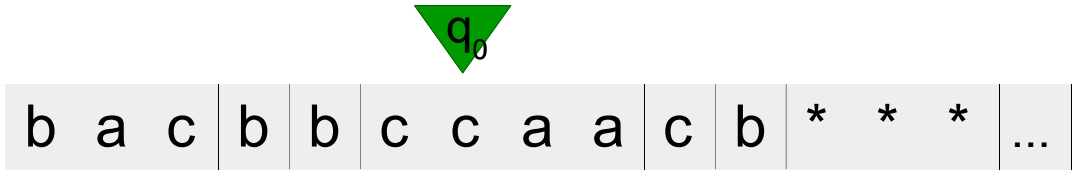
Γ = {a,b,c,*}

input = bacbbccaacb

S = $\{q_0, q_{acc}, q_{rej}\}$

Transition function:

	a	b	c	*
q_0	(q_1,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_1	(q_2,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_2	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$



4.3 Turing machines

Example: the following Turing machine looks for the three consequtive occurrences of letter **a** in the input string. If it is found, the input is accepted, otherwise it is rejected.

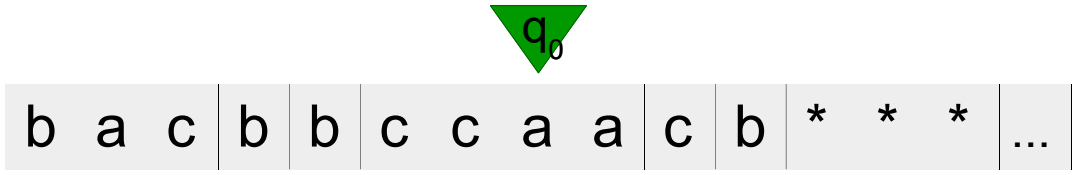
Γ = {a,b,c,*}

input = bacbbccaacb

S = $\{q_0, q_{acc}, q_{rej}\}$

Transition function:

	a	b	c	*
q_0	(q_1,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_1	(q_2,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_2	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$



4.3 Turing machines

Example: the following Turing machine looks for the three consequtive occurrences of letter **a** in the input string. If it is found, the input is accepted, otherwise it is rejected.

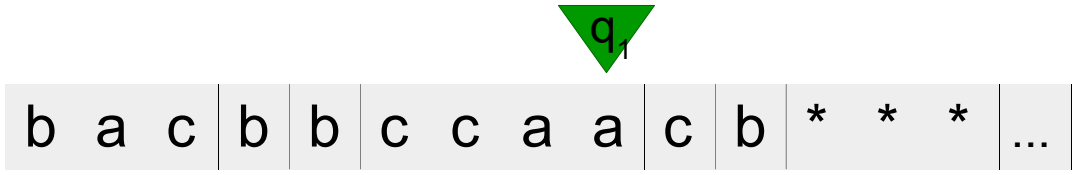
Γ = {a,b,c,*}

input = bacbbccaacb

S = $\{q_0, q_{acc}, q_{rej}\}$

Transition function:

	a	b	c	*
q_0	(q_1,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_1	(q_2,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_2	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$



4.3 Turing machines

Example: the following Turing machine looks for the three consequtive occurrences of letter **a** in the input string. If it is found, the input is accepted, otherwise it is rejected.

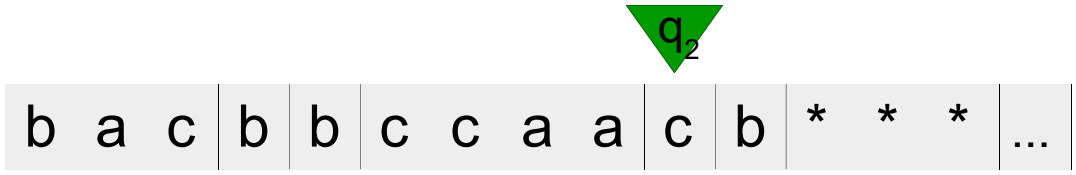
Γ = {a,b,c,*}

input = bacbbccaacb

S = $\{q_0, q_{acc}, q_{rej}\}$

Transition function:

	a	b	c	*
q_0	(q_1,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_1	(q_2,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_2	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$



4.3 Turing machines

Example: the following Turing machine looks for the three consequtive occurrences of letter **a** in the input string. If it is found, the input is accepted, otherwise it is rejected.

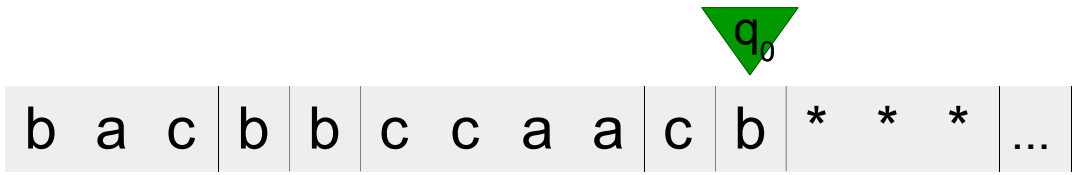
Γ = {a,b,c,*}

input = bacbbccaacb

S = $\{q_0, q_{acc}, q_{rej}\}$

Transition function:

	a	b	c	*
q_0	(q_1,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_1	(q_2,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_2	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$



4.3 Turing machines

Example: the following Turing machine looks for the three consequtive occurrences of letter **a** in the input string. If it is found, the input is accepted, otherwise it is rejected.

Γ = {a,b,c,*}

input = bacbbccaacb

S = $\{q_0, q_{acc}, q_{rej}\}$

Transition function:

	a	b	c	*
q_0	(q_1,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_1	(q_2,a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$
q_2	(q_{acc},a,R)	(q_0,b,R)	(q_0,c,R)	$(q_{rej},*,R)$