CS 6041
Theory of Computation

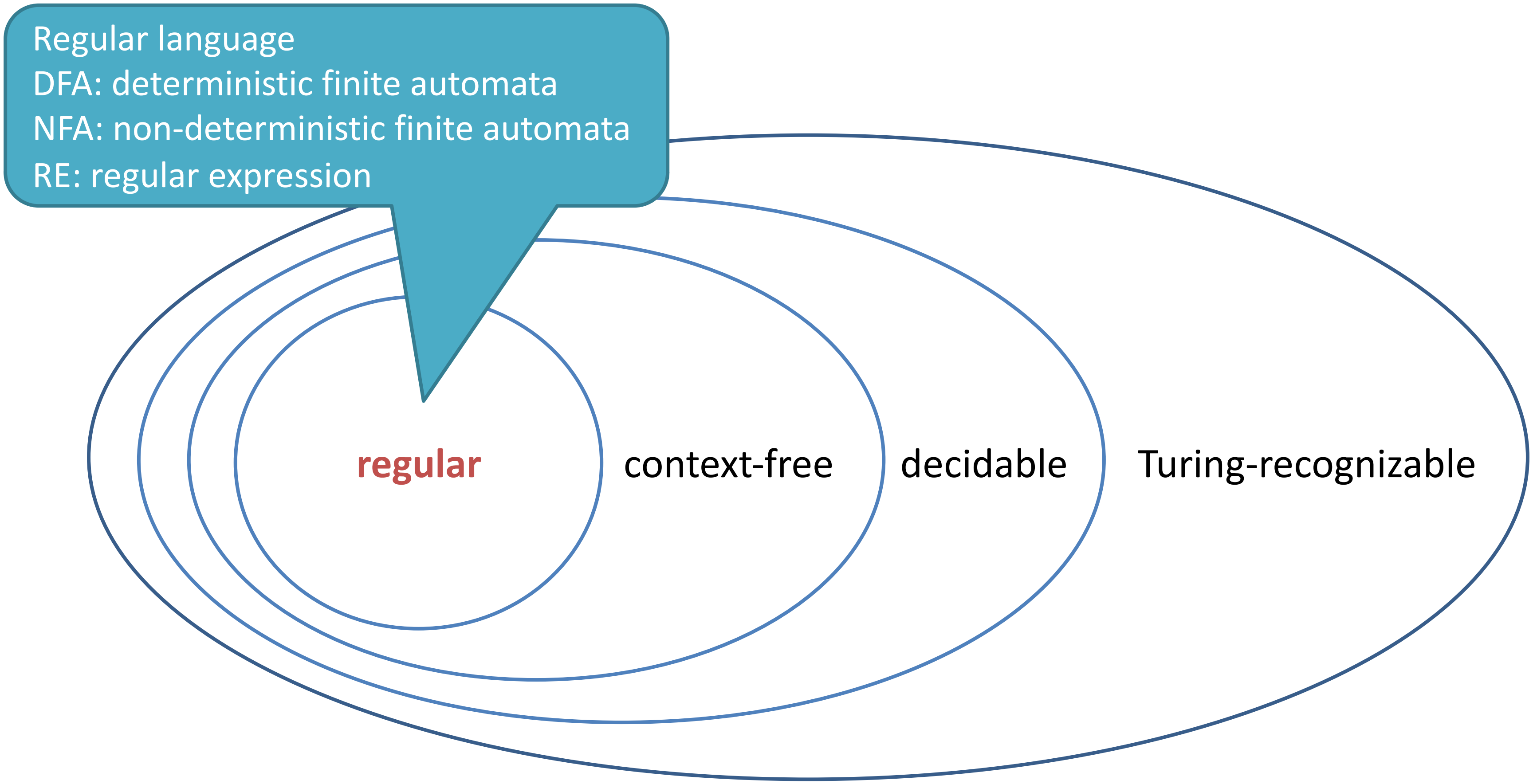
Turing machine

Kun Suo

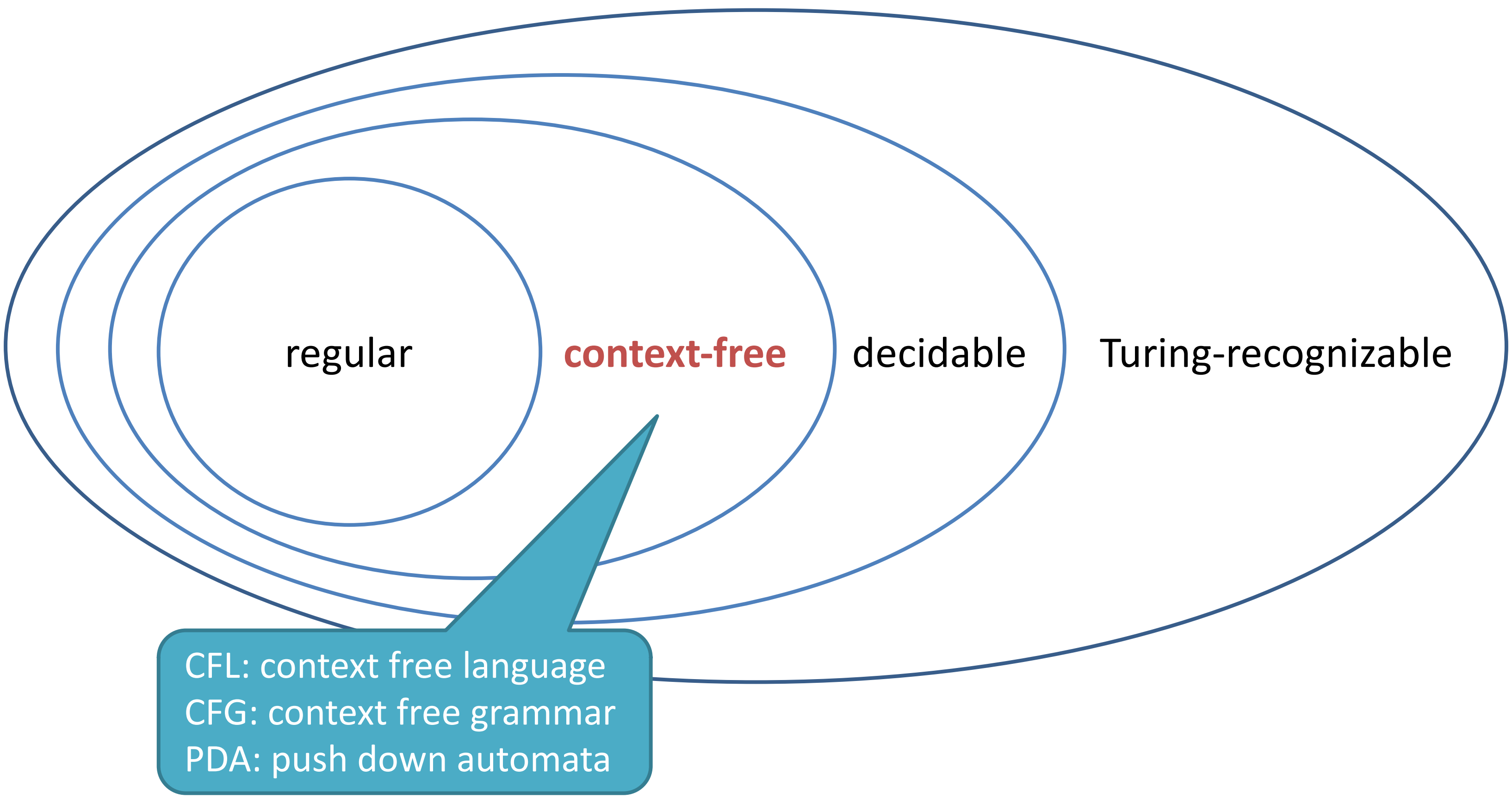
Computer Science, Kennesaw State University

https://kevinsuo.github.io/

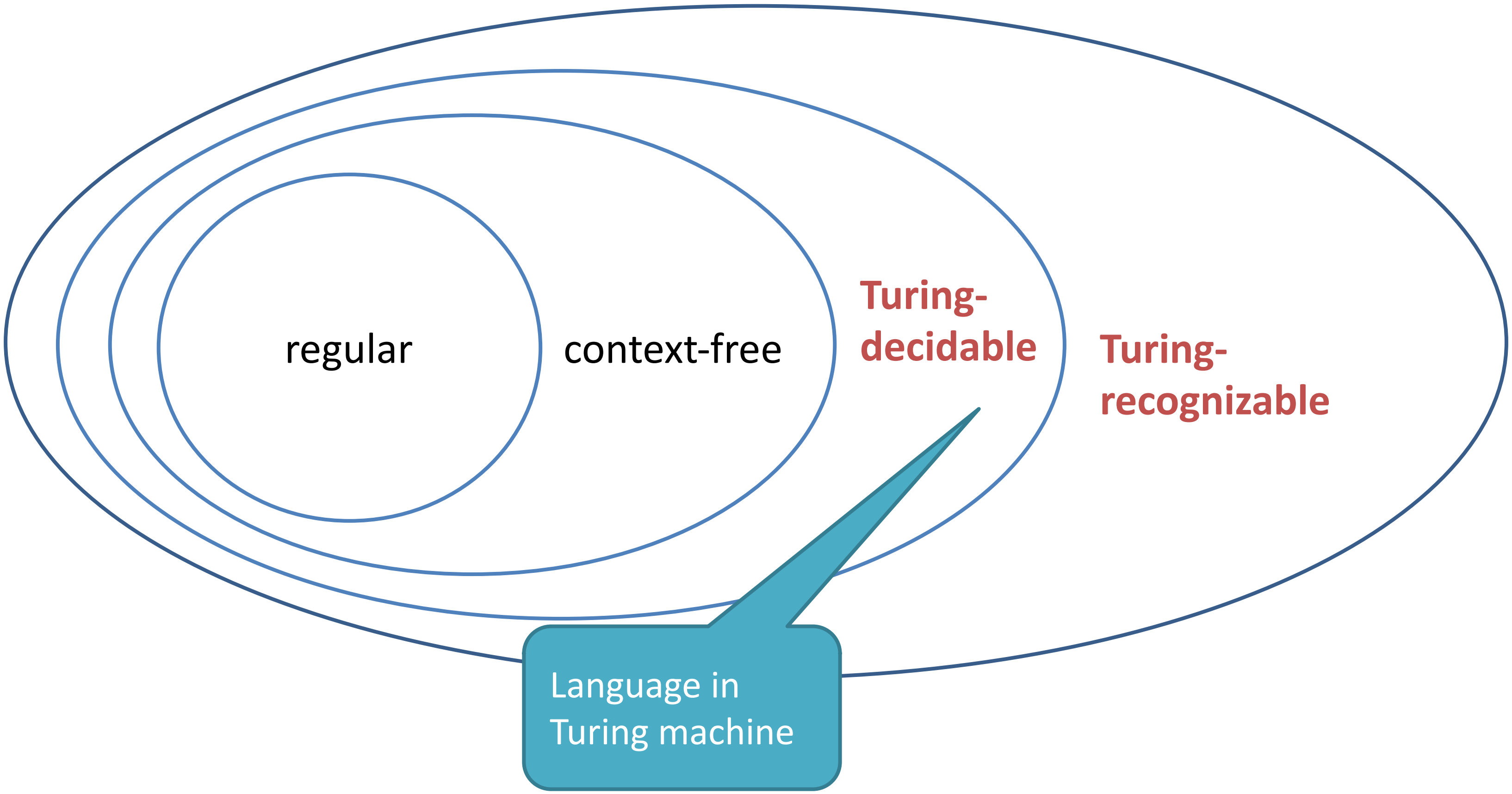
Turing machine



Turing machine



Turing machine



What does Turing Machine look like?

https://www.youtube.com/watch?v=E3keLeMwfHY

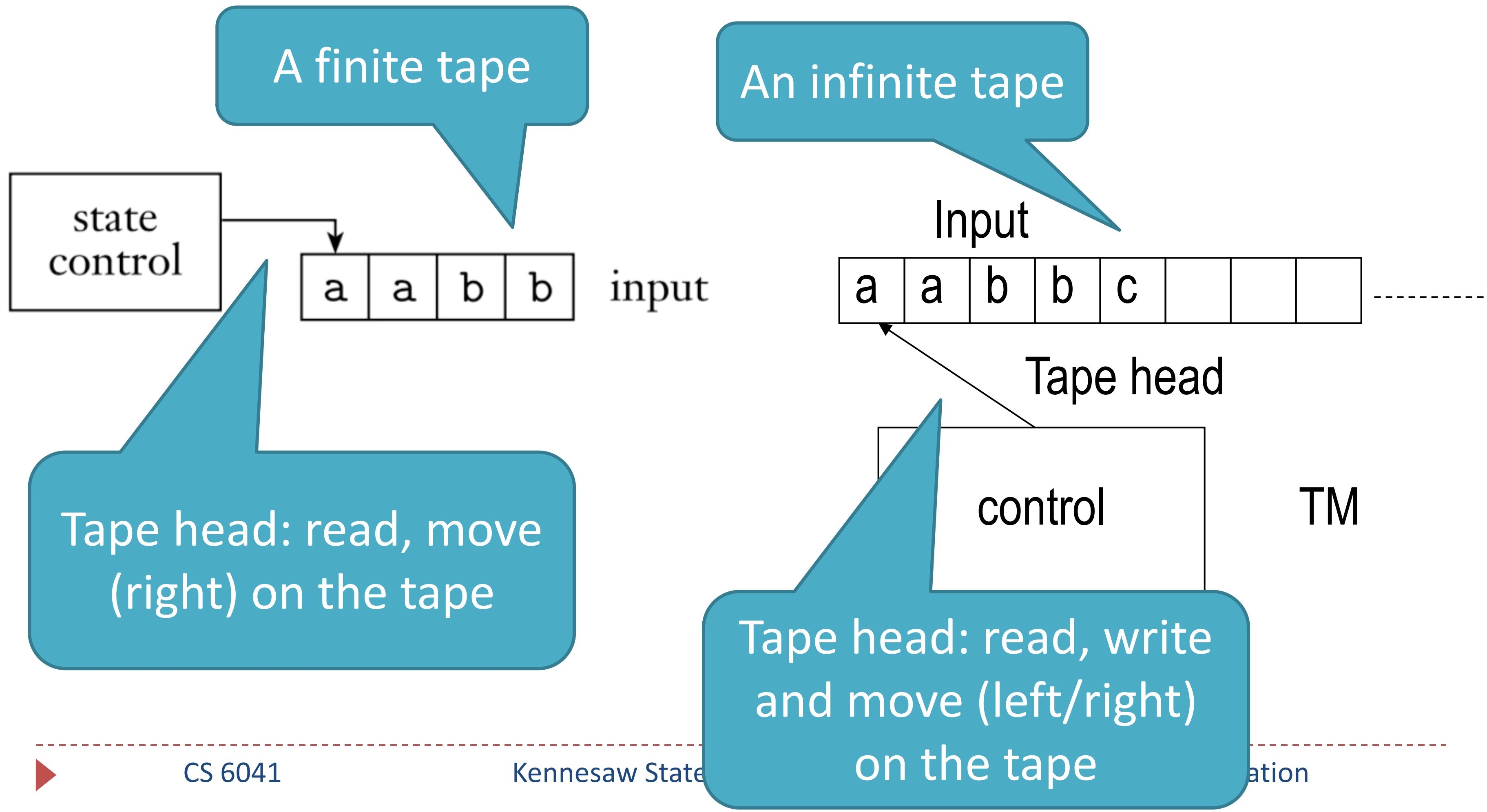
Turing machine

- *On computable numbers, with an application to the Entscheidungsproblem, 1930s*

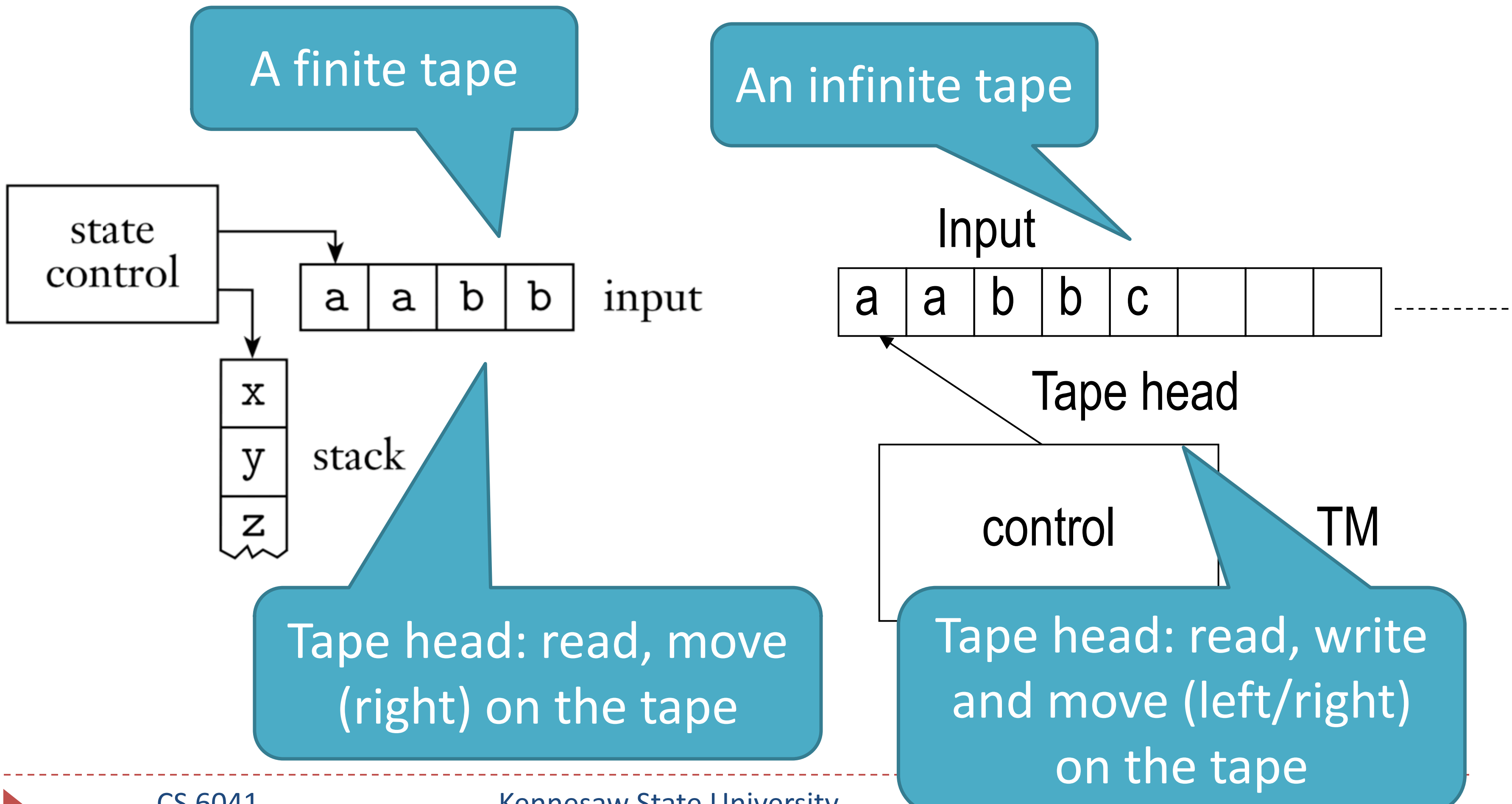
https://londmathsoc.onlinelibrary.wiley.com/doi/pdf/10.1112/plms/s2-42.1.230

Alan Turing

Question: based on the above video, what is the similarity and difference between finite automata and Turing machine?

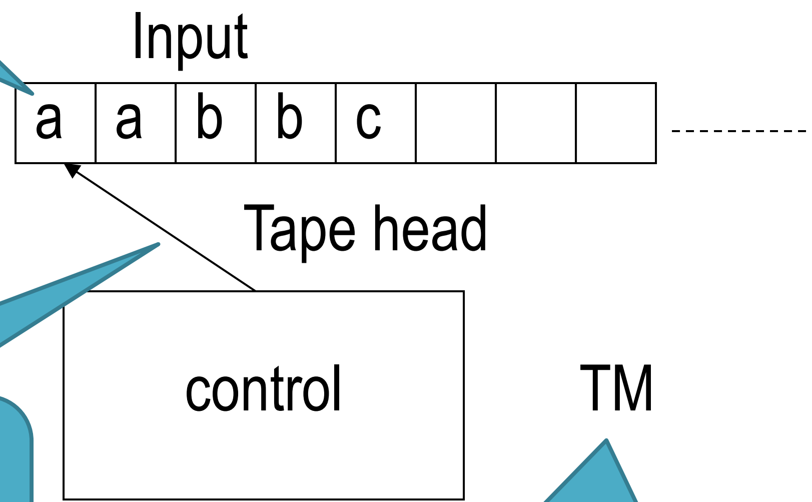


Question: based on the above video, what is the similarity and difference between pushdown automata and Turing machine?



Turing Machine

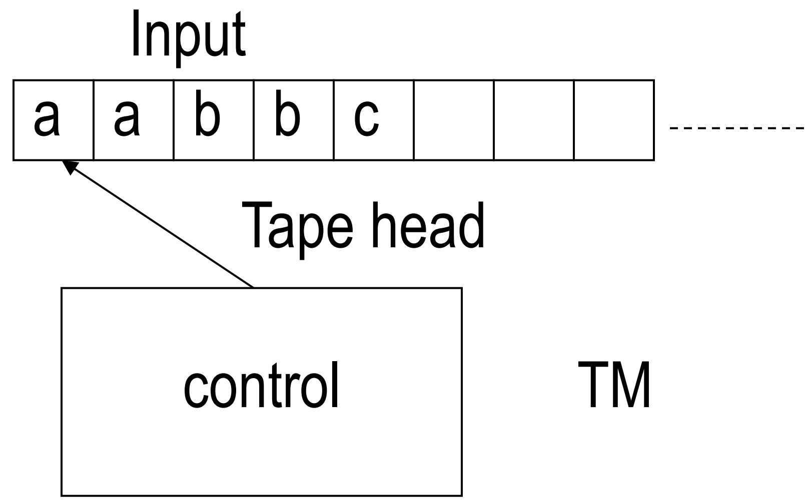
An infinite tape: unlimited memory



Tape head: read, write and move (left/right) on the tape

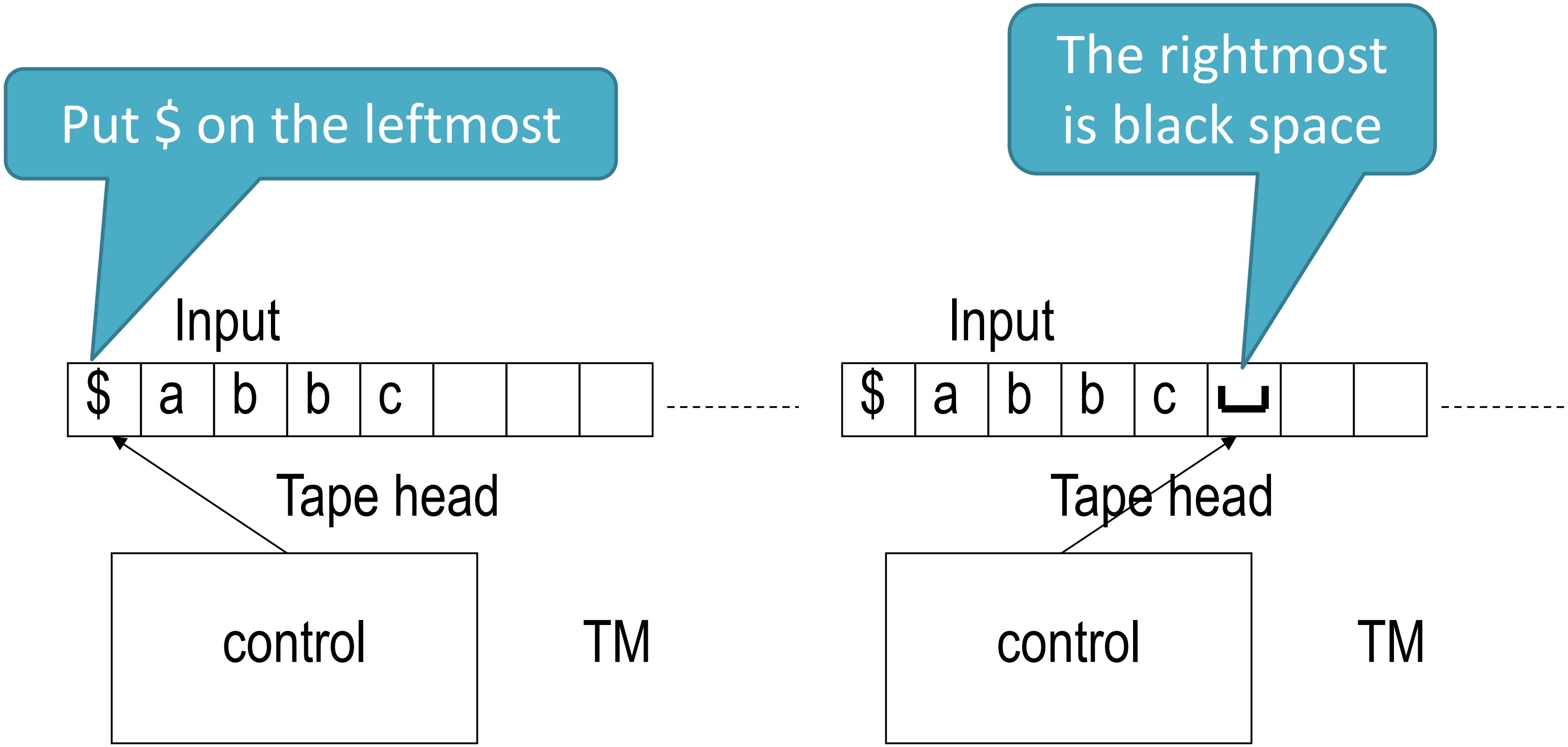
The machine continues computing until it produces an output (accept/reject)

Turing machine vs. finite automata vs. Pushdown automata

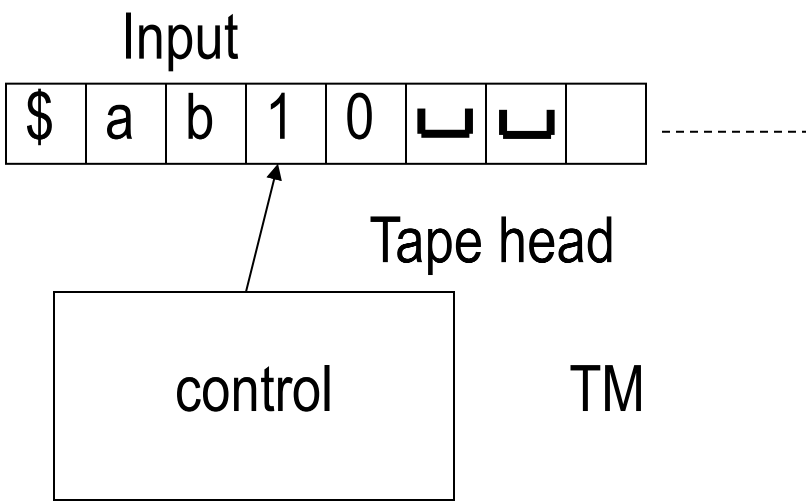


	Finite automata	Pushdown automata	Turing machine
Header			
Header move			
Input			
Output			

Turing machine: left end and right end of tape

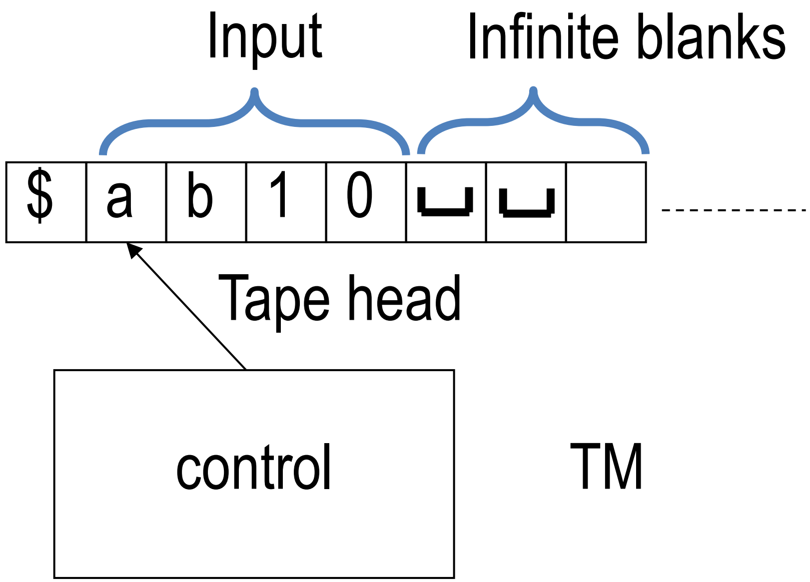


Input on the tape of TM



- $\Sigma = \{a, b, 0, 1, \ldots\}$
- $␣ \notin \Sigma$
- The blank symbol is just used to fill the infinite tape of TM

Initial state and operations of TM



- Operations:
  - Read symbol below the head
  - Write symbol below the head
  - Move head one step left
  - Move head one step right

Definition of Turing Machine

- TM $M=(Q,\Sigma,\Gamma,\delta,q_0,q_{acc},q_{rej})$

1) Q is the set of states
2) Σ is the input alphabet, not containing blank symbol $B \notin \Sigma$
3) Γ is the tape alphabet, $\Sigma \cup \{B\} \subseteq \Gamma$,
4) δ: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$ is the transition function
5) $q_0 \in Q$ is the start state
6) $q_{acc} \in Q$ is the accept state
7) $q_{rej} \in Q$ is the reject state, $q_{acc} \neq q_{rej}$

Tape includes input alphabet and space

Definition comparison

A ***pushdown automaton*** is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$, where Q, Σ, Γ, and F are all finite sets, and

1. Q is the set of states,
2. Σ is the input alphabet,
3. Γ is the stack alphabet,
4. $\delta\colon Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \longrightarrow \mathcal{P}(Q \times \Gamma_\varepsilon)$ is the transition function,
5. $q_0 \in Q$ is the start state, and
6. $F \subseteq Q$ is the set of accept states.

A ***Turing machine*** is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the ***blank symbol*** $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta\colon Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{reject}} \neq q_{\text{accept}}$.

Question: Y or N?

- Can a Turing machine ever write the blank symbol ␣ on its tape?
  - Yes. The tape alphabet Γ contains ␣. A Turing machine can write any characters in Γ on its tape.

Question: Y or N?

- Can the tape alphabet Γ be the same as the input alphabet Σ?
  - No. Σ never contains ␣, but Γ always contains ␣. So they cannot be equal.

Question: Y or N?

- Can a Turing machine's head *ever* be in the same location in two successive steps?

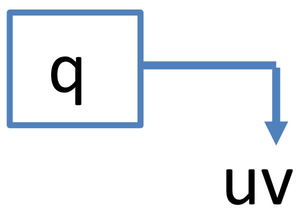
  - Yes. If the Turing machine attempts to move its head off the left-hand end of the tape, it remains on the same tape cell.

Question: Y or N?

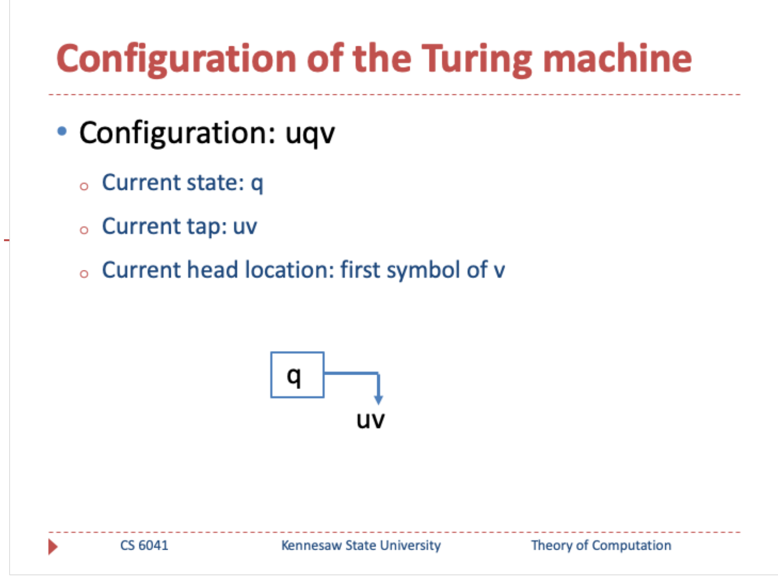
- Can a Turing machine contain just a single state?
  - No. Any Turing machine must contain two distinct states: q_{accept} and q_{reject}. So, a Turing machine contains at least two states.

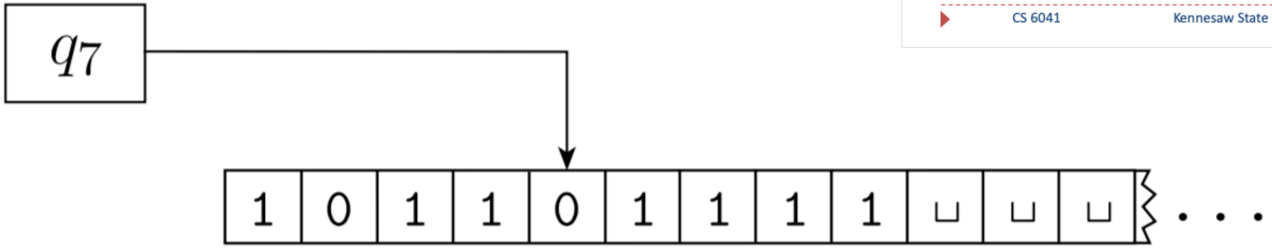
Configuration of the Turing machine

- Configuration: uqv
  - Current state: q
  - Current tap: uv
  - Current head location: first symbol of v



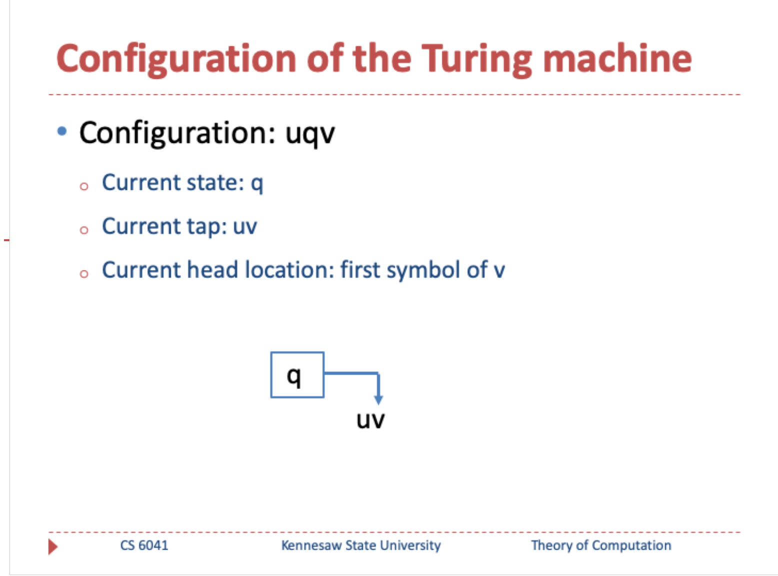
Question: what is the current configuration?





configuration $1011q_701111$

Question: what is the start configuration?



- TM $M=(Q,\Sigma,\Gamma,\delta,q_0,q_{acc},q_{rej})$
- w is the input

Start configuration: $q_0\, w$

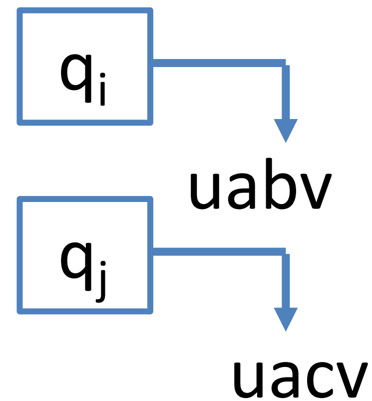
Configuration of the Turing machine

- Start configuration: q_0w, w is the input
- Accepting configuration: $uq_{accept}v$
- Rejecting configuration: $uq_{reject}v$
- Halting configuration: $uq_{accept}v$, $uq_{reject}v$

Yield configuration

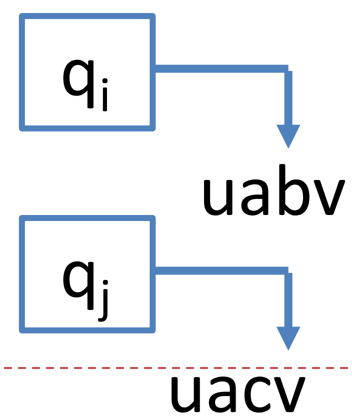
- Configuration C_1 ***yields*** configuration C_2 if the Turing machine can legally go from C_1 to C_2 in a single step.
  - **If $\delta(q_i,b)=(q_j,c,L)$, then**
    (1) uaq_ibv yields uq_jacv
    (2) q_ibv yeilds q_jcv
    (when the header is already left-most)

Under state qi,
input with b
-->
Change to state qj,
b changes to c,
header move to left



  - **If $\delta(q_i,b)=(q_j,c,R)$, then**
    uaq_ibv yields $uacq_jv$

Under state qi,
input with b
-->
Change to state qj,
b changes to c,
header move to right



A Turing machine M *accepts* input w

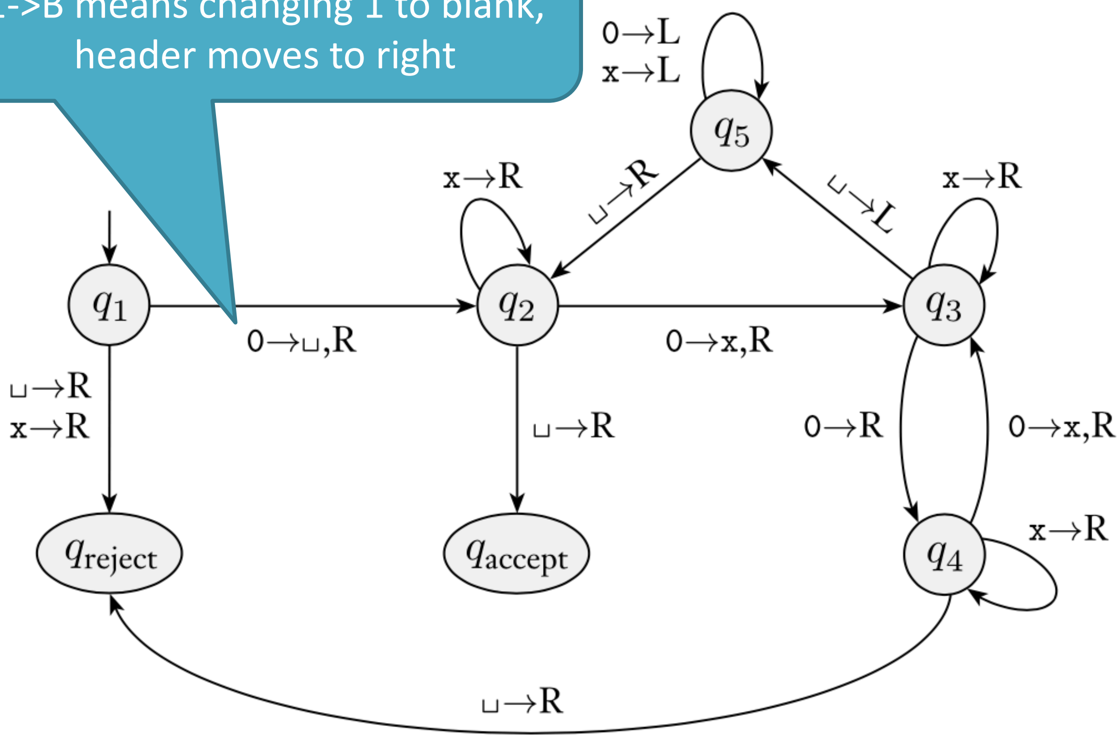
- M *accepts* input w

  if a sequence of configurations $C_1, C_2, \ldots, C_k$ exists, where

  - **1.** C_1 is the start configuration of M on input w,
  - **2.** each C_i yields C_{i+1}, and
  - **3.** C_k is an accepting configuration.

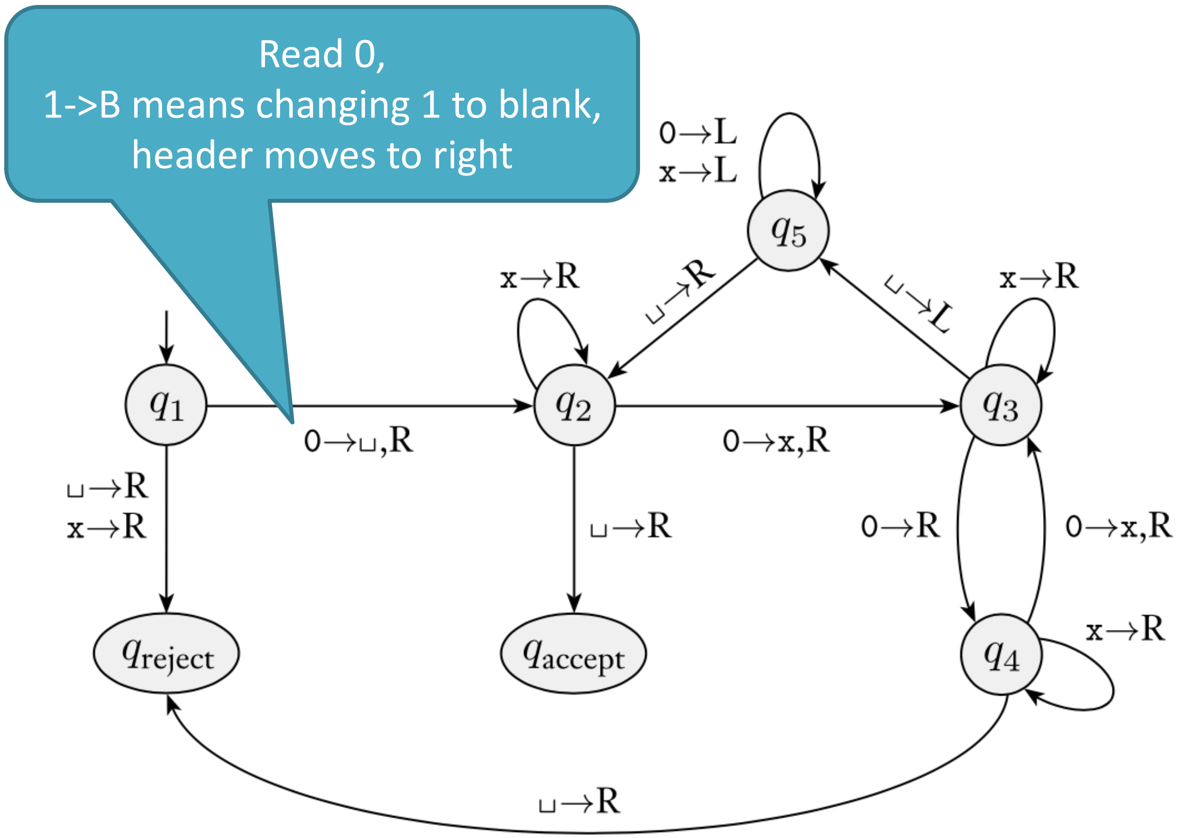
Question: give the sequence of configurations that M_2 enters when started on the indicated input string.

Read 0,
1->B means changing 1 to blank,
header moves to right



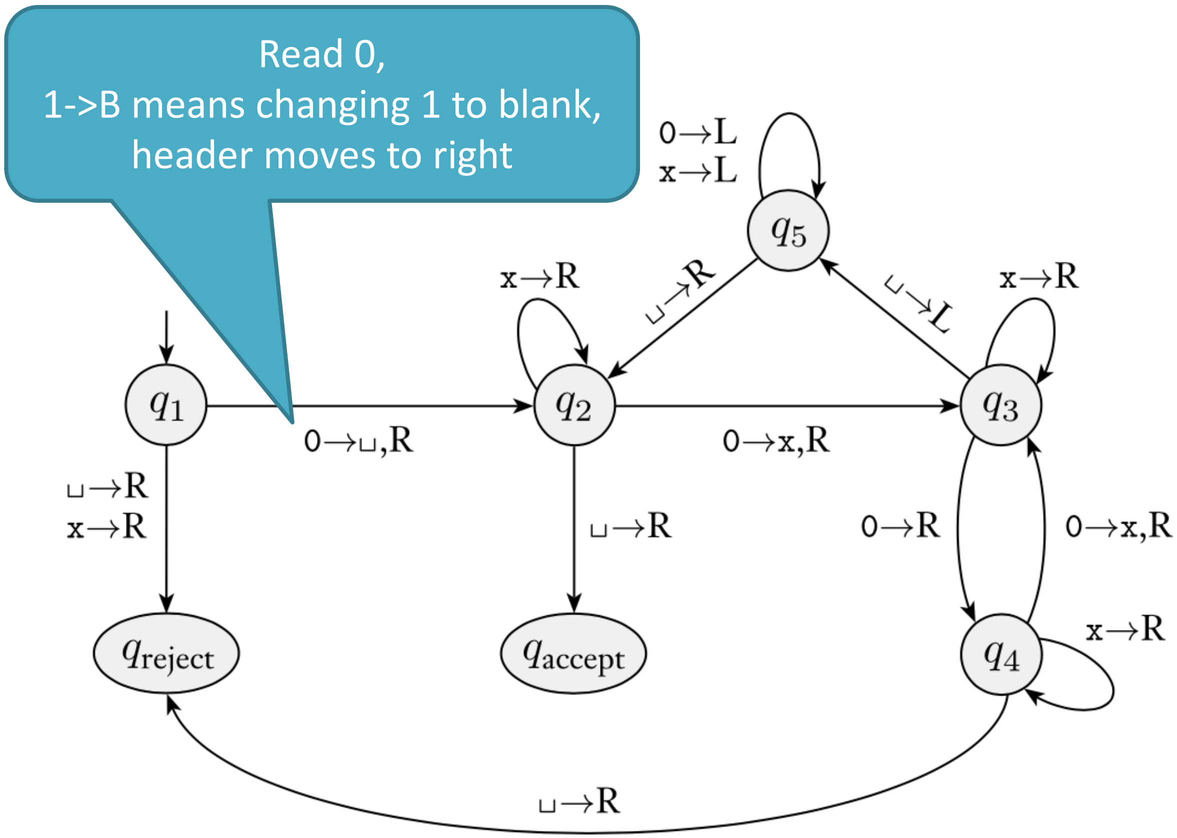
- 0
  - $q_1 0$
  - $\sqcup q_2 \sqcup$
  - $\sqcup \sqcup q_{accept}$

Question: give the sequence of configurations that M_2 enters when started on the indicated input string.



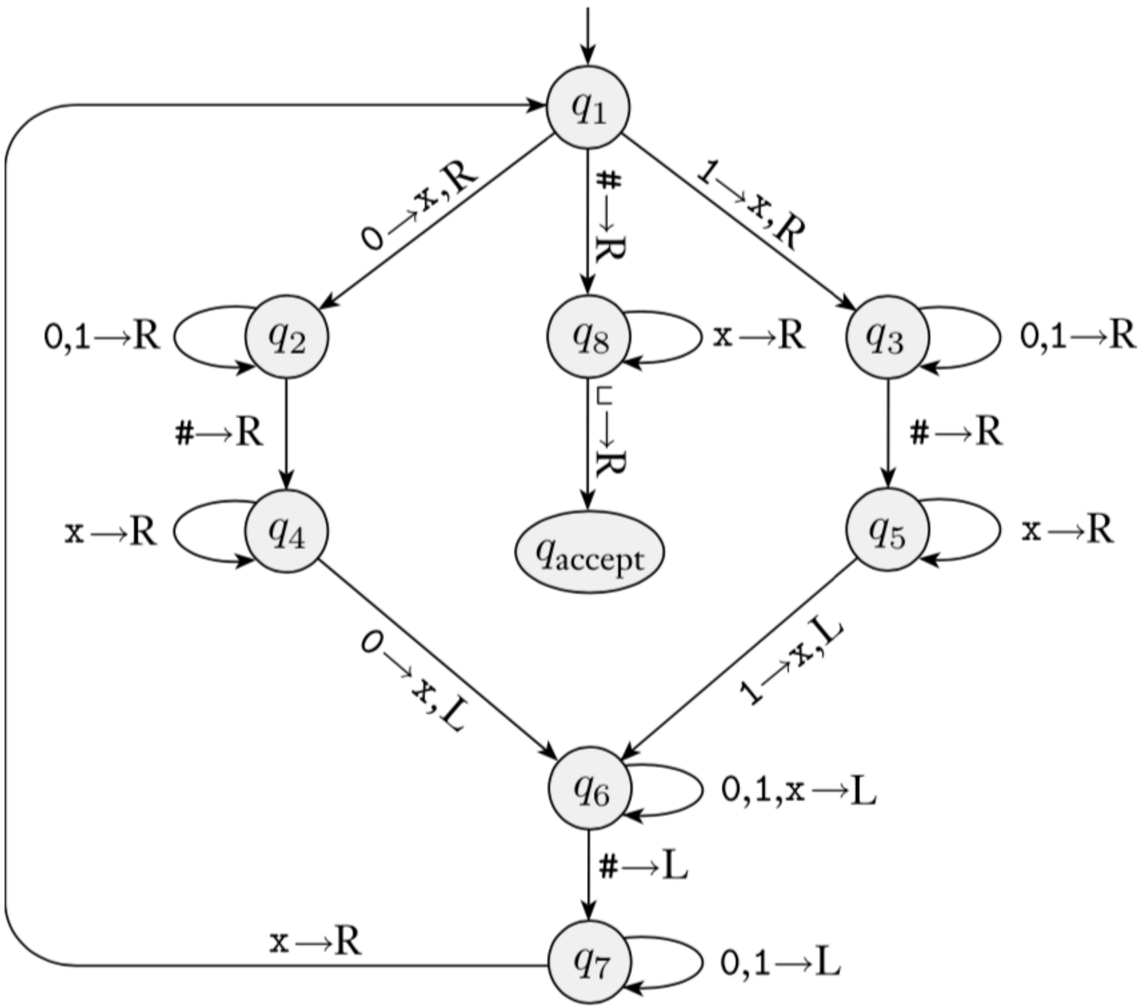
- 00
  - $q_1 00$

Question: give the sequence of configurations that M_2 enters when started on the indicated input string.



- 000
  - $q_1 000$

Question: give the sequence of configurations that M_1 enters when started on the indicated input string.



- 11

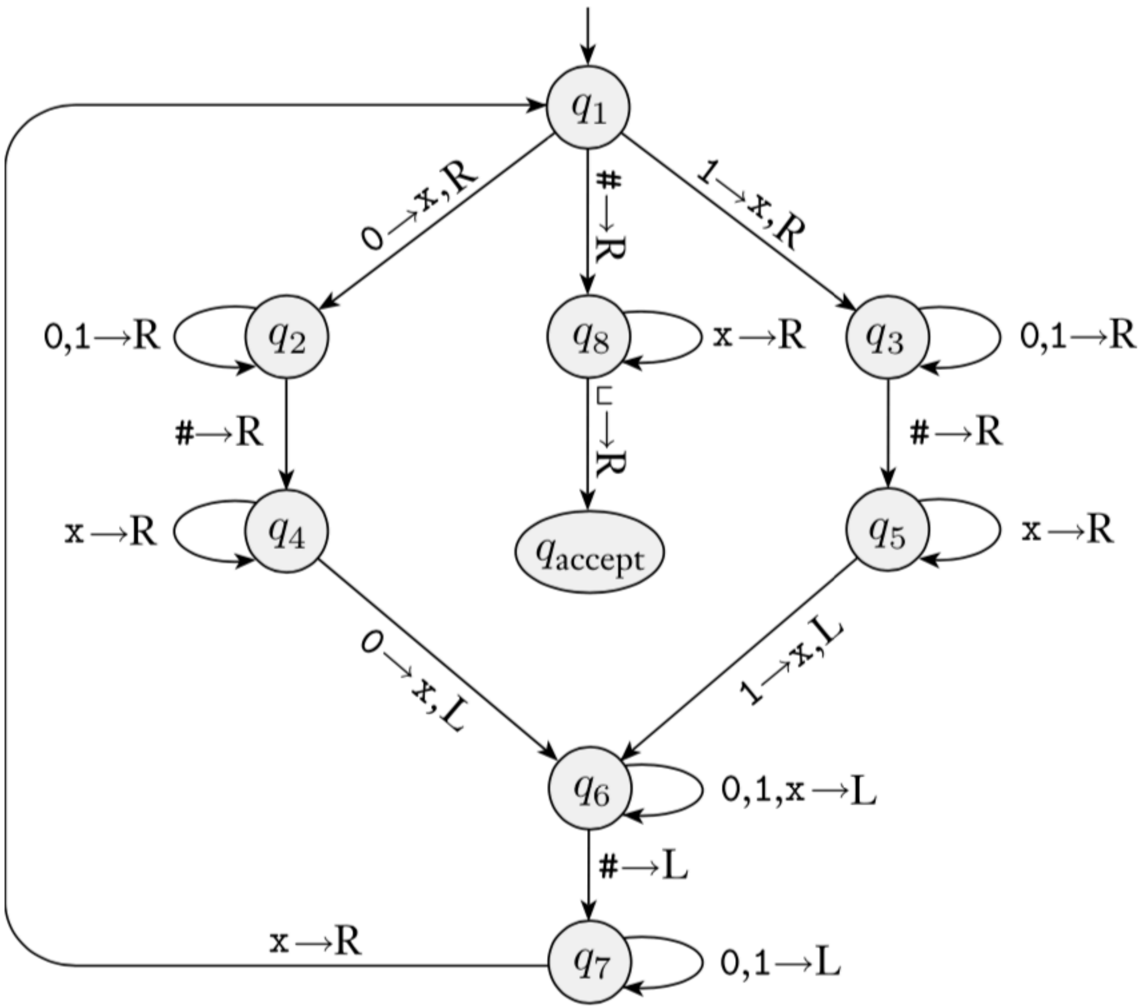
  $q_1 11$

  $\sqcup\ q_3 1$

  $\sqcup\ 1 q_3 \sqcup$

  $\sqcup\ 1 \sqcup q_{reject}$

Question: give the sequence of configurations that M_1 enters when started on the indicated input string.



- 1##1

  $q_1$1##1

  ␣ q_3##1

  ␣ #q_5#1

  ␣ ## q_{reject}1

The output of Turing Machine

- Accept
- Reject
- Loop

$\left.\begin{array}{l}\text{Accept}\\\text{Reject}\end{array}\right\}$ Halt

Loop = Never Halt

For finite automata and pushdown automata, they will halt

Examples of TM accepts, rejects and loop

- Accept
- Reject

} Halt

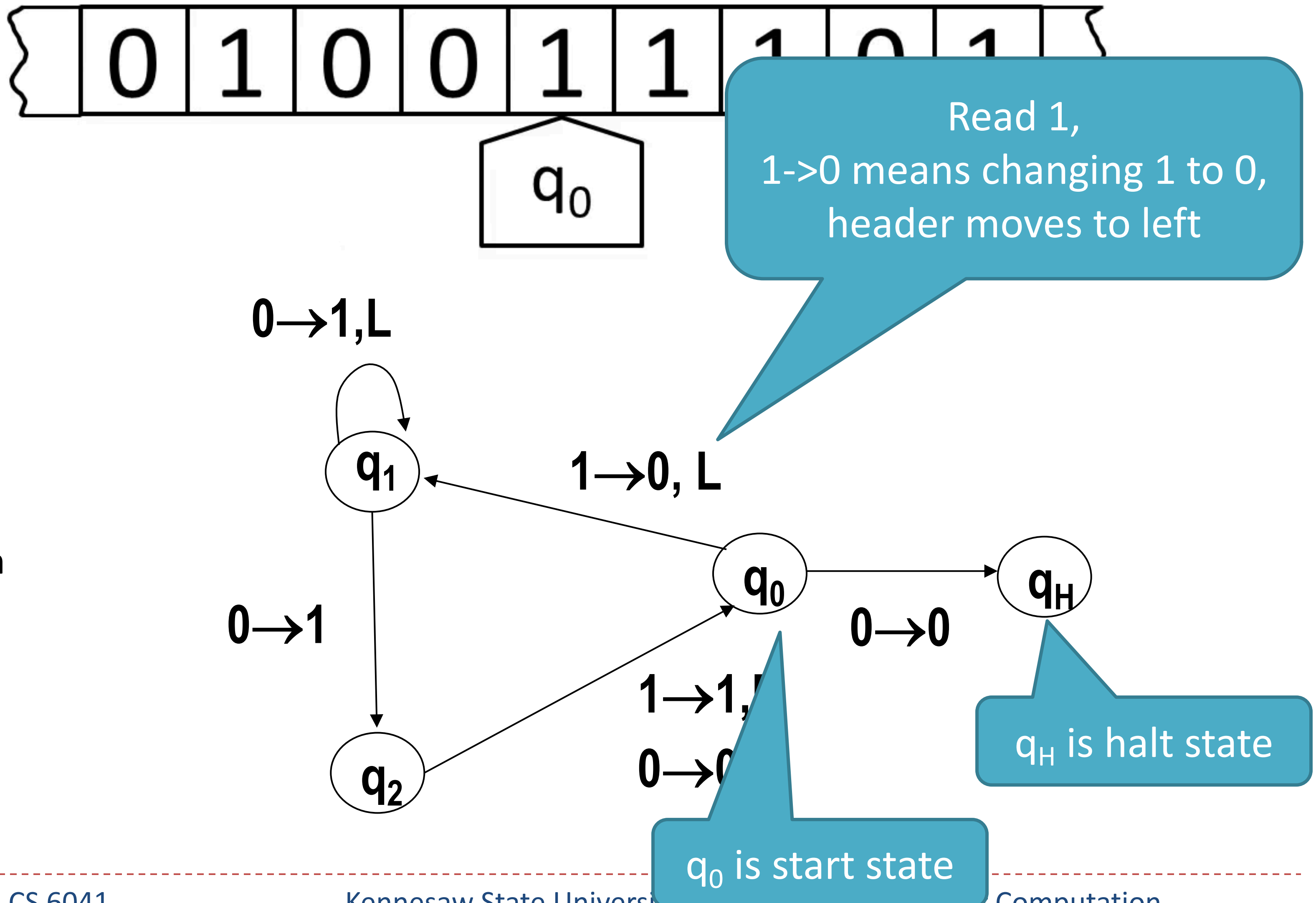
- Loop = Never

Can anyone give an example of never halting?

a.c

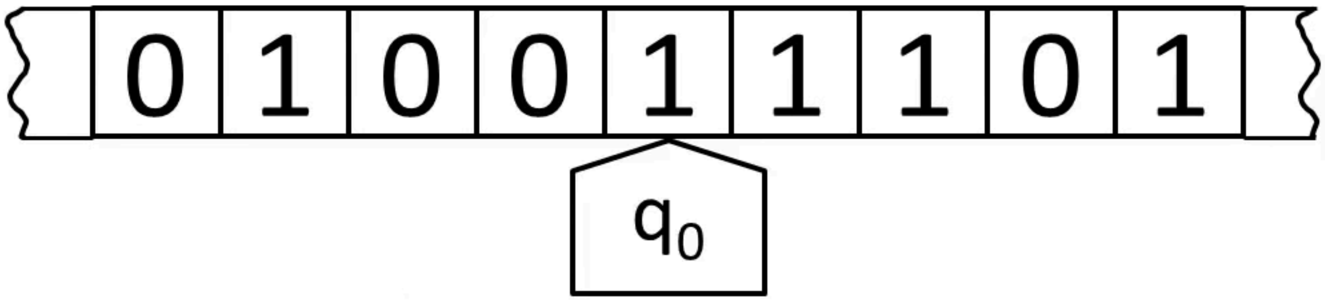
```c
#include <stdio.h>
int main()
{
    int c;

    printf( "Enter a value :");
    c = getchar( );

    switch(c) {
        case '0' :
            printf("accept!\n" );
            break;
        case '1' :
            printf("reject\n" );
            break;
        default :
            while(1)
            {
                //do nothing
            };
    }

    return 0;
}
```

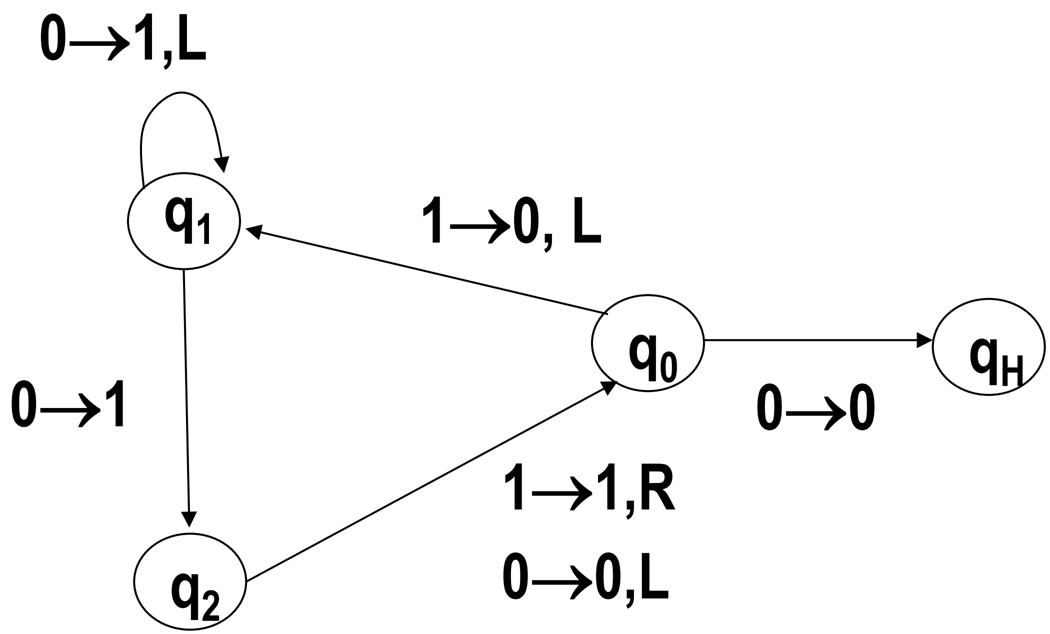
TM example



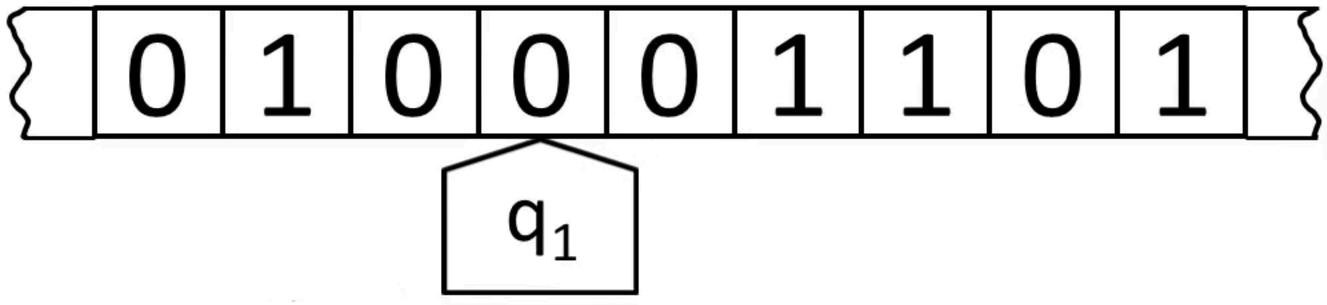
Transition function

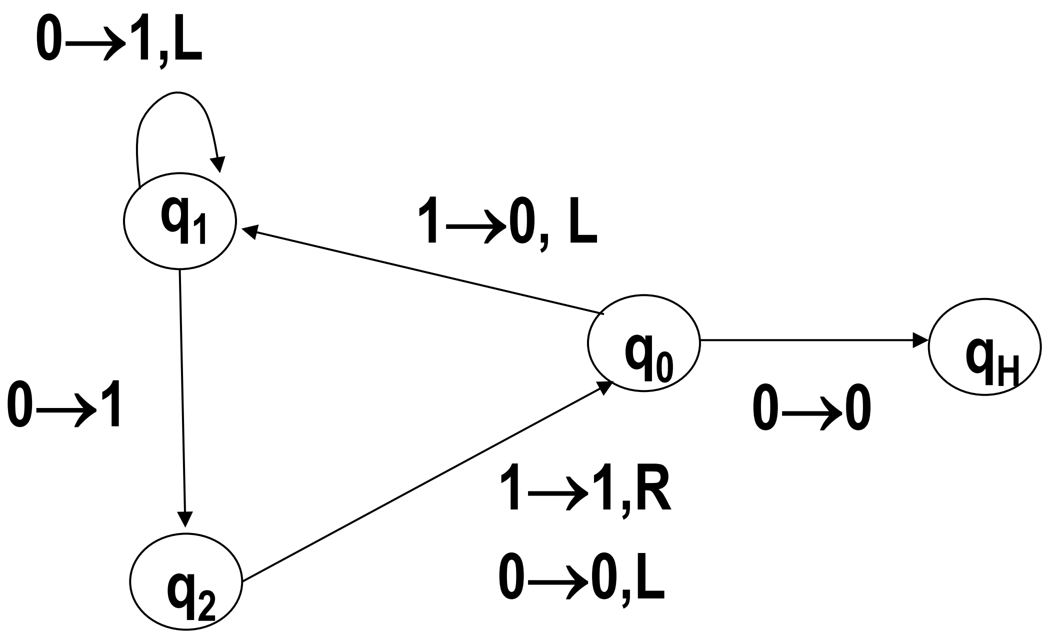
TM example



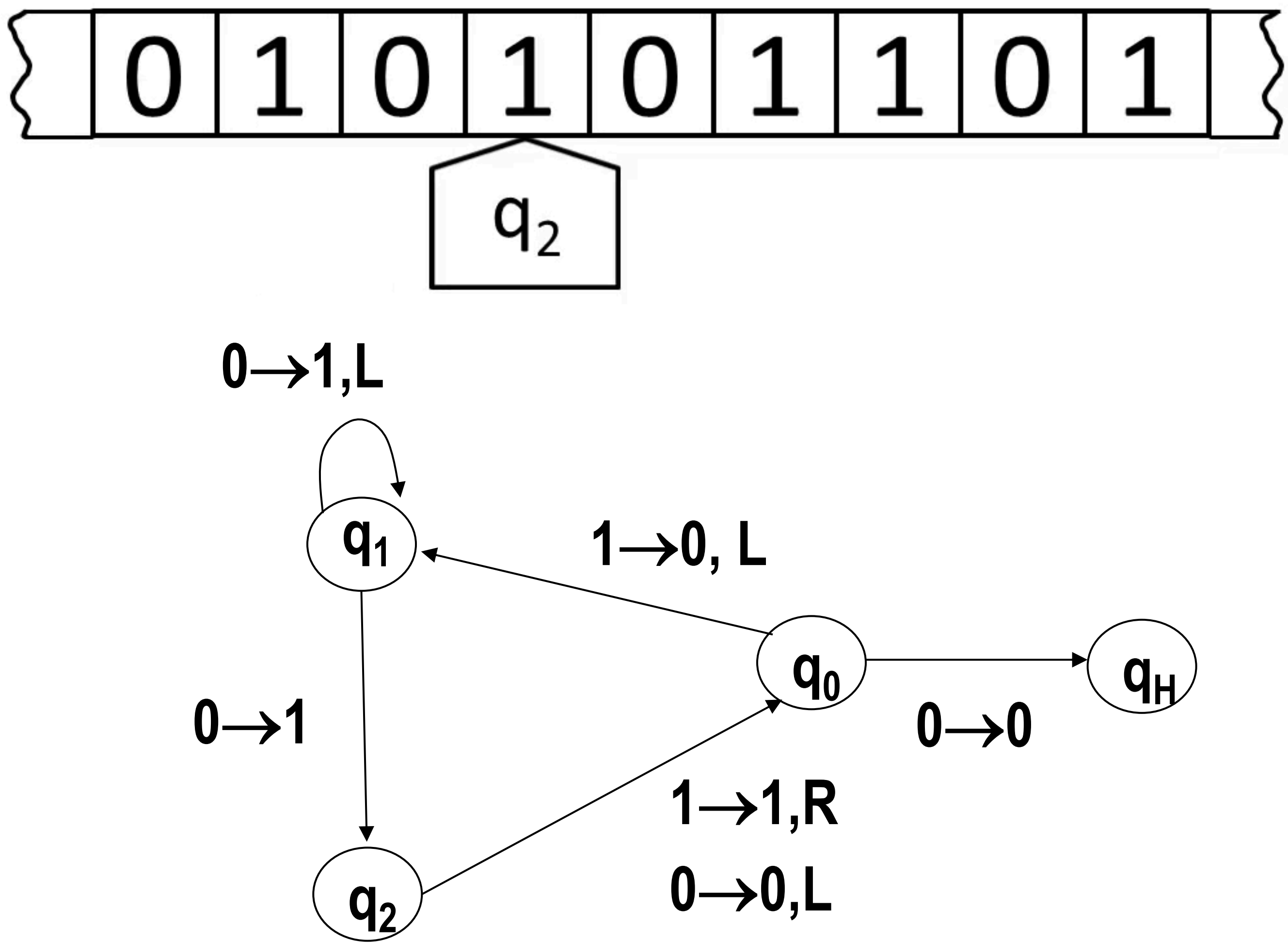


TM example

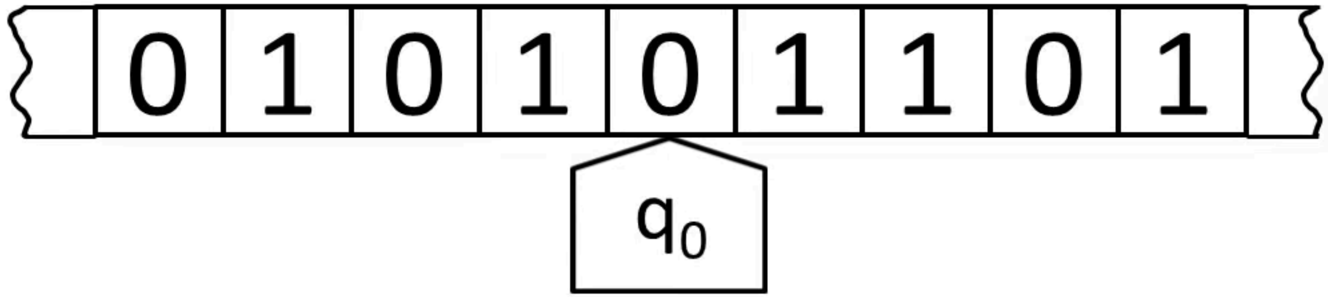


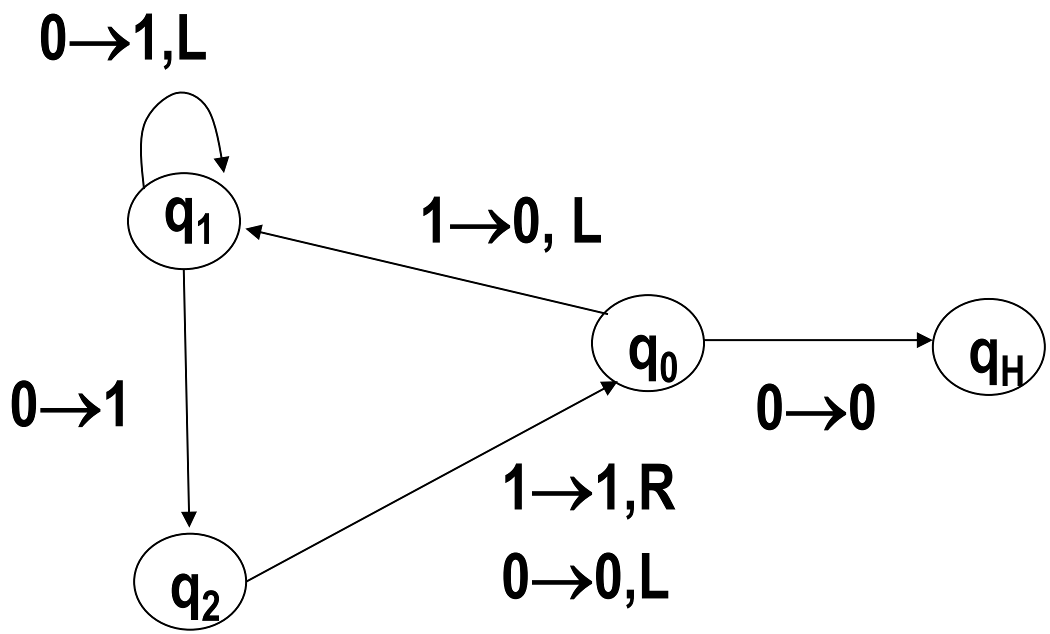


TM example

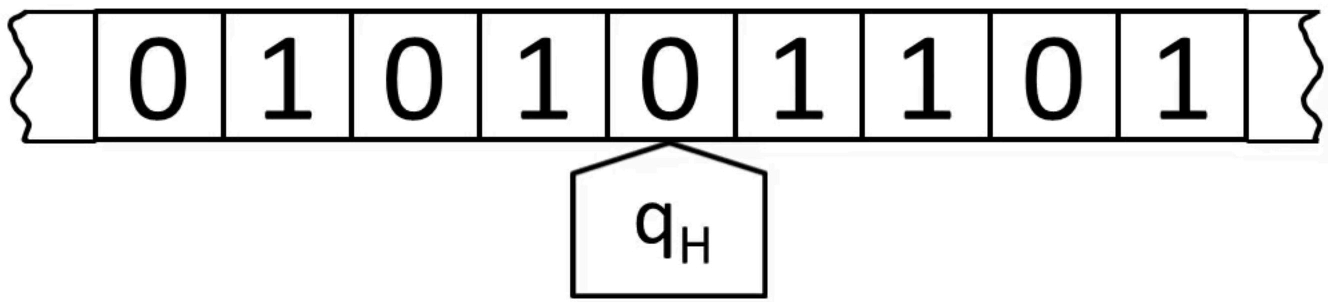


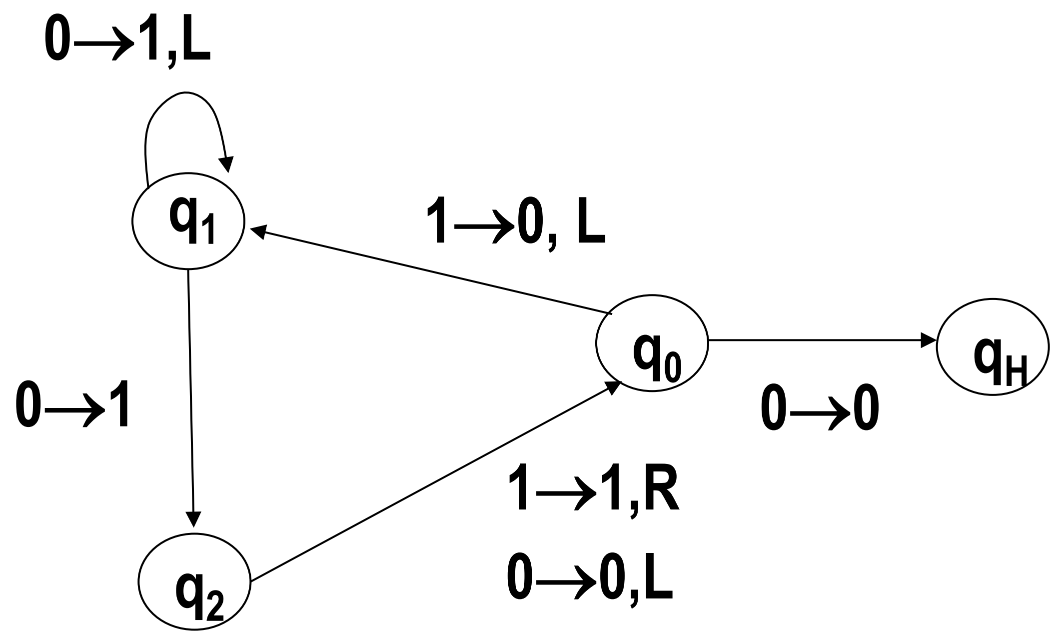
TM example



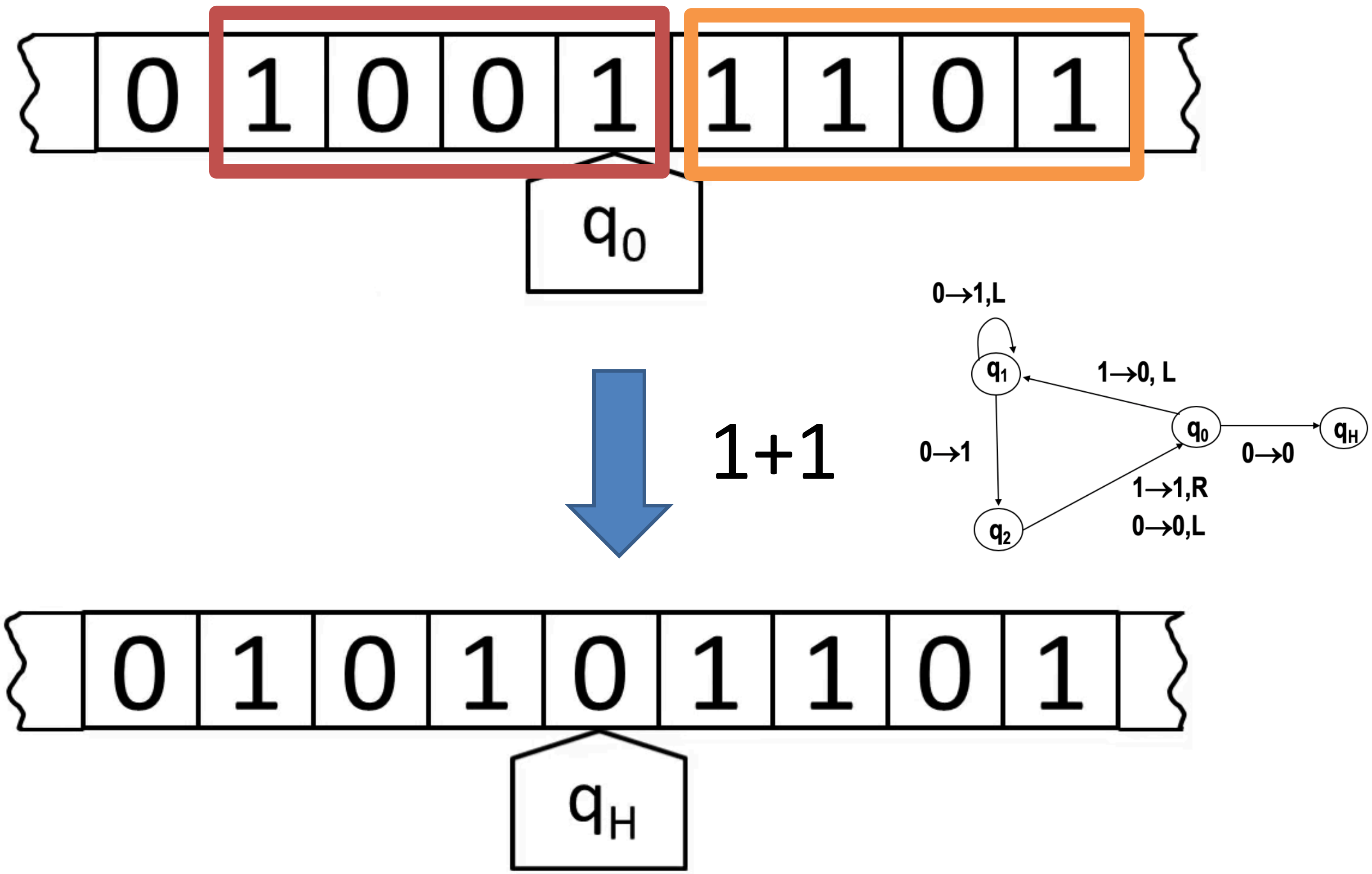


TM example





TM example: 1+1 operation



TM example: $B = \{ w\#w \mid w \in \{0,1\}^* \}$

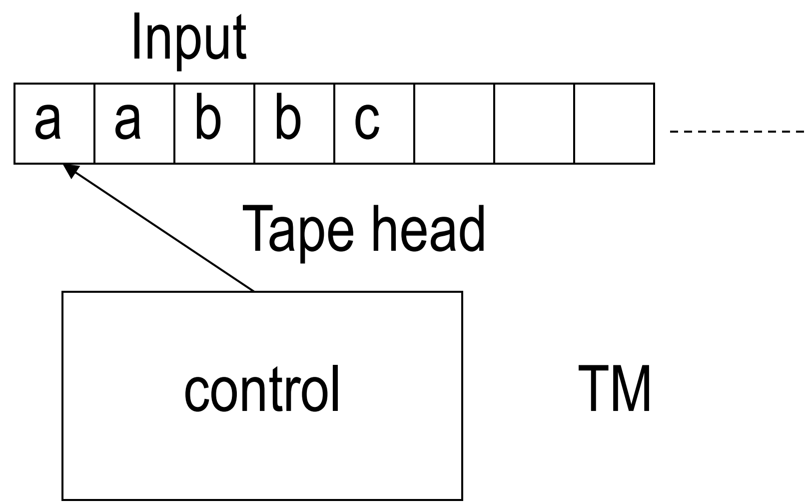
- M_1= "for input string x":
  1. Scan the input to make sure there exists only one "#", otherwise reject;
  2. Move to the same positions on both sides between "#", check whether there exist same symbols. If not, reject; otherwise, cross off the checked symbols;
  3. If all symbols on the left of "#" are crossed off, check whether there exists other remaining symbols on the right. If yes, reject; otherwise, accept.

M1 computation for 011000#011000

0 1 1 0 0 0 # 0 1 1 0 0 0

↑

q_{start} state



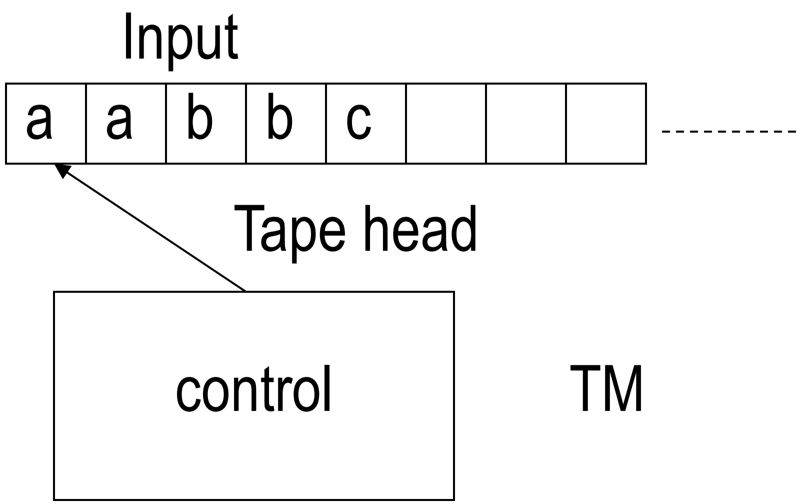
M1 computation for 011000#011000

X 1 1 0 0 0 # 0 1 1 0 0 0

$\uparrow_0$

q_{start} state

q_0 state: crossed off a 0



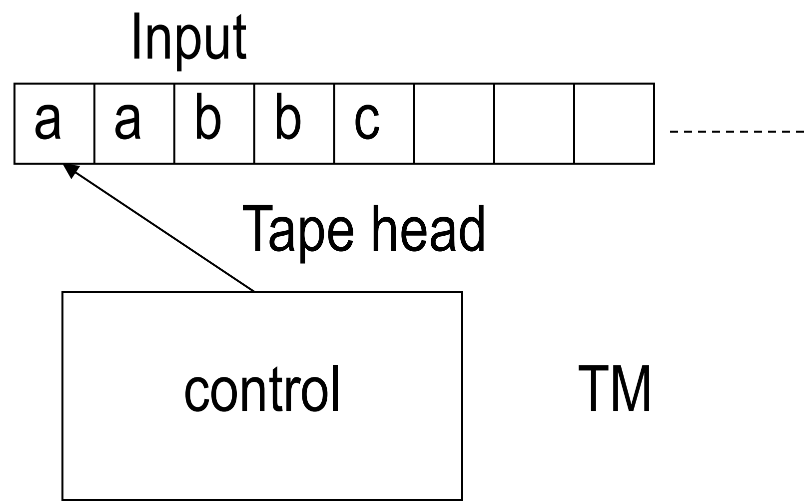
M1 computation for 011000#011000

X 1 1 0 0 0 # 0 1 1 0 0 0

$\uparrow_{\mathbf{0}}$

q_{start} state

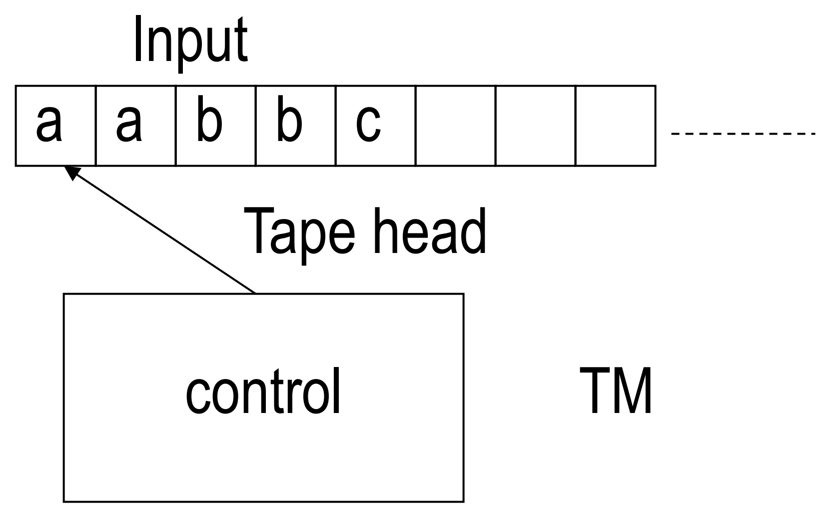
q_0 state: crossed off a 0



M1 computation for 011000#011000

X 1 1 0 0 0 # 0 1 1 0 0 0

$\uparrow_{0,\#}$



q_0 state: crossed off a 0

$q_{0\#}$ state: crossed off a 0, read a #

M1 computation for 011000#011000

X 1 1 0 0 0 # X 1 1 0 0 0

$\uparrow_{\leftarrow}$

Under this state, if read one 0, cross off the 0, then move to the left

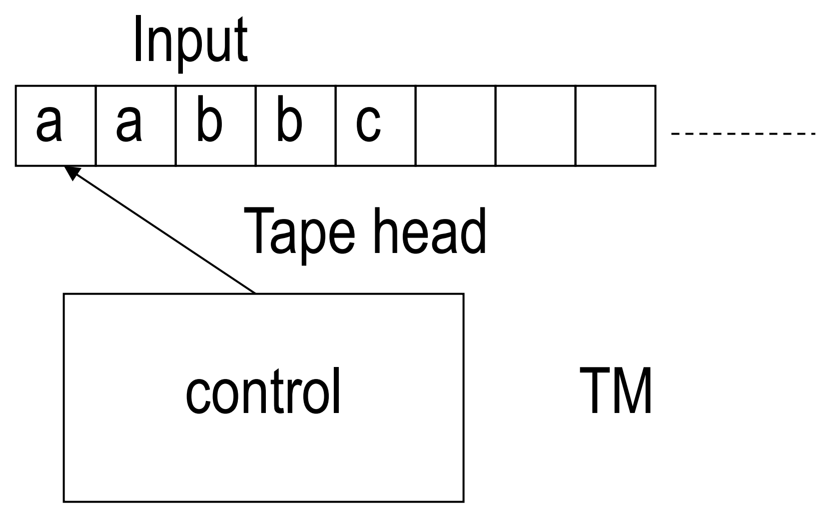
$q_{0\#}$ state: crossed off a 0, read a #

q state: normal state

M1 computation for 011000#011000

X 1 1 0 0 0 # X 1 1 0 0 0

$\uparrow_{\leftarrow,\#}$



q state: normal state

$q_{\#}$ state: read a #

M1 computation for 011000#011000

X 1 1 0 0 0 # X 1 1 0 0 0

$\uparrow_{\leftarrow,\#}$

Under this state, if read one X, stop

$q_{\#}$ state: read a #

M1 computation for 011000#011000

X 1 1 0 0 0 # X 1 1 0 0 0

↑

Keep doing the first step, read the input

$q_{\#}$ state: read a #

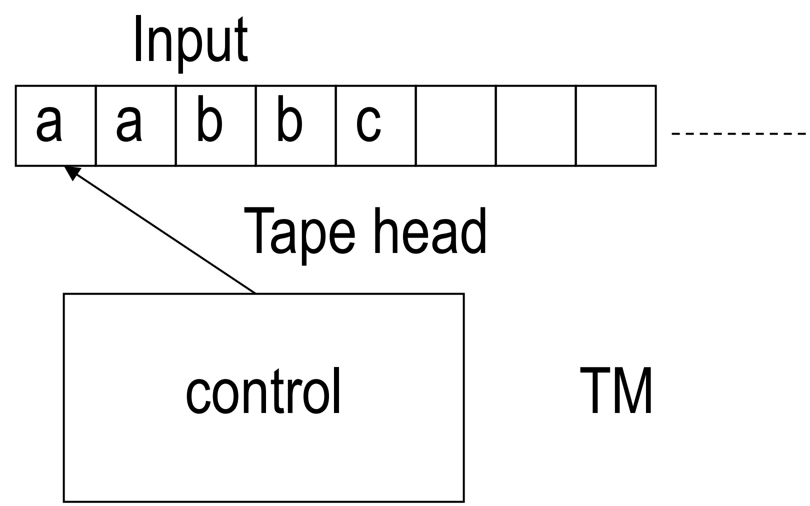
q state: normal state

M1 computation for 011000#011000

X X 1 0 0 0 # X 1 1 0 0 0

$\uparrow_1$

Keep doing the first step, read the input



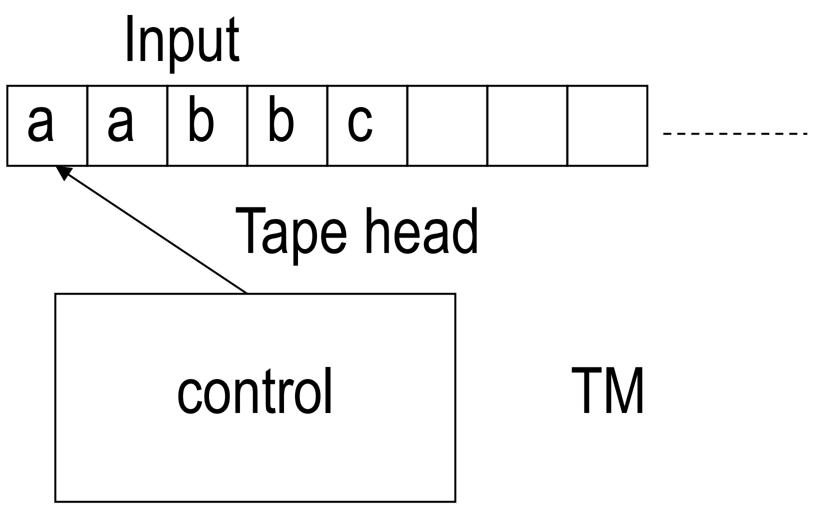
q state: normal state

q_1 state: crossed off a 1

M1 computation for 011000#011000

X X 1 0 0 0 # X 1 1 0 0 0

$\uparrow_{1}$

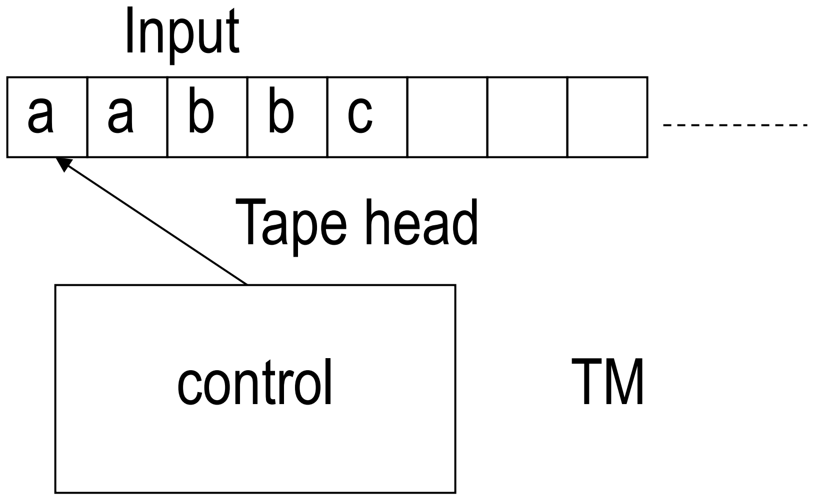


q_1 state: crossed off a 1

M1 computation for 011000#011000

X X 1 0 0 0 # X 1 1 0 0 0

$\uparrow_1$

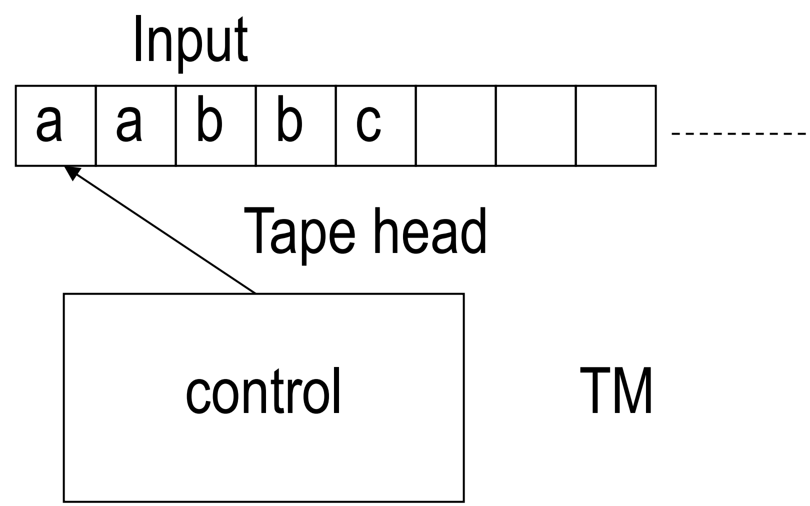


q_1 state: crossed off a 1

M1 computation for 011000#011000

X X 1 0 0 0 # X 1 1 0 0 0

↑ 1,#



q_1 state: crossed off a 1

$q_{1\#}$ state: crossed off a 1, read a #

M1 computation for 011000#011000

X X 1 0 0 0 # X 1 1 0 0 0

$\uparrow_{\mathbf{1,\#}}$

$q_{1\#}$ state: crossed off a 1, read a #

M1 computation for 011000#011000

X X 1 0 0 0 # X X 1 0 0 0

↑←

Under this state, if read one 1, cross off the 1, then move to the left

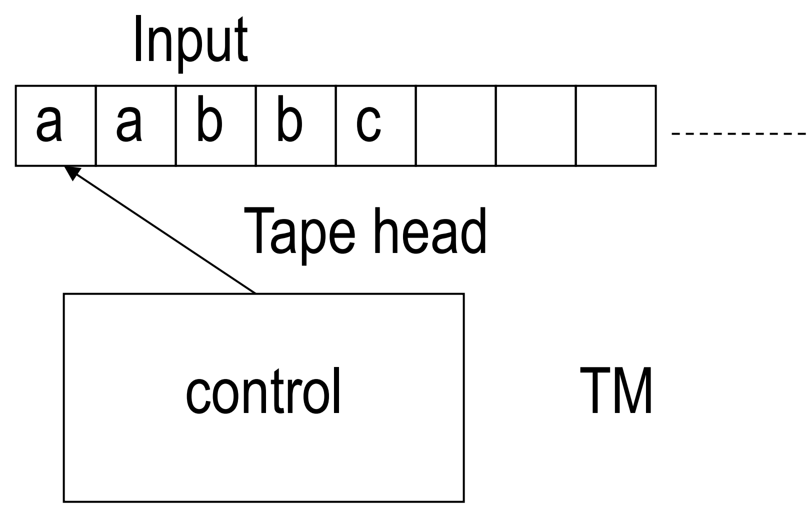
$q_{1\#}$ state: crossed off a 1, read a #

q state: normal state

M1 computation for 011000#011000

X X 1 0 0 0 # X X 1 0 0 0

↑ $_{\leftarrow,\#}$



q state: normal state

$q_{\#}$ state: read a #

M1 computation for 011000#011000

X X 1 0 0 0 # X X 1 0 0 0

$\uparrow_{\leftarrow,\#}$

Under this state, if read one X, stop

$q_{\#}$ state: read a #

M1 computation for 011000#011000

X X 1 0 0 0 # X X 1 0 0 0

↑

Keep doing the first step, read the input

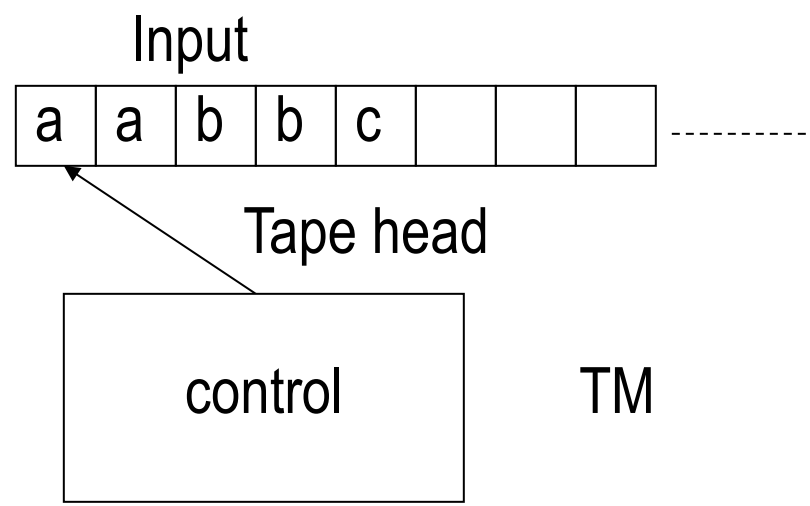
$q_{\#}$ state: read a #

q state: normal state

M1 computation for 011000#011000

X X X 0 0 0 # X X 1 0 0 0

↑ **1**



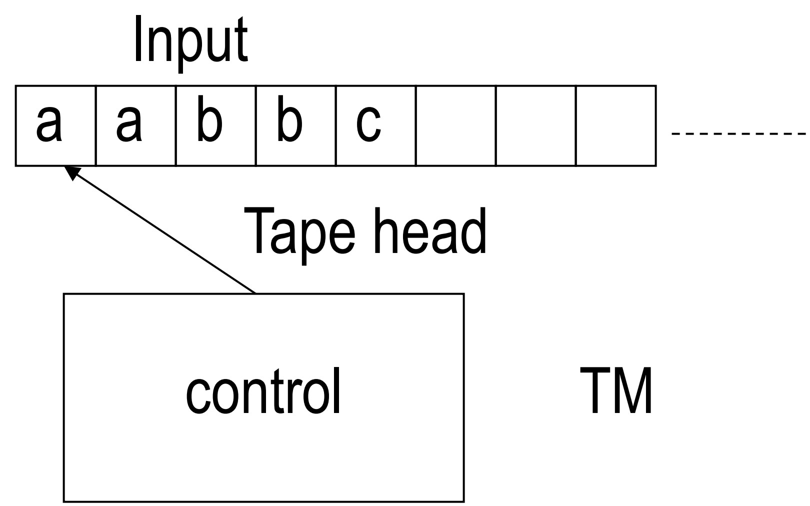
q state: normal state

q_1 state: crossed off a 1

M1 computation for 011000#011000

X X X 0 0 0 # X X 1 0 0 0

$\uparrow_{1,\#}$



q_1 state: crossed off a 1

$q_{1\#}$ state: crossed off a 1, read a #

M1 computation for 011000#011000

X X X 0 0 0 # X X 1 0 0 0

$\uparrow_{1,\#}$

Under this state, if read one 1, cross off the 1, then move to the left

$q_{1\#}$ state: crossed off a 1, read a #

M1 computation for 011000#011000

X X X 0 0 0 # X X X 0 0 0

↑←

Under this state, if read one 1, cross off the 1, then move to the left

$q_{1\#}$ state: crossed off a 1, read a #

⇓

q state: normal state

M1 computation for 011000#011000

X X X X X 0 # X X X X X 0

↑

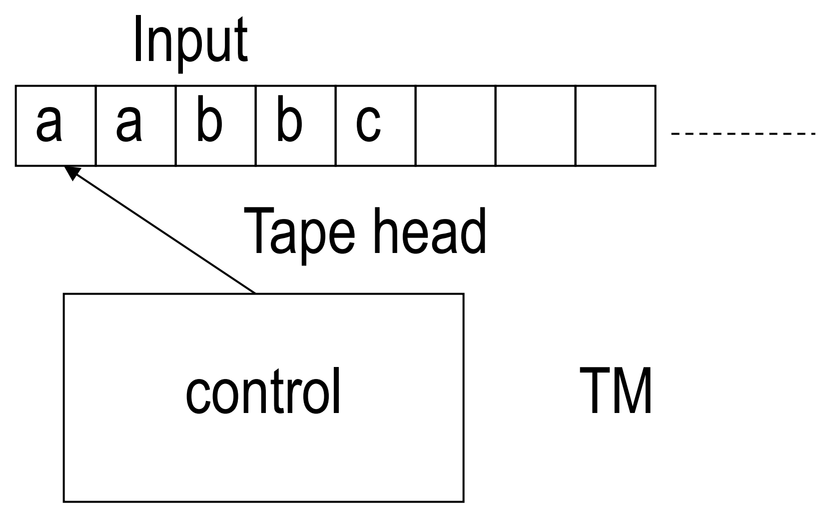
Keep doing the first step, read the input

q state: normal state

M1 computation for 011000#011000

X X X X X X # X X X X X 0

$\uparrow_0$



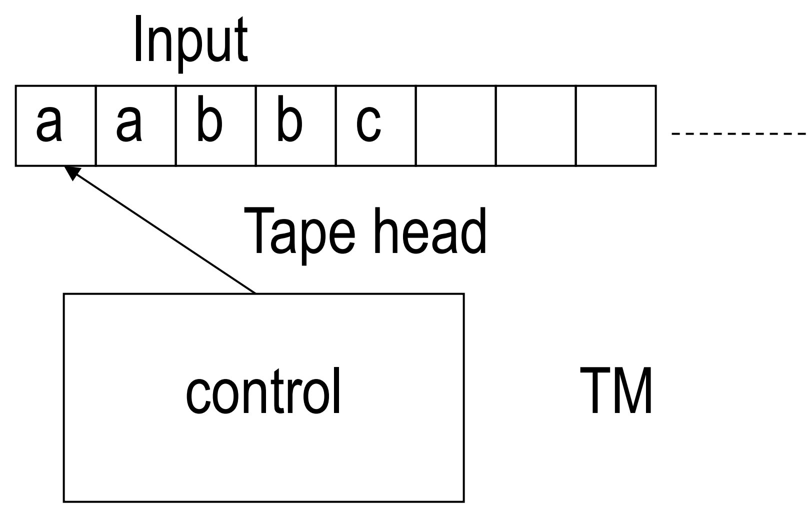
q_0 state: crossed off a 0

M1 computation for 011000#011000

Under this state, if read one 0, cross off the 0, then move to the left

X X X X X X # X X X X X 0

↑ **0,#**



q_0 state: crossed off a 0

$q_{0\#}$ state: crossed off a 0, read a #

M1 computation for 011000#011000

Under this state, if read one 0, cross off the 0, then move to the left

X X X X X X # X X X X X X

↑←

$q_{0\#}$ state: crossed off a 0, read a #

q state: normal state

M1 computation for 011000#011000

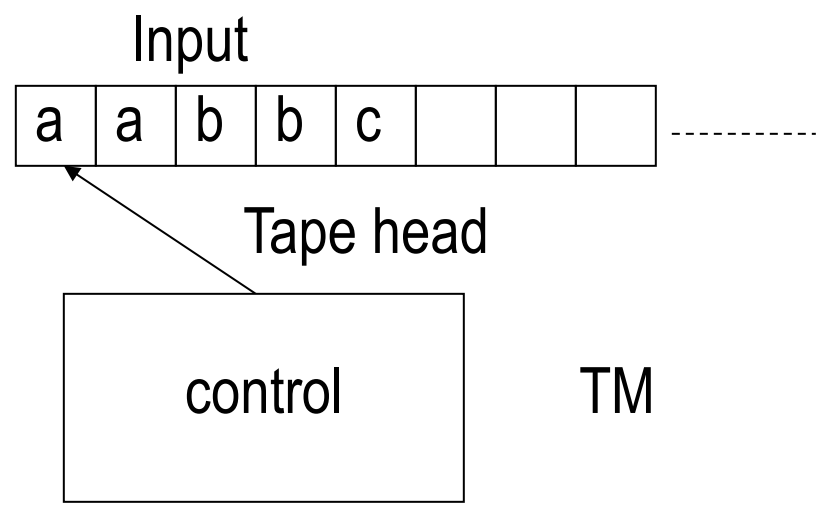
X X X X X X # X X X X X X

$\uparrow_{\leftarrow,\#}$

Under this state, if read one X, stop

q state: normal state

$q_{\#}$ state: read a #



M1 computation for 011000#011000

x x x x x x # x x x x x x

↑

Keep doing the first step, read the input

Currently, there is no 0s or 1s. Change to another state q_B

M1 computation for 011000#011000

X X X X X X # X X X X X X ␣

↑ B

Currently, there is no 0s or 1s. Change to another state q_B and head moves to the last of X

M1 computation for 011000#011000

X X X X X X # X X X X X X ␣

$\uparrow_{\mathbf{B}}$

Go to the last of X
After X is a blank space

Under state q_B, if there is no 0 or 1 after last X, accept; Otherwise, reject.

M1 computation for 011000#011000

X X X X X X # X X X X X X

↑ **accept**

The string on the left and right of # are the same.

Under state q_B, if there is no 0 or 1 after last X, accept; Otherwise, reject.

TM example: $L = \{ w\#w \mid w \in \{0,1\}^* \}$

- M_1= "for input string x":
  1. Scan the input to make sure there exists only one "#", otherwise reject;
  2. Move to the same positions on both sides between "#", check whether there exist same symbols. If not, reject; otherwise, cross off the checked symbols;
  3. If all symbols on the left of "#" are crossed off, check whether there exists other remaining symbols on the right. If yes, reject; otherwise, accept.

Give descriptions of TM that decide the following languages over the alphabet {a,b}.

- {w| w contains an equal number of a and b}
  - 1. Scan the tape and mark the first ‘a’ which has not been marked. If there is no unmarked ‘a’ , go to stage 4. Otherwise, move the head back to the front of the tape.
  - 2. Scan the tape and mark the first ‘b’ which has not been marked. If there is no unmarked ‘b’ , reject.
  - 3. Move the head back to the front of the tape and repeat stage 1.
  - 4. Move the head back to the front of the tape. Scan the tape to see if any unmarked ‘b’s remain. If there are none, accept. Otherwise, reject.

M computation for aabbbbaa

$ a a b b b b a a ␣ ␣

↑

M computation for aabbbbaa

$ x a b b b b a a ␣ ␣

↑

M computation for aabbbbaa

$ x a b b b b a a ␣ ␣

↑

M computation for aabbbbaa

$ x a x b b b a a ␣ ␣

↑

M computation for aabbbbaa

$ x a x b b b a a ␣ ␣

↑

M computation for aabbbbaa

$ x a x b b b a a ␣ ␣

↑

M computation for aabbbbaa

$ x x x b b b a a ␣ ␣

↑

M computation for aabbbbaa

$ x x x b b b a a ␣ ␣

↑

M computation for aabbbbaa

$ x x x x b b a a ␣ ␣

↑

M computation for aabbbbaa

$ x x x x b b a a ␣ ␣

↑

M computation for aabbbbaa

$ x x x x b b a a ␣ ␣

M computation for aabbbbaa

$ x x x x b b a a ␣ ␣

↑

M computation for aabbbbaa

$ x x x x b b x a ␣ ␣

↑

M computation for aabbbbaa

$ x x x x b b x a ␣ ␣

M computation for aabbbbaa

$ x x x x b b x a ␣ ␣

M computation for aabbbbaa

$ x x x x b b x a ␣ ␣

M computation for aabbbbaa

$ x x x x x b x a ␣ ␣

↑

M computation for aabbbbaa

$ x x x x x b x a ␣ ␣

↑

M computation for aabbbbaa

$ x x x x x b x a ␣ ␣

↑

M computation for aabbbbaa

$ x x x x x b x x ␣ ␣

↑

M computation for aabbbbaa

$ x x x x x b x x ␣ ␣

↑

M computation for aabbbbaa

$ x x x x x b x x ␣ ␣

↑

M computation for aabbbbaa

$ x x x x x x x x ␣ ␣

↑

M computation for aabbbbaa

$ x x x x x x x x ␣ ␣

↑

M computation for aabbbbaa

$ x x x x x x x x ␣ ␣

↑

M computation for aabbbbaa

$ x x x x x x x x ␣ ␣

↑

M computation for aabbbbaa

$ a a b b b b a a ␣ ␣

↑

q_0

q0 means start state

M computation for aabbbbaa

$\$ \; x \; a \; b \; b \; b \; b \; a \; a \; ␣ \; ␣$

$\uparrow$

q_0

q_1

q1 means a is marked off

M computation for aabbbbaa

$ x a b b b b a a ␣ ␣

↑

q_0

q_1 q_1 q_1

M computation for aabbbbaa

$ x a x b b b a a ␣ ␣

↑

q_0

q_1 q_1 q_1

q_3

q3 means b is marked off

M computation for aabbbbaa

$ x a x b b b a a ␣ ␣

↑

q_0

q_1 q_1 q_1

q_3 q_3 q_3

M computation for aabbbbaa

$ x a x b b b a a ␣ ␣

↑

q_0

$q_1 \quad q_1 \quad q_1$

$q_3 \quad q_3 \quad q_3 \quad q_3$

q_0

M computation for aabbbbaa

$ x a x b b b a a ␣ ␣

↑

q_0

q_1 q_1 q_1

q_3 q_3 q_3 q_3

q_0 q_0 q_0

M computation for aabbbbaa

$ x x x b b b a a ␣ ␣

↑

q_0

q_1 q_1 q_1

q_3 q_3 q_3 q_3

q_0 q_0 q_0

q_1

q1 means a is marked off

M computation for aabbbbaa

$ x x x b b b a a ␣ ␣

↑

q_0

q_1 q_1 q_1

q_3 q_3 q_3 q_3

q_0 q_0 q_0

q_1 q_1 q_1

M computation for aabbbbaa

\$ x x x b b b a a ␣ ␣

↑

q_0

q_1 q_1 q_1

q_3 q_3 q_3 q_3

q_0 q_0 q_0

q_1 q_1 q_1

q_1 q_1 q_1 q_1

M computation for aabbbbaa

$ x x x x b b a a ␣ ␣

↑

q_0

$q_1 \quad q_1 \quad q_1$

$q_3 \quad q_3 \quad q_3 \quad q_3$

$q_0 \quad q_0 \quad q_0$

$q_1 \quad q_1 \quad q_1$

$q_1 \quad q_1 \quad q_1 \quad q_1$

q_3

q3 means b is marked off

M computation for aabbbbaa

$ x x x x b b a a ␣ ␣

↑

q_0

$q_1 \quad q_1 \quad q_1$

$q_3 \quad q_3 \quad q_3 \quad q_3$

$q_0 \quad q_0 \quad q_0$

$q_1 \quad q_1 \quad q_1$

$q_1 \quad q_1 \quad q_1 \quad q_1$

$q_3 \quad q_3 \quad q_3 \quad q_3$

q_0

M computation for aabbbbaa

$ x x x x x x x x ␣ ␣

↑

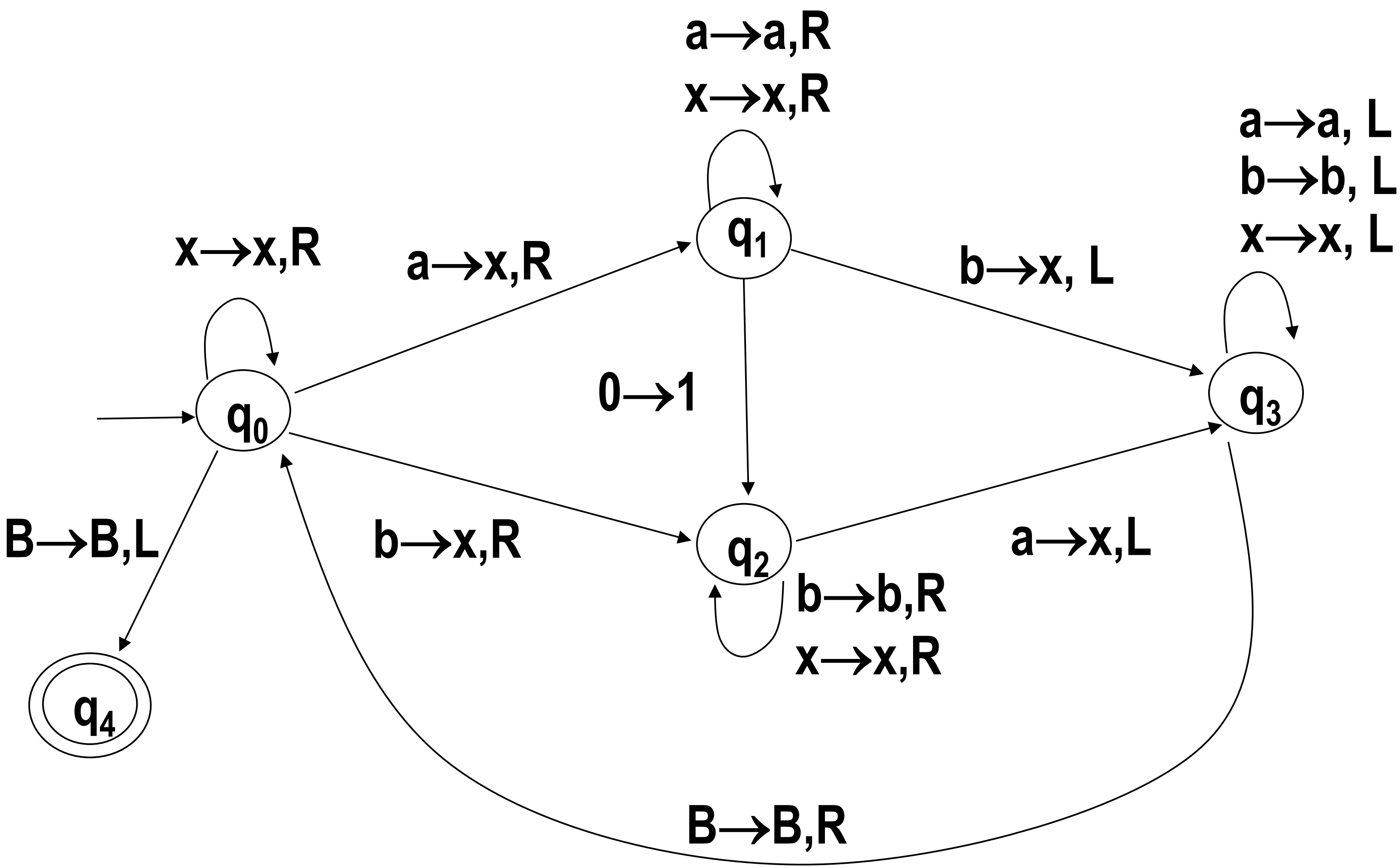
...

q_0 q_0 q_0 q_0 q_0 q_0 q_0 q_0 q_0

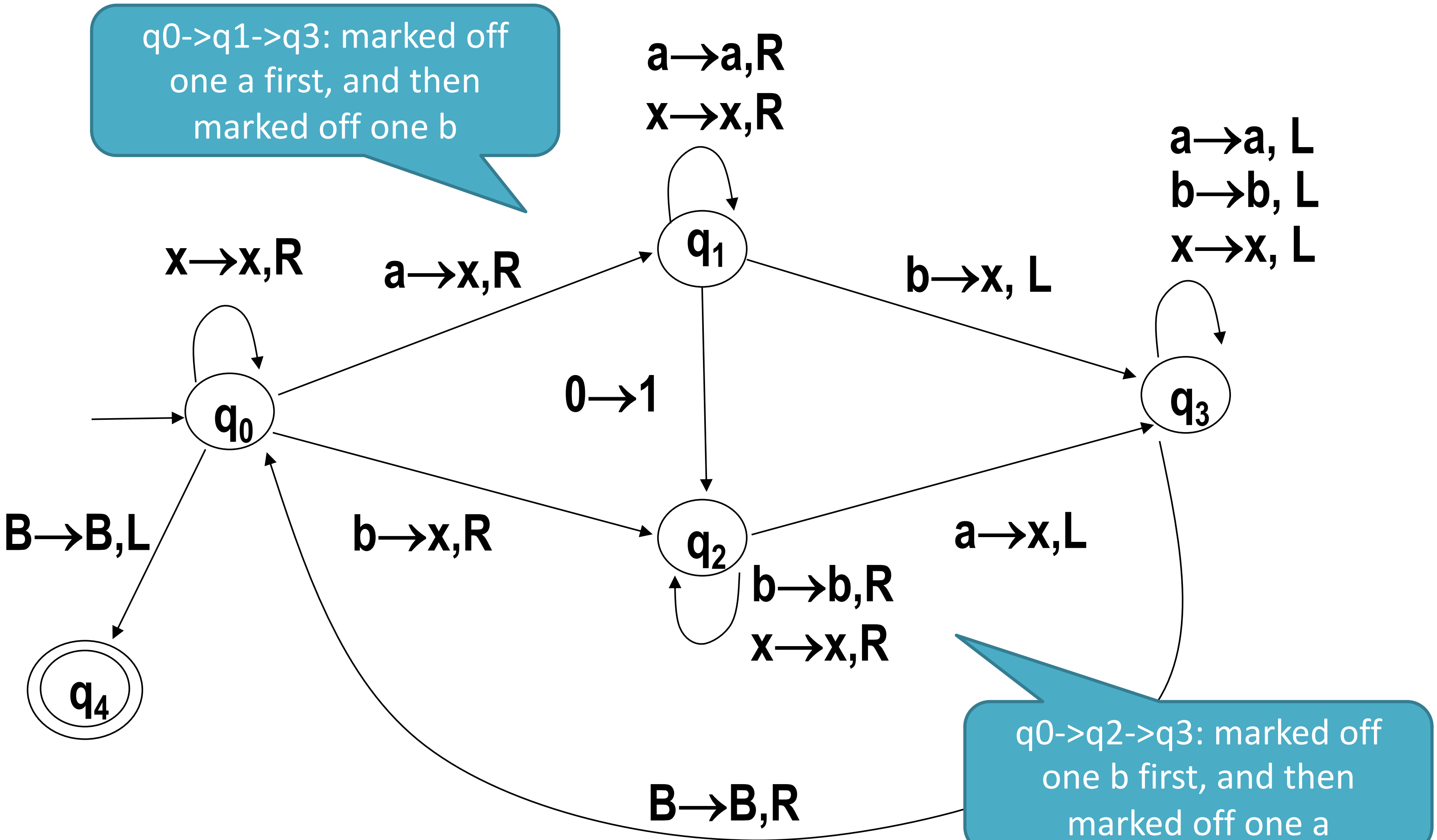
q_4

q4 means accept the input

M computation for aabbbbaa



M computation for aabbbbaa



M computation for aabbbbaa

- TM $M=(Q,\Sigma,\Gamma,\delta,q_0,q_{acc},q_{rej})$

1) Q =
2) Σ =
3) Γ =
4) δ:
5) q_0
6) q_{acc} =
7) q_{rej} =