Lesson 5. Turing machines

CSIE 3110 – Formal Languages and Automata Theory

Tony Tan
Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Table of contents

Table of contents

Turing machines and algorithms

In computer science we are often interested in designing algorithms for some computational problems.

Turing machines and algorithms

In computer science we are often interested in designing algorithms for some computational problems.

For example:

Graph-reachability

Input: A graph $G = (V, E)$ and two vertices $s, t \in V$.
Task: Output `True`, if there is a path from s to t in the graph G. Otherwise, output `False`.

Turing machines and algorithms

In computer science we are often interested in designing algorithms for some computational problems.

For example:

Graph-reachability

Input: A graph $G = (V, E)$ and two vertices $s, t \in V$.
Task: Output `True`, if there is a path from s to t in the graph G. Otherwise, output `False`.

For such problem, we may write a piece of program in some well-known programming languages such as C++ or psuedo-codes in some acceptable format.

Turing machines and algorithms

In computer science we are often interested in designing algorithms for some computational problems.

For example:

Graph-reachability

Input: A graph $G = (V, E)$ and two vertices $s, t \in V$.
Task: Output `True`, if there is a path from s to t in the graph G. Otherwise, output `False`.

For such problem, we may write a piece of program in some well-known programming languages such as C++ or psuedo-codes in some acceptable format.

We can rightly call such codes/pseudo-codes "algorithms."

Turing machines and algorithms – continued

(Question) Is there a precise mathematical definition of "algorithms"?

Turing machines and algorithms – continued

(Question) Is there a precise mathematical definition of "algorithms"?

- The definition should not depend on any programming languages.
  Such languages are ever changing and depend on their "design."
- It must not be ambiguous.
  Pseudo-codes are often "ambiguous" on what constitutes "basic instructions." For example, "sorting" may/may not be regarded as a basic instruction.

Church-Turing thesis

It is universally accepted that Turing machine is a mathematical definition of "algorithm."

Church-Turing thesis

It is universally accepted that Turing machine is a mathematical definition of "algorithm."

Church-Turing thesis
Every "algorithm" is equivalent to a Turing machine.

Church-Turing thesis

It is universally accepted that Turing machine is a mathematical definition of "algorithm."

Church-Turing thesis

Every "algorithm" is equivalent to a Turing machine.

(Note) Turing machine is not the only mathematical definition of algorithm. There are other models such as λ-calculus, recursive functions, etc, but they are all equivalent.

The formal definition of Turing machines

We reserve a special symbol $\sqcup$ called *the blank symbol* and $\triangleleft$ called *the left-end symbol*.

(Def.) A *Turing machine* (TM) is a system $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathrm{acc}}, q_{\mathrm{rej}}, \delta \rangle$:

- Σ is a finite alphabet, called *the input alphabet*, where $\sqcup, \triangleleft \notin \Sigma$.
- Γ is a finite alphabet, called *the tape alphabet*, where $\Sigma \subseteq \Gamma$ and $\sqcup, \triangleleft \in \Gamma$.
- Q is a finite set of states and $q_0 \in Q$ is the initial state.
- $q_{\mathrm{acc}}, q_{\mathrm{rej}} \in Q$ are two special states called the *accept* and *reject* states, respectively.
- $\delta : Q - \{q_{\mathrm{acc}}, q_{\mathrm{rej}}\} \times \Gamma \to Q \times \Gamma \times \{\mathtt{Left}, \mathtt{Right}\}$ is the transition function, whose elements are written in the form:

$$(p, a) \;\to\; (q, b, \alpha)$$

where $p, q \in Q$, $a, b \in \Gamma$ and $\alpha \in \{\mathtt{Left}, \mathtt{Right}\}$.

What does a Turing machine $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathbf{acc}}, q_{\mathbf{rej}}, \delta \rangle$ have?

Assume $\Sigma = \{0, 1\}$ and $\Gamma = \{\triangleleft, 0, 1, x, \#, \sqcup\}$.

What does a Turing machine $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathbf{acc}}, q_{\mathbf{rej}}, \delta \rangle$ have?

Assume $\Sigma = \{0, 1\}$ and $\Gamma = \{\triangleleft, 0, 1, x, \#, \sqcup\}$.

- The input word is a word over Σ.

What does a Turing machine $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathbf{acc}}, q_{\mathbf{rej}}, \delta \rangle$ have?

Assume $\Sigma = \{0, 1\}$ and $\Gamma = \{\triangleleft, 0, 1, x, \#, \sqcup\}$.

- The input word is a word over Σ.
- It has a finite number of states.

What does a Turing machine $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathbf{acc}}, q_{\mathbf{rej}}, \delta \rangle$ have?

Assume $\Sigma = \{0, 1\}$ and $\Gamma = \{\triangleleft, 0, 1, x, \#, \sqcup\}$.

- The input word is a word over Σ.
- It has a finite number of states.
- It has one tape, which is divided into infinitely many cells.
  Each cell contains exactly one symbol from Γ.

$\triangleleft$	0	x	1	$\#$	1	$\#$	0	x	x	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

What does a Turing machine $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathbf{acc}}, q_{\mathbf{rej}}, \delta \rangle$ have?

Assume $\Sigma = \{0, 1\}$ and $\Gamma = \{\triangleleft, 0, 1, x, \#, \sqcup\}$.

- The input word is a word over Σ.
- It has a finite number of states.
- It has one tape, which is divided into infinitely many cells.
  Each cell contains exactly one symbol from Γ.

$\triangleleft$	0	x	1	$\#$	1	$\#$	0	x	x	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

- It has one head reading one cell at a time.

How does it work?

On input word $w \in \Sigma^*$:

How does it work?

On input word $w \in \Sigma^*$:

- It starts in the initial state q_0.

How does it work?

On input word $w \in \Sigma^*$:

- It starts in the initial state q_0.
- The tape initially contains the input word w:
  For example, if the input word $w = 10010$:

$\triangleleft$	1	0	0	1	0	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

How does it work?

On input word $w \in \Sigma^*$:

- It starts in the initial state q_0.
- The tape initially contains the input word w:
  For example, if the input word $w = 10010$:

$\triangleleft$	1	0	0	1	0	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

- The head initially reads the first symbol of the input word.

How does it work?

On input word $w \in \Sigma^*$:

- It starts in the initial state q_0.
- The tape initially contains the input word w:
  For example, if the input word $w = 10010$:

$\triangleleft$	1	0	0	1	0	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

- The head initially reads the first symbol of the input word.
- It moves from state to state according to the transitions.

  The transitions also specify where the head moves to and what symbol it writes onto the tape.

How does it work?

On input word $w \in \Sigma^*$:

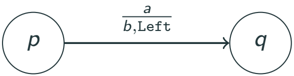
- It starts in the initial state q_0.
- The tape initially contains the input word w:
  For example, if the input word $w = 10010$:

$\triangleleft$	1	0	0	1	0	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$	$\cdots$

- The head initially reads the first symbol of the input word.
- It moves from state to state according to the transitions.

  The transitions also specify where the head moves to and what symbol it writes onto the tape.
- It stops when it is in q_{acc} or q_{rej}.

Illustration of a transition $(p, a) \to (q, b, \texttt{Left})$



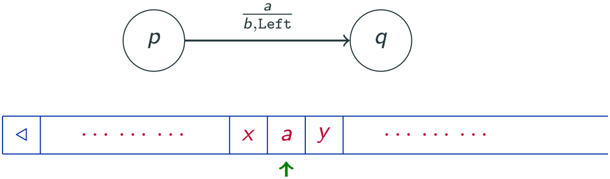
(The intuitive meaning) If:

- the TM is in state p,
- the head is reading symbol a,

then:

- it writes symbol b (on top of a),
- the head moves left,
- the TM enters state q,

Illustration of a transition $(p, a) \rightarrow (q, b, \texttt{Left})$



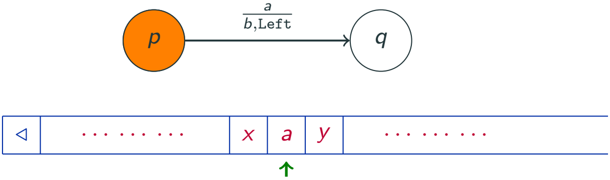
(The intuitive meaning) If:

- the TM is in state p,
- the head is reading symbol a,

then:

- it writes symbol b (on top of a),
- the head moves left,
- the TM enters state q,

Illustration of a transition $(p, a) \to (q, b, \texttt{Left})$



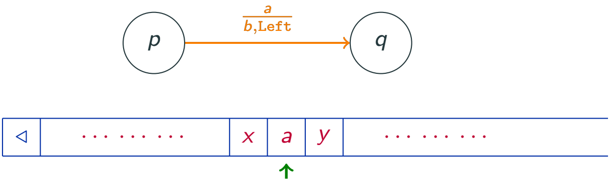
(The intuitive meaning) If:

- the TM is in state p,
- the head is reading symbol a,

then:

- it writes symbol b (on top of a),
- the head moves left,
- the TM enters state q,

Illustration of a transition $(p, a) \rightarrow (q, b, \texttt{Left})$



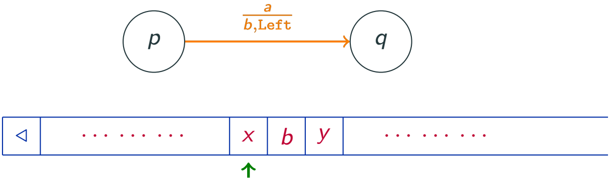
(The intuitive meaning) If:

- the TM is in state p,
- the head is reading symbol a,

then:

- it writes symbol b (on top of a),
- the head moves left,
- the TM enters state q,

Illustration of a transition $(p, a) \to (q, b, \texttt{Left})$



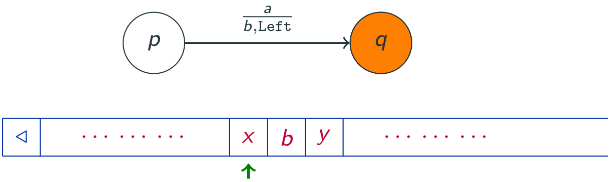
(The intuitive meaning) If:

- the TM is in state p,
- the head is reading symbol a,

then:

- it writes symbol b (on top of a),
- the head moves left,
- the TM enters state q,

Illustration of a transition $(p, a) \to (q, b, \texttt{Left})$



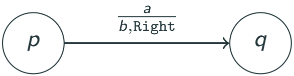
(The intuitive meaning) If:

- the TM is in state p,
- the head is reading symbol a,

then:

- it writes symbol b (on top of a),
- the head moves left,
- the TM enters state q,

Illustration of a transition $(p, a) \to (q, b, \texttt{Right})$



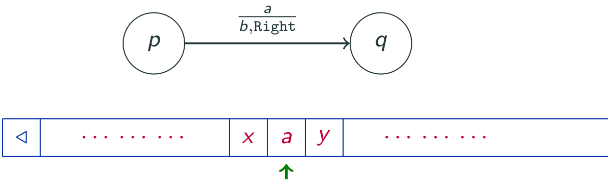
(The intuitive meaning) If:

- the TM is in state p,
- the head is reading symbol a,

then:

- it writes symbol b (on top of a),
- the head moves right,
- the TM enters state q,

Illustration of a transition $(p, a) \to (q, b, \texttt{Right})$



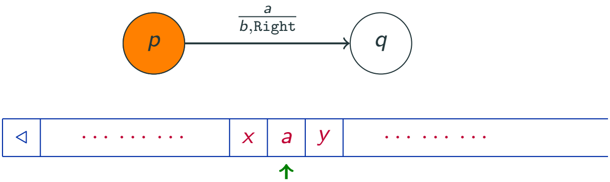
(The intuitive meaning) If:

- the TM is in state p,
- the head is reading symbol a,

then:

- it writes symbol b (on top of a),
- the head moves right,
- the TM enters state q,

Illustration of a transition $(p, a) \to (q, b, \mathtt{Right})$



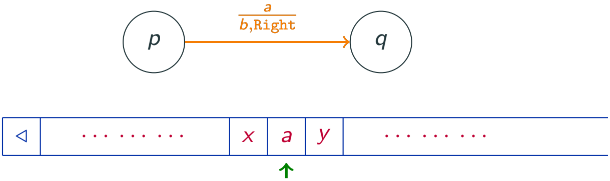
(The intuitive meaning) If:

- the TM is in state p,
- the head is reading symbol a,

then:

- it writes symbol b (on top of a),
- the head moves right,
- the TM enters state q,

Illustration of a transition $(p, a) \to (q, b, \texttt{Right})$



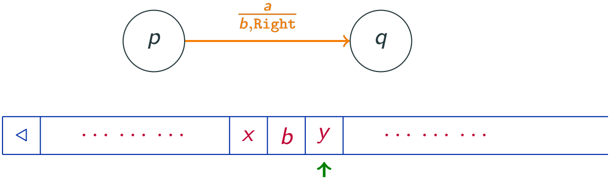
(The intuitive meaning) If:

- the TM is in state p,
- the head is reading symbol a,

then:

- it writes symbol b (on top of a),
- the head moves right,
- the TM enters state q,

Illustration of a transition $(p, a) \to (q, b, \texttt{Right})$



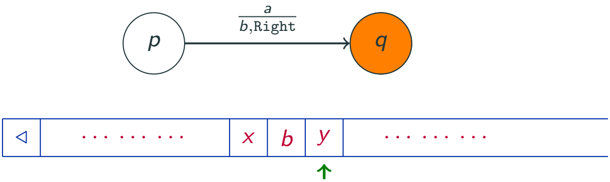
(The intuitive meaning) If:

- the TM is in state p,
- the head is reading symbol a,

then:

- it writes symbol b (on top of a),
- the head moves right,
- the TM enters state q,

Illustration of a transition $(p, a) \to (q, b, \texttt{Right})$



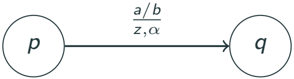
(The intuitive meaning) If:

- the TM is in state p,
- the head is reading symbol a,

then:

- it writes symbol b (on top of a),
- the head moves right,
- the TM enters state q,

Some conventions in depicting transitions – part 1



This means if:

- the TM is in state p,
- the head reads a or b,

then:

- the head writes z,
- moves according to $\alpha \in \{\texttt{Left}, \texttt{Right}\}$,
- the TM enters state q.

Some conventions in depicting transitions – part 2

$$\begin{tikzcd} p \arrow[r, "\frac{a_1/a_2/\cdots/a_k}{z,\alpha}"] & q \end{tikzcd}$$

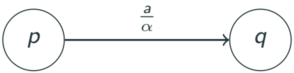
This means if:

- the TM is in state p,
- the head reads a_1 or a_2 or $\ldots$ or a_k,

then:

- the head writes z,
- moves according to $\alpha \in \{\mathtt{Left}, \mathtt{Right}\}$,
- the TM enters state q.

Some conventions in depicting transitions – part 3



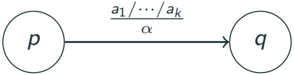
We don't specify the symbol it writes. This means if:

- the TM is in state p,
- the head reads a,

then:

- the head writes a (i.e., the same symbol it reads),
- moves according to $\alpha \in \{\texttt{Left}, \texttt{Right}\}$,
- the TM enters state q.

Some conventions in depicting transitions – part 4



This means if:

- the TM is in state p,
- the head reads a_1 or a_2 or ... or a_k,

then:

- the head writes the same symbol it reads,
- moves according to $\alpha \in \{\texttt{Left}, \texttt{Right}\}$,
- the TM enters state q.

Some conventions in depicting transitions – part 5

q_{rej} is not depicted explicitly in a Turing machine.

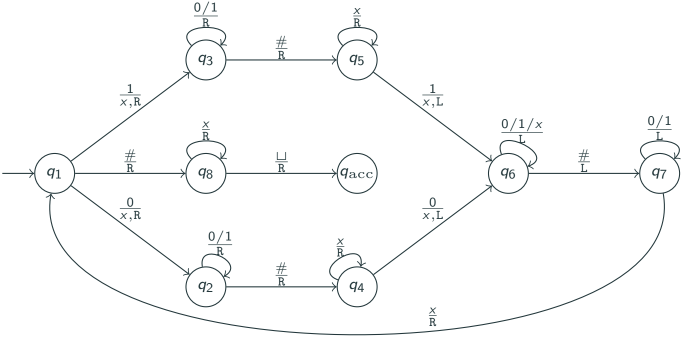
Any transition not depicted is assumed to enter q_{rej}.

We assume that the head always moves right when it reads $\triangleleft$ (the left-end marker).

The moves `Left` and `Right` are often abbreviated as `L` and `R`.

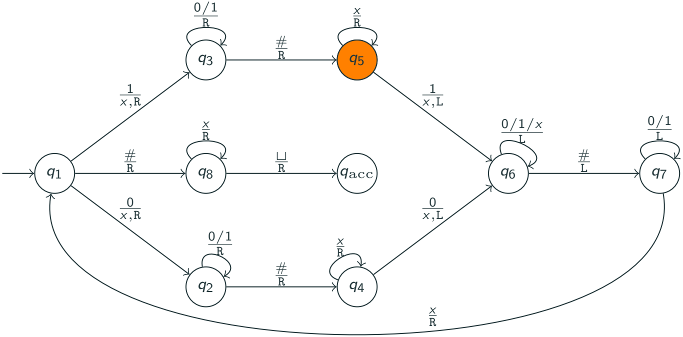
An example of a Turing machine where $\Gamma = \{\triangleleft, 0, 1, x, \#, \sqcup\}$

q_1 is the initial state.



An example of a Turing machine where $\Gamma = \{\triangleleft, 0, 1, x, \#, \sqcup\}$

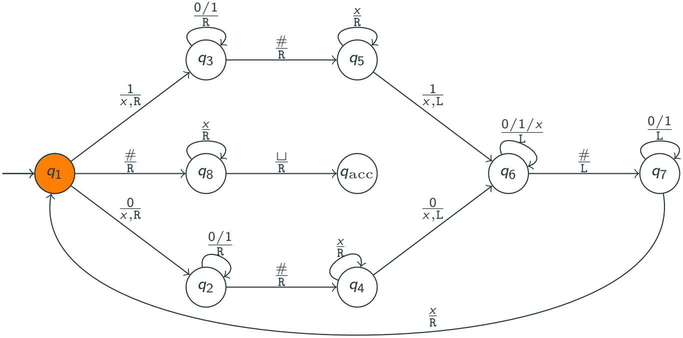
q_1 is the initial state.



Note, for example, we do not depict the transition when it is in state q_5 and the head reads symbol 0.

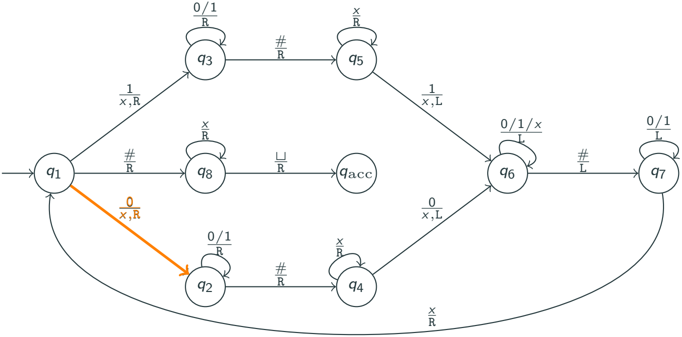
This means the TM enters q_{rej}.

An example of a Turing machine on input 01101#01101



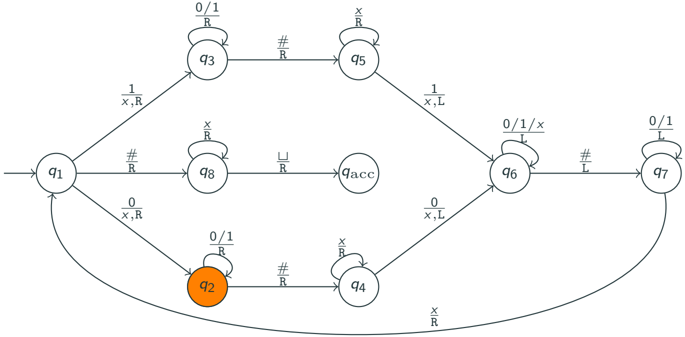
$\triangleleft$	0	1	1	0	1	#	0	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$
	↑													

An example of a Turing machine on input 01101#01101

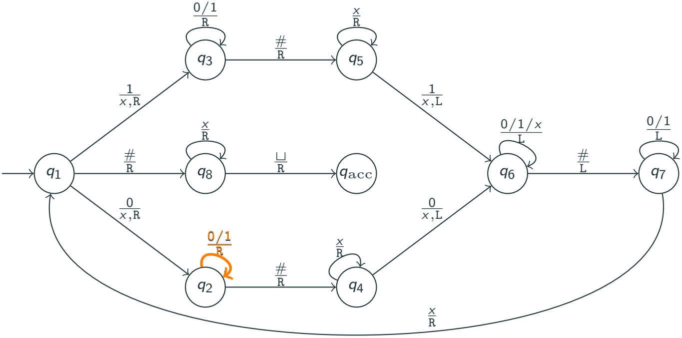


◁	0	1	1	0	1	#	0	1	1	0	1	⊔	⊔	···
	↑													

An example of a Turing machine on input 01101#01101

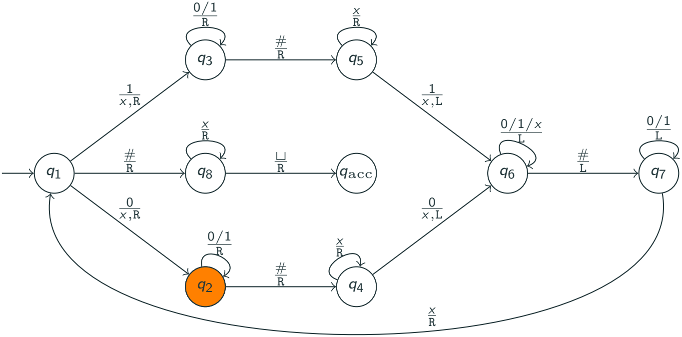


An example of a Turing machine on input 01101#01101



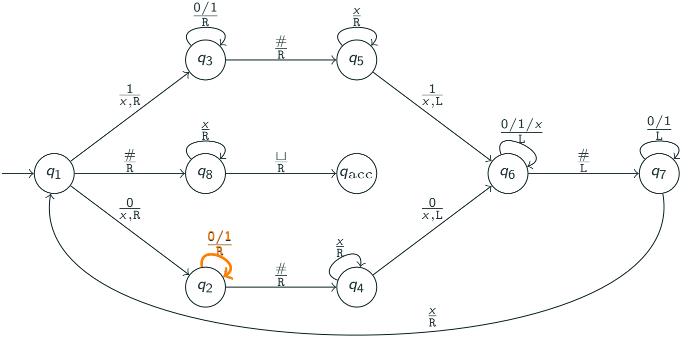
$\triangleleft$	x	1	1	0	1	#	0	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$
		$\uparrow$												

An example of a Turing machine on input 01101#01101



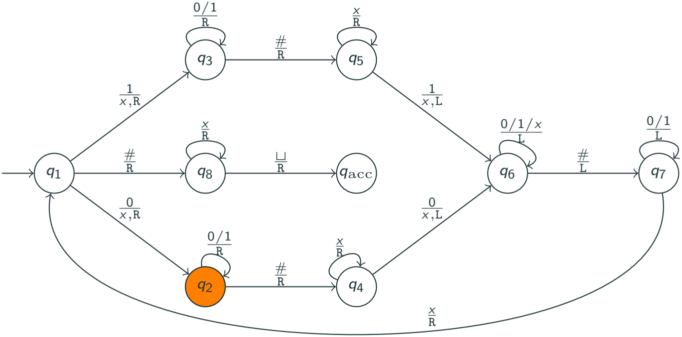
$\triangleleft$	x	1	1	0	1	#	0	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$
			↑											

An example of a Turing machine on input 01101#01101



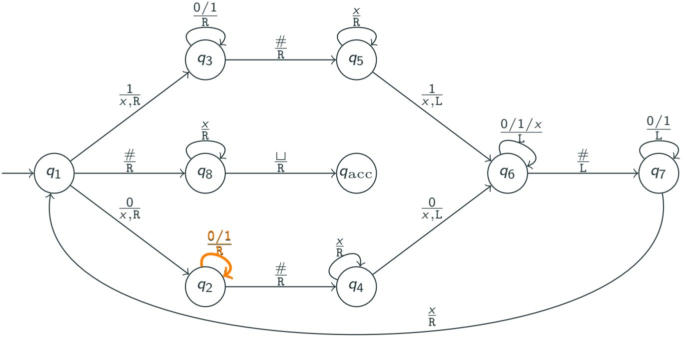
$\triangleleft$	x	1	1	0	1	#	0	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input $01101\#01101$



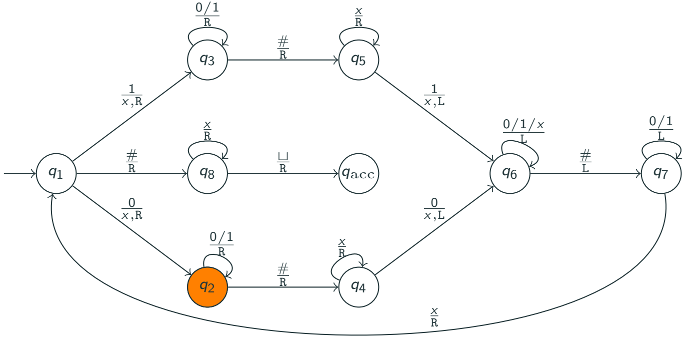
$\triangleleft$	x	1	1	0	1	$\#$	0	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



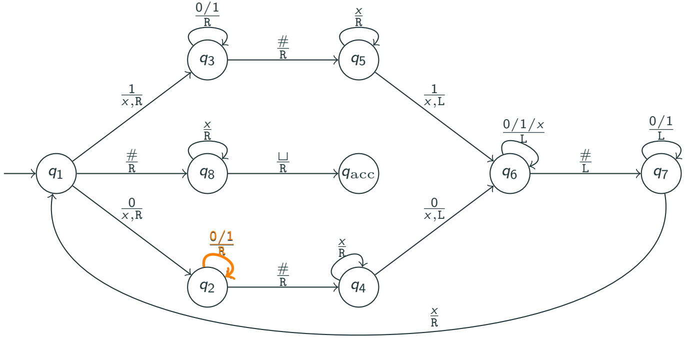
◁	x	1	1	0	1	#	0	1	1	0	1	⊔	⊔	⋯

An example of a Turing machine on input 01101#01101



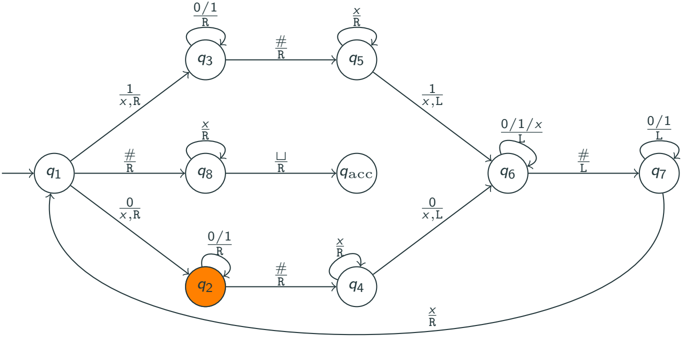
$\triangleleft$	x	1	1	0	1	#	0	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$
					$\uparrow$									

An example of a Turing machine on input 01101#01101



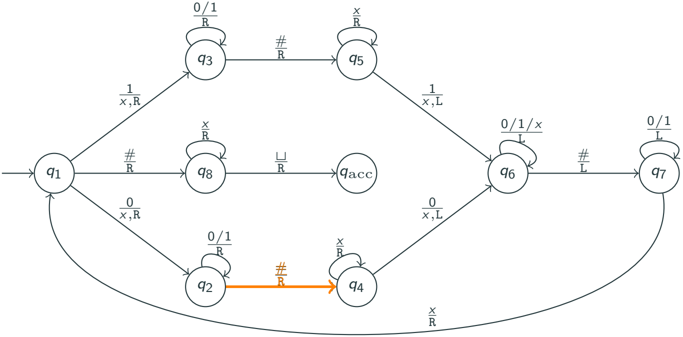
◁	x	1	1	0	1	#	0	1	1	0	1	⊔	⊔	···

An example of a Turing machine on input 01101#01101



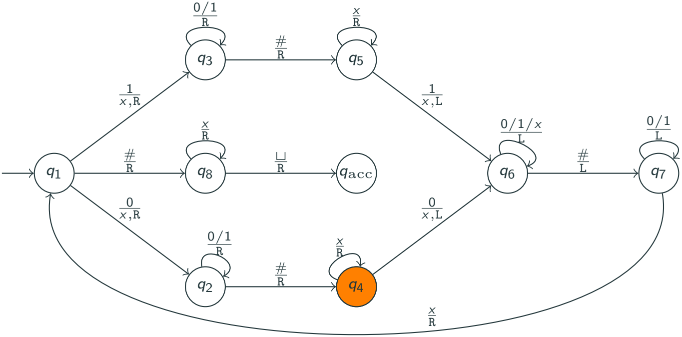
$\triangleleft$	x	1	1	0	1	#	0	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



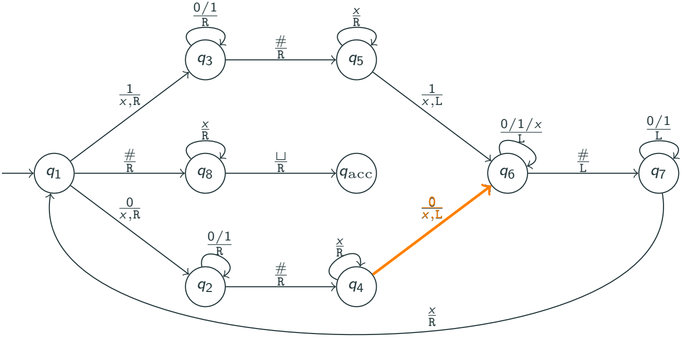
$\triangleleft$	x	1	1	0	1	#	0	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

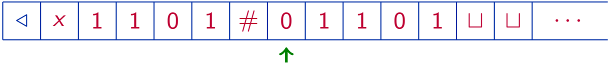
An example of a Turing machine on input $01101\#01101$



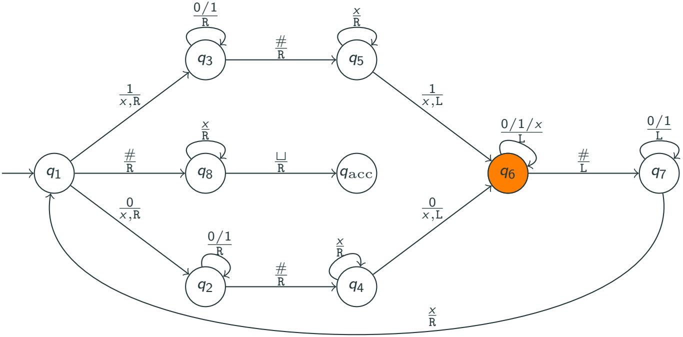
$\triangleleft$	x	1	1	0	1	$\#$	0	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

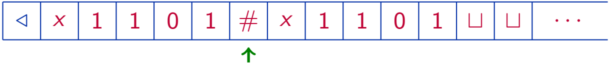
An example of a Turing machine on input 01101#01101



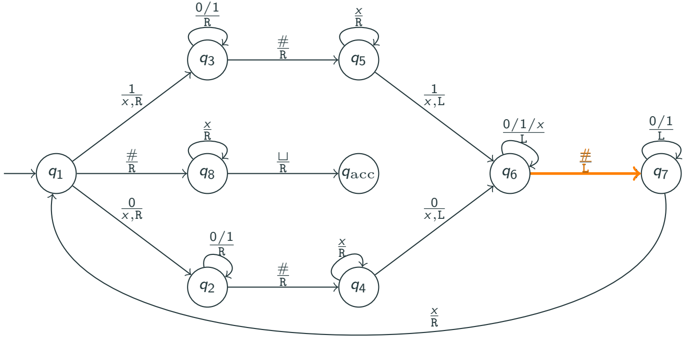


An example of a Turing machine on input 01101#01101



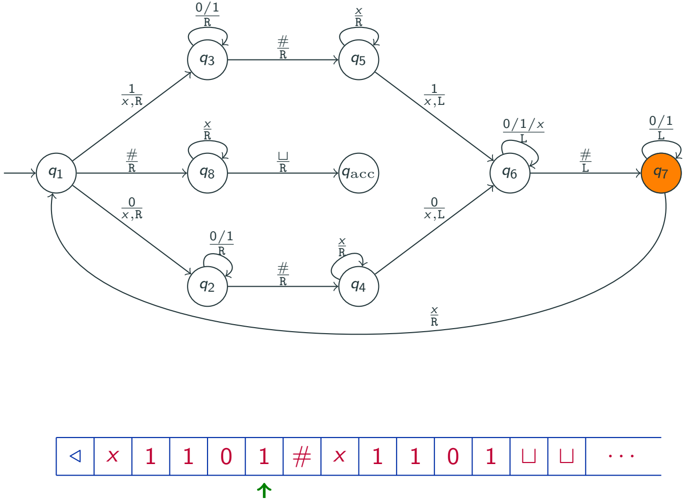


An example of a Turing machine on input 01101#01101

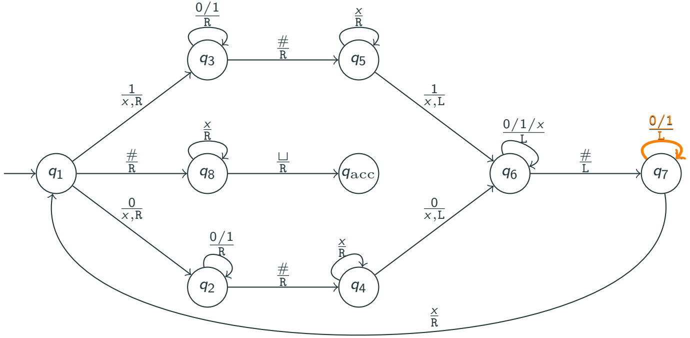


$\triangleleft$	x	1	1	0	1	#	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

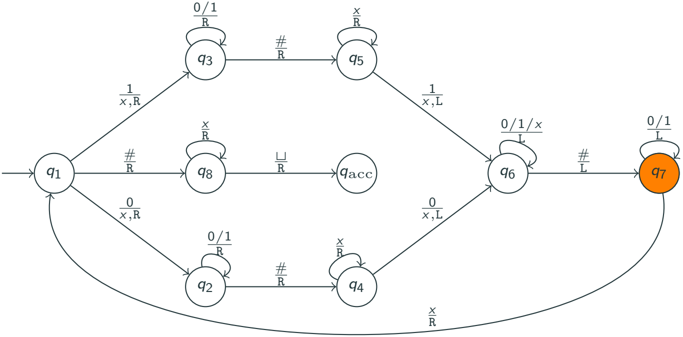


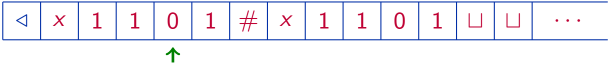
An example of a Turing machine on input 01101#01101



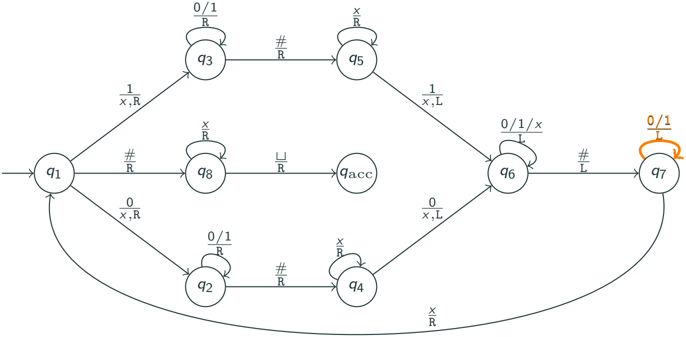
$\triangleleft$	x	1	1	0	1	#	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



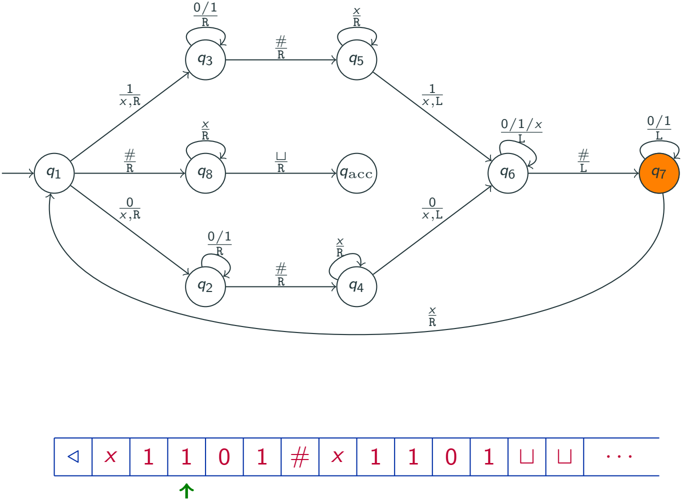


An example of a Turing machine on input 01101#01101

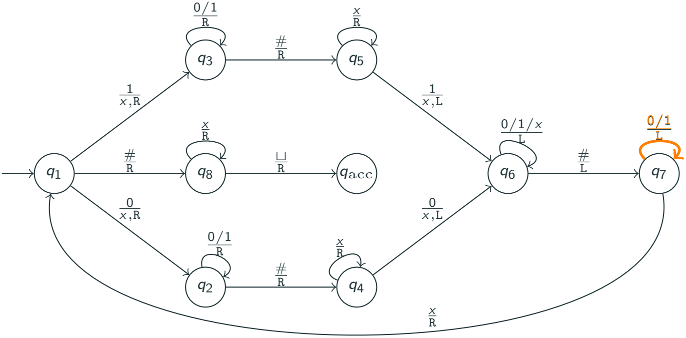


◁	x	1	1	0	1	#	x	1	1	0	1	⊔	⊔	⋯

An example of a Turing machine on input 01101#01101

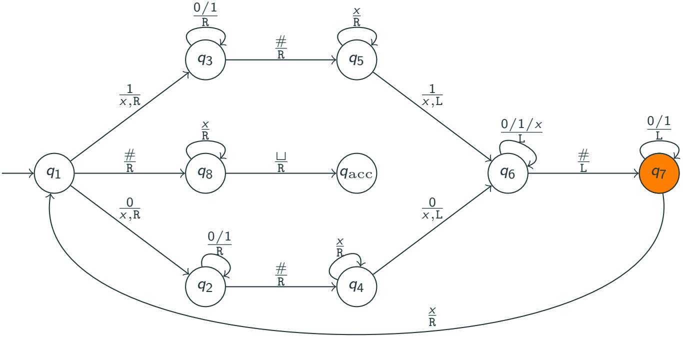


An example of a Turing machine on input 01101#01101



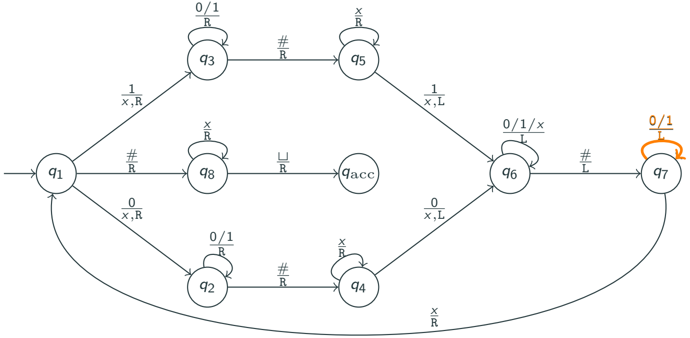
$\triangleleft$	x	1	1	0	1	#	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



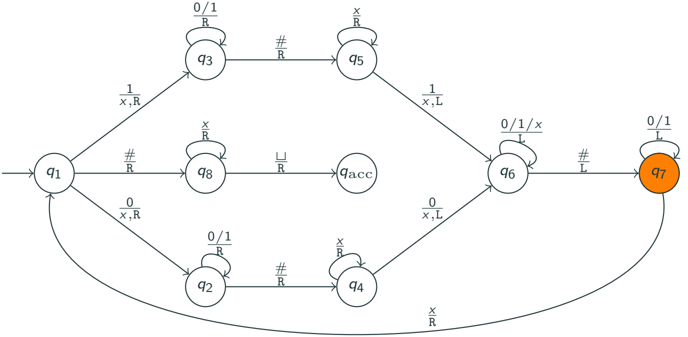
$\triangleleft$	x	1	1	0	1	#	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



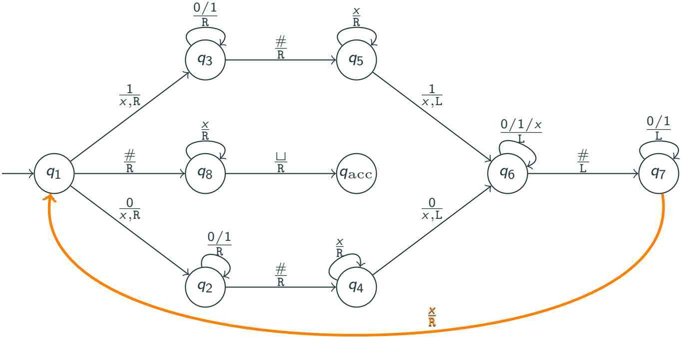
$\triangleleft$	x	1	1	0	1	#	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



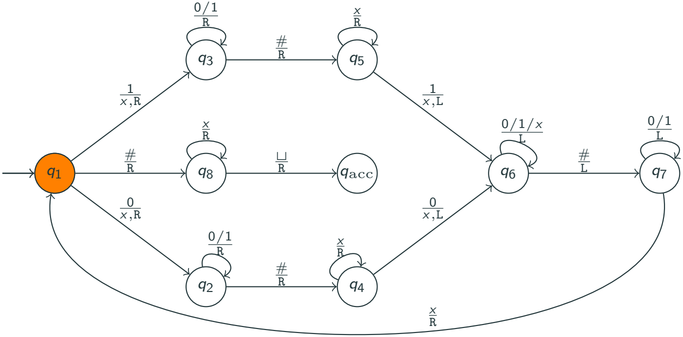
$\triangleleft$	x	1	1	0	1	#	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



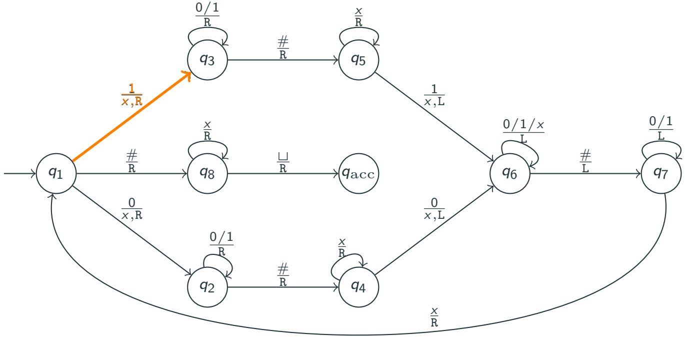
$\triangleleft$	x	1	1	0	1	#	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



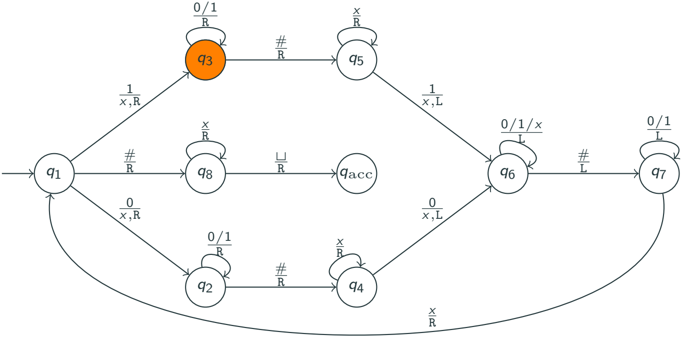
$\triangleleft$	x	1	1	0	1	#	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$
		$\uparrow$												

An example of a Turing machine on input 01101#01101



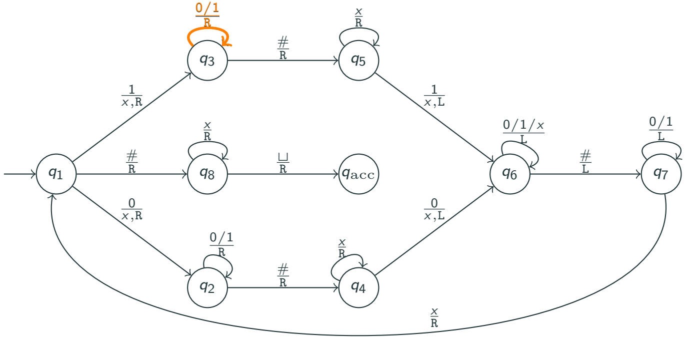
$\triangleleft$	x	1	1	0	1	#	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



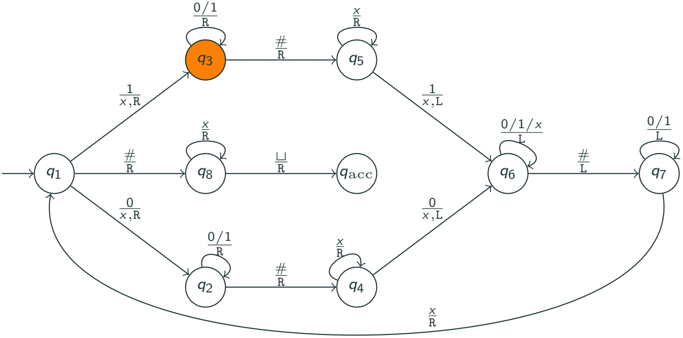
◁	x	x	1	0	1	#	x	1	1	0	1	⊔	⊔	···

An example of a Turing machine on input 01101#01101



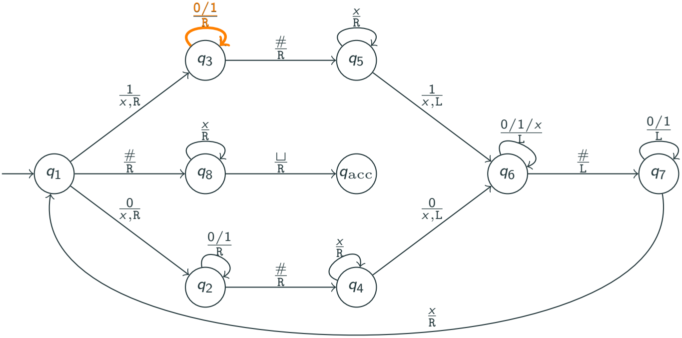
$\triangleleft$	x	x	1	0	1	#	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

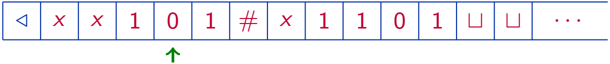
An example of a Turing machine on input 01101#01101



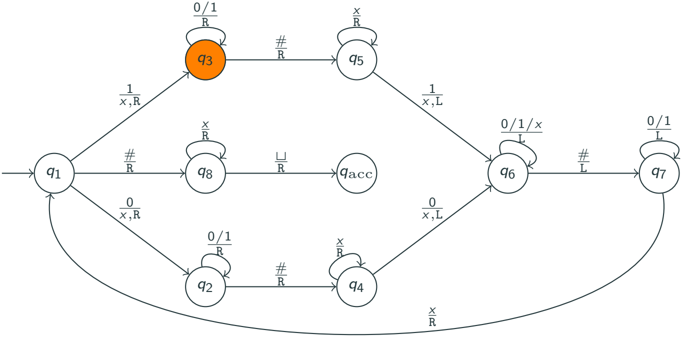
$\triangleleft$	x	x	1	0	1	#	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



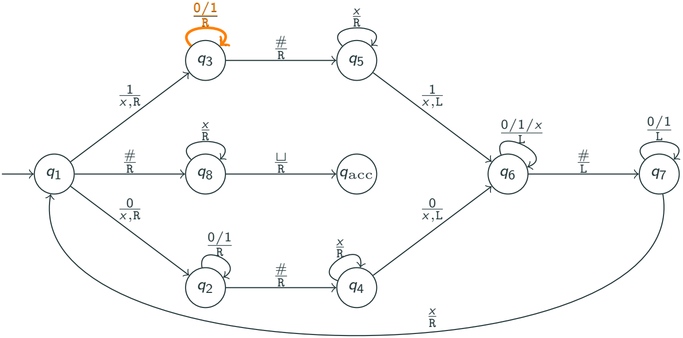


An example of a Turing machine on input 01101#01101



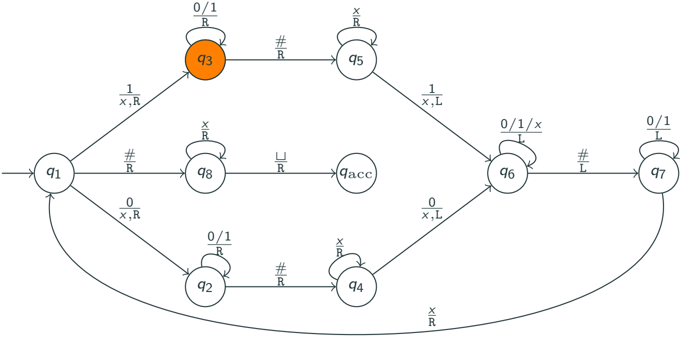
$\triangleleft$	x	x	1	0	1	#	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



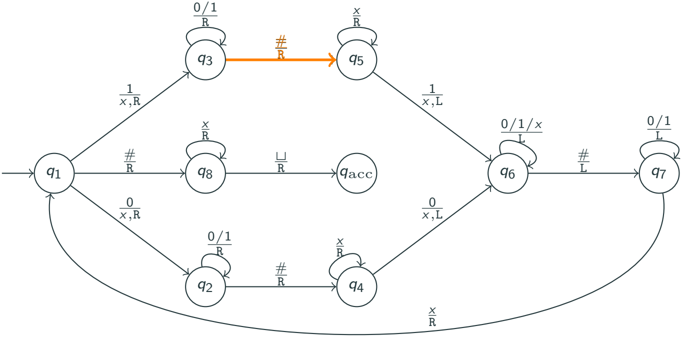
$\triangleleft$	x	x	1	0	1	#	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



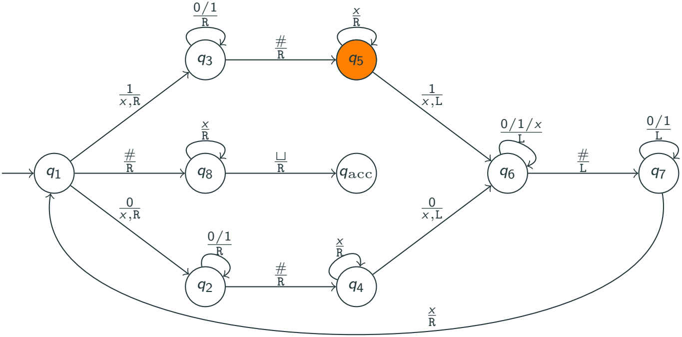
$\triangleleft$	x	x	1	0	1	#	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$
						↑								

An example of a Turing machine on input 01101#01101



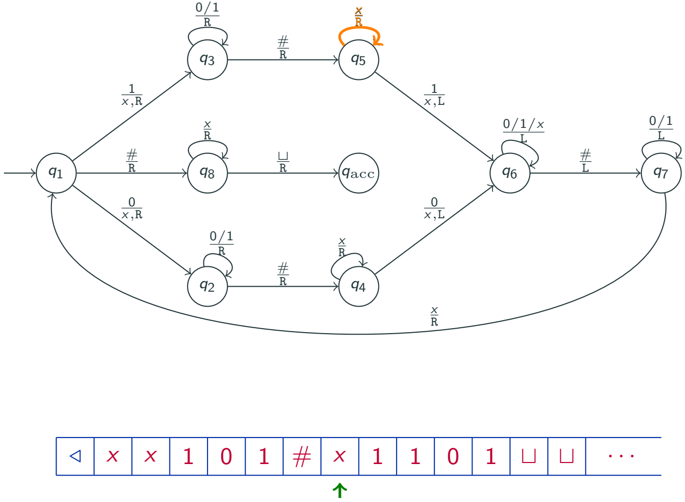
◁	x	x	1	0	1	#	x	1	1	0	1	⊔	⊔	⋯

An example of a Turing machine on input $01101\#01101$

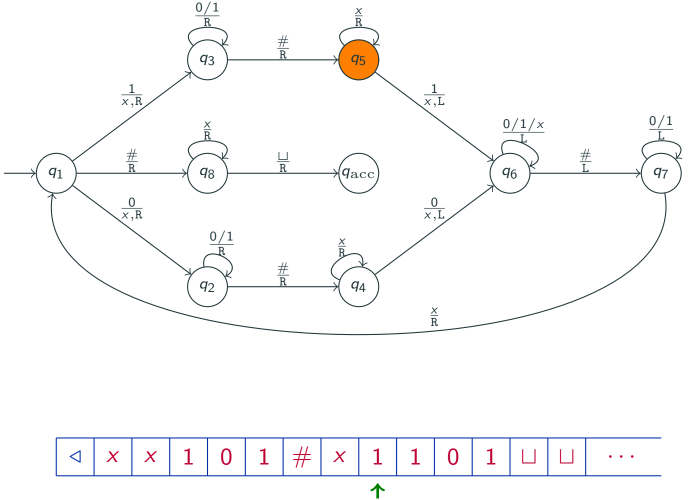


$\triangleleft$	x	x	1	0	1	$\#$	x	1	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

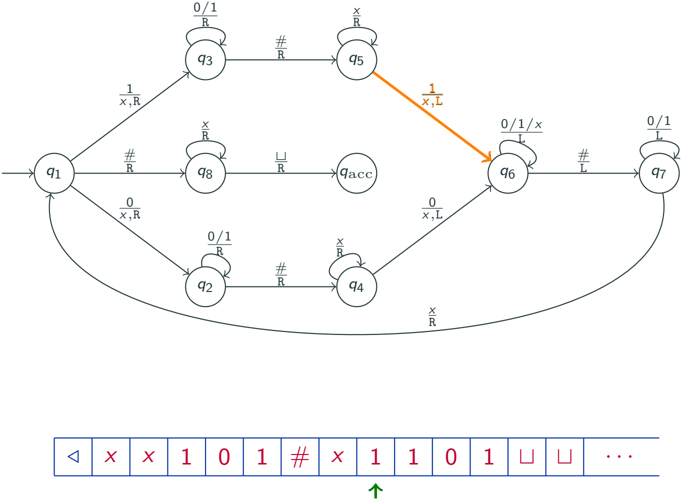
An example of a Turing machine on input 01101#01101



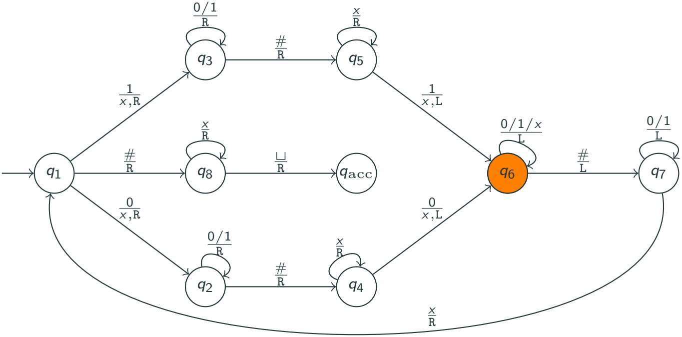
An example of a Turing machine on input 01101#01101



An example of a Turing machine on input 01101#01101

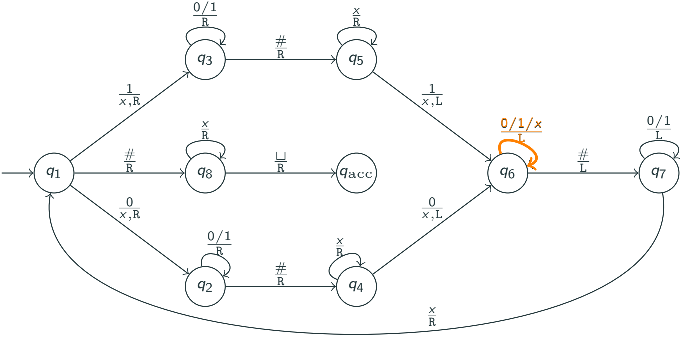


An example of a Turing machine on input 01101#01101



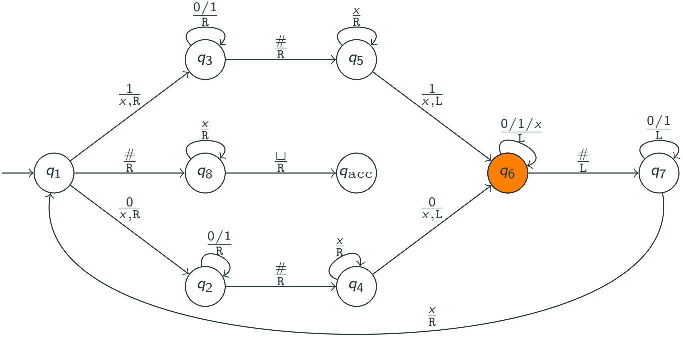
$\triangleleft$	x	x	1	0	1	#	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



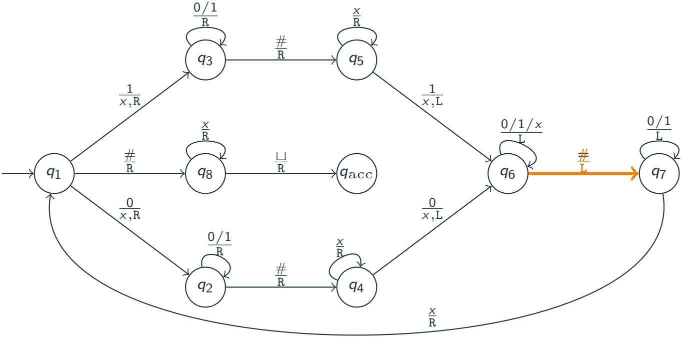
◁	x	x	1	0	1	#	x	x	1	0	1	⊔	⊔	⋯

An example of a Turing machine on input 01101#01101



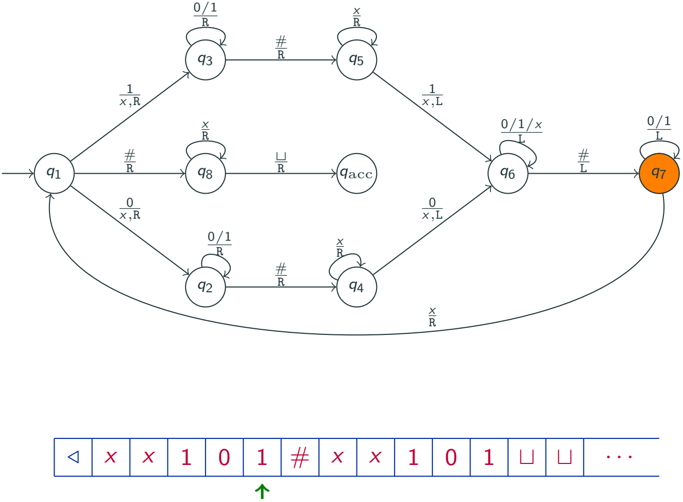
$\triangleleft$	x	x	1	0	1	#	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

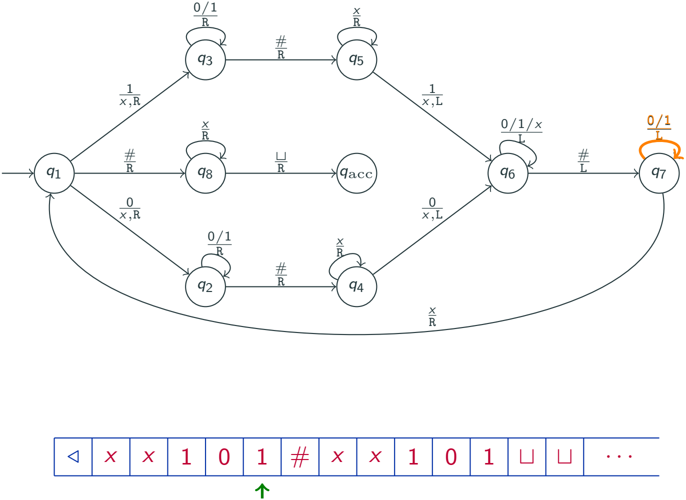


◁	x	x	1	0	1	#	x	x	1	0	1	⊔	⊔	···

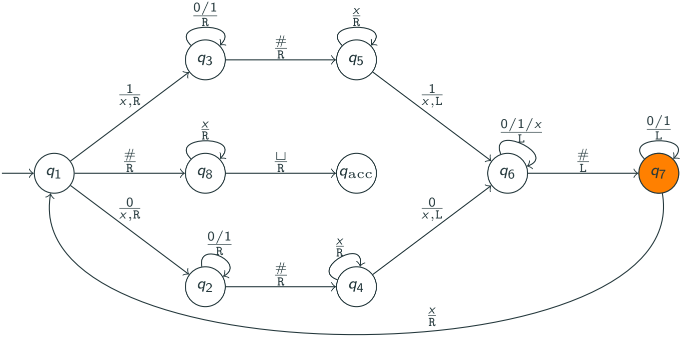
An example of a Turing machine on input 01101#01101



An example of a Turing machine on input 01101#01101

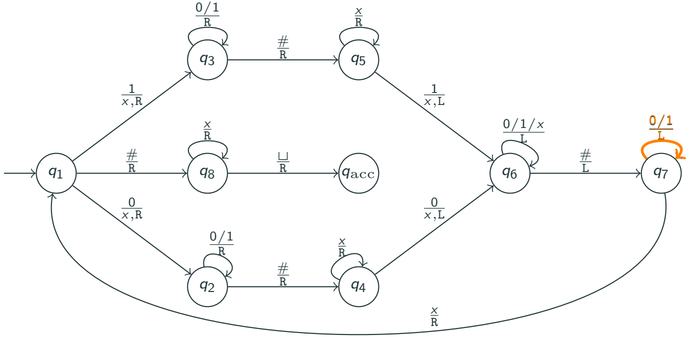


An example of a Turing machine on input 01101#01101



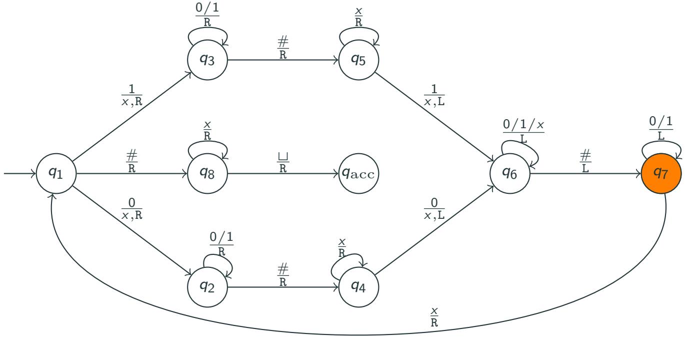
$\triangleleft$	x	x	1	0	1	#	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

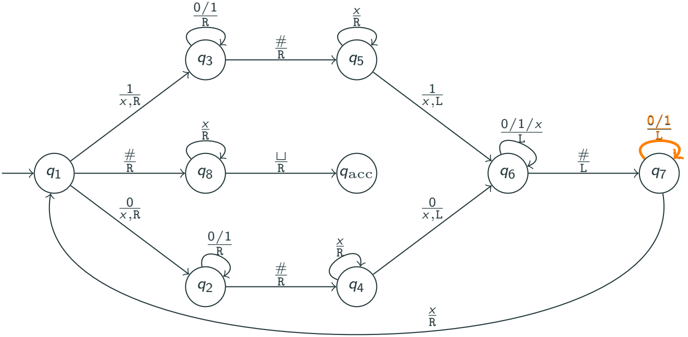


◁	x	x	1	0	1	#	x	x	1	0	1	⊔	⊔	···

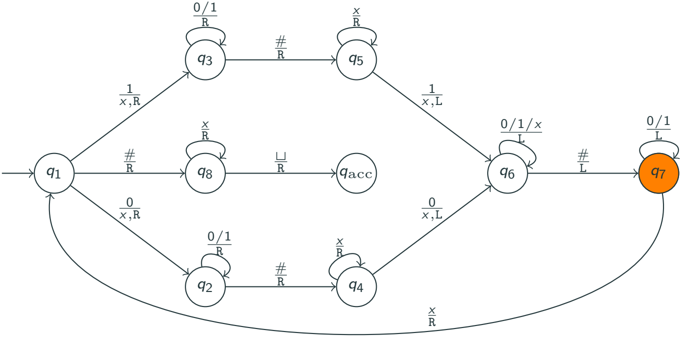
An example of a Turing machine on input 01101#01101



An example of a Turing machine on input 01101#01101

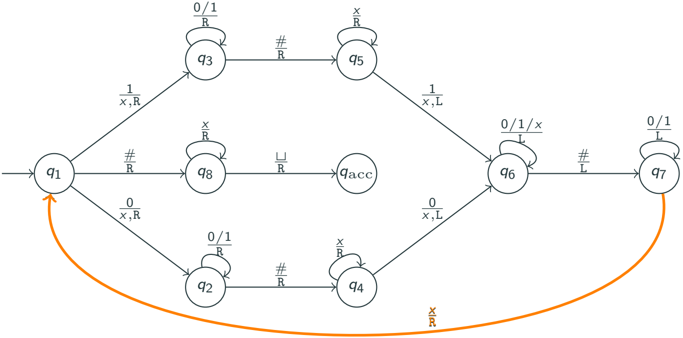


An example of a Turing machine on input 01101#01101



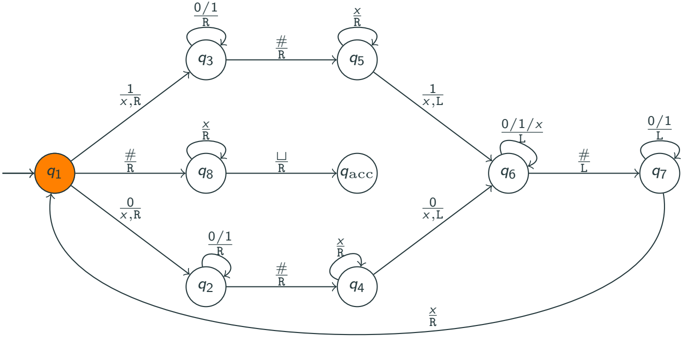
$\triangleleft$	x	x	1	0	1	#	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$
		↑												

An example of a Turing machine on input 01101#01101



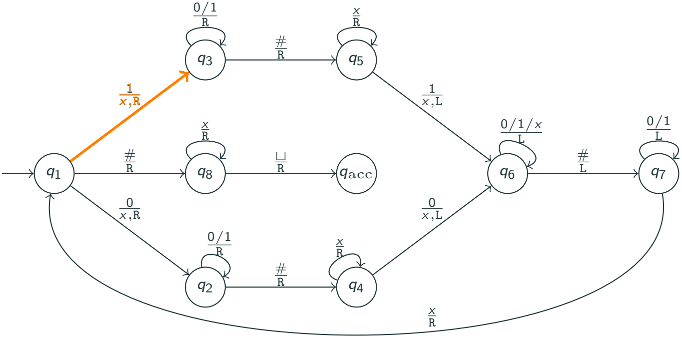
$\triangleleft$	x	x	1	0	1	#	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input $01101\#01101$



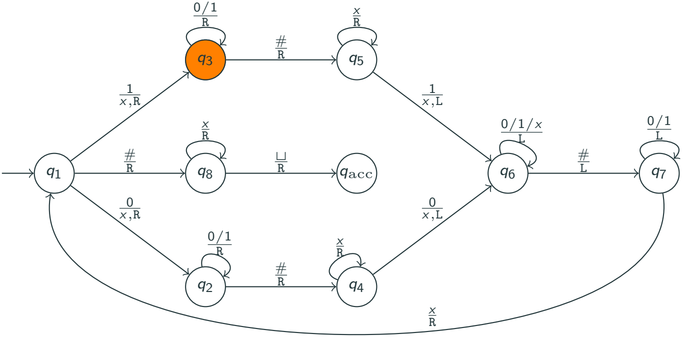
$\triangleleft$	x	x	1	0	1	$\#$	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$
			$\uparrow$											

An example of a Turing machine on input $01101\#01101$

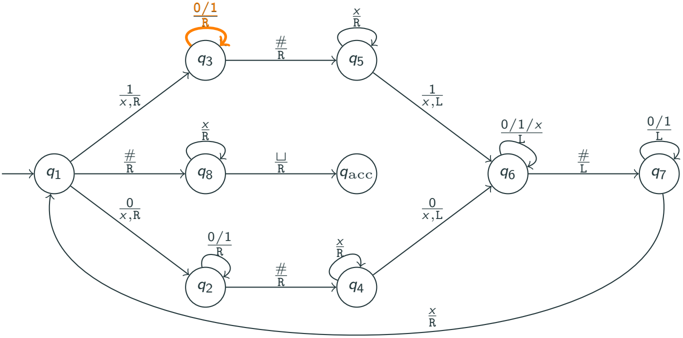


$\triangleleft$	x	x	1	0	1	$\#$	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$
			$\uparrow$											

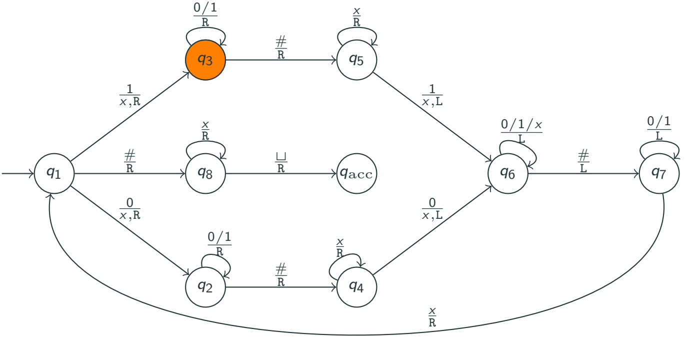
An example of a Turing machine on input 01101#01101



An example of a Turing machine on input 01101#01101

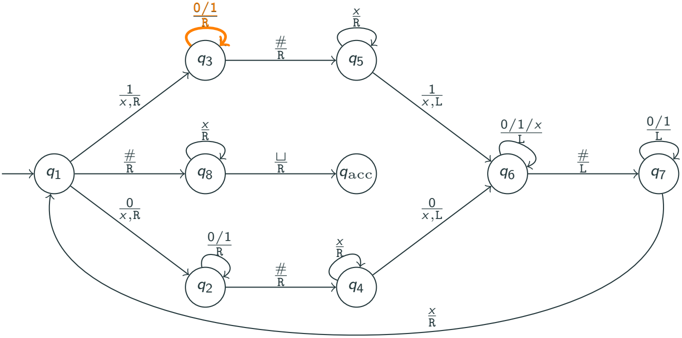


An example of a Turing machine on input 01101#01101



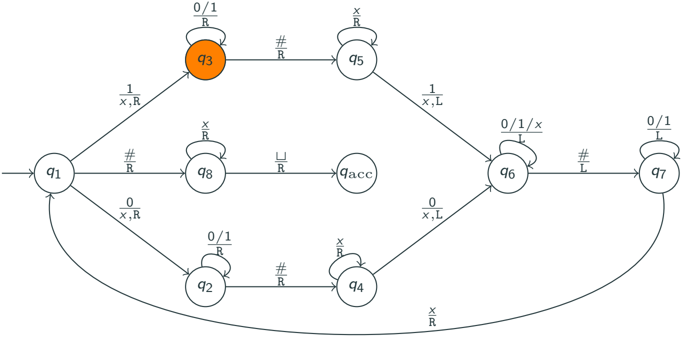
$\triangleleft$	x	x	x	0	1	#	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



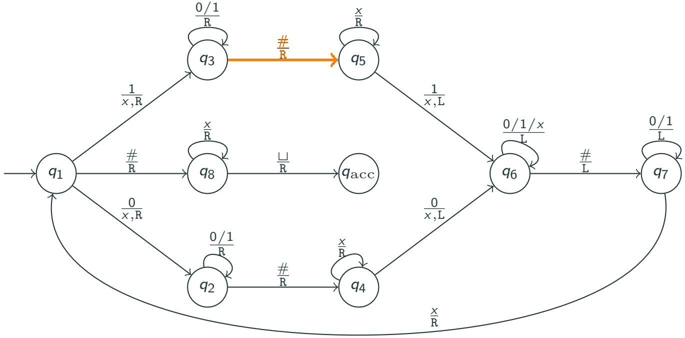
$\triangleleft$	x	x	x	0	1	#	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



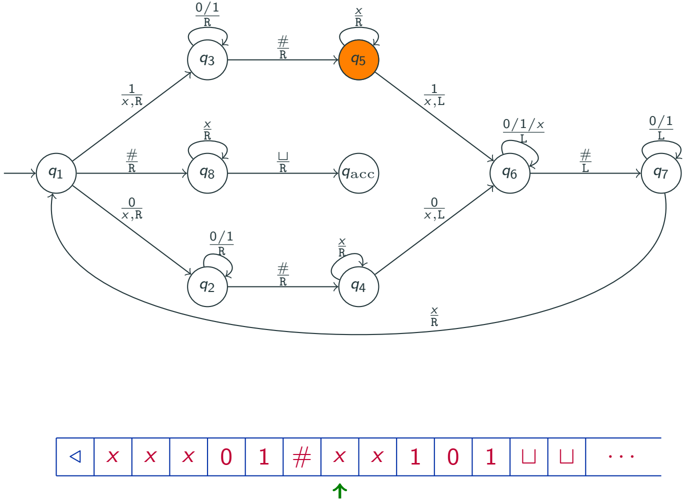
$\triangleleft$	x	x	x	0	1	#	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input $01101\#01101$

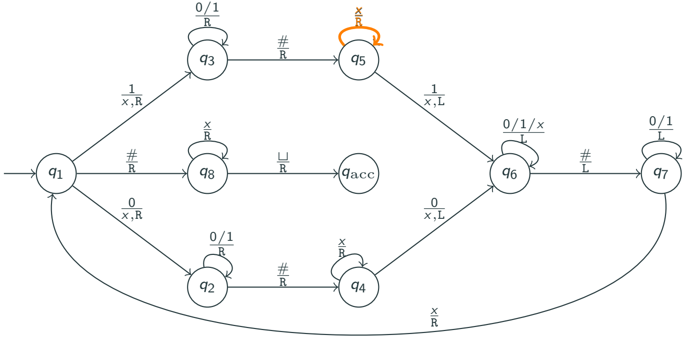


$\triangleleft$	x	x	x	0	1	$\#$	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

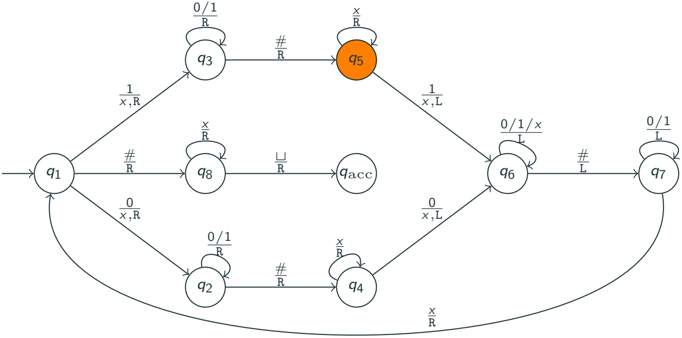


An example of a Turing machine on input 01101#01101



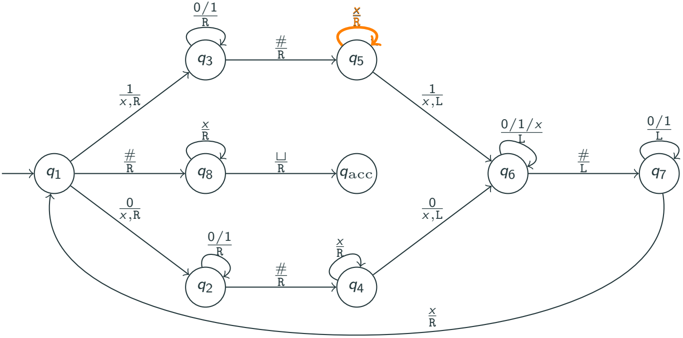
$\triangleleft$	x	x	x	0	1	#	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



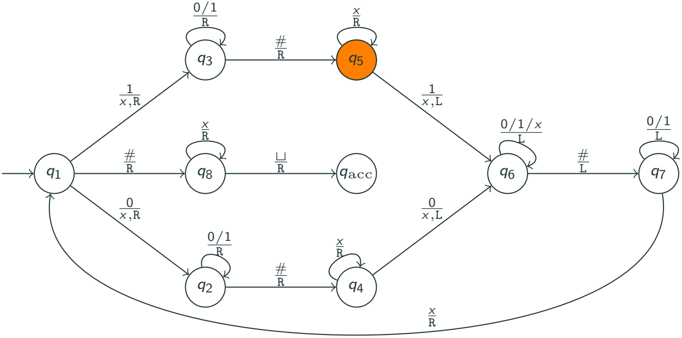
◁	x	x	x	0	1	#	x	x	1	0	1	⊔	⊔	···

An example of a Turing machine on input $01101\#01101$



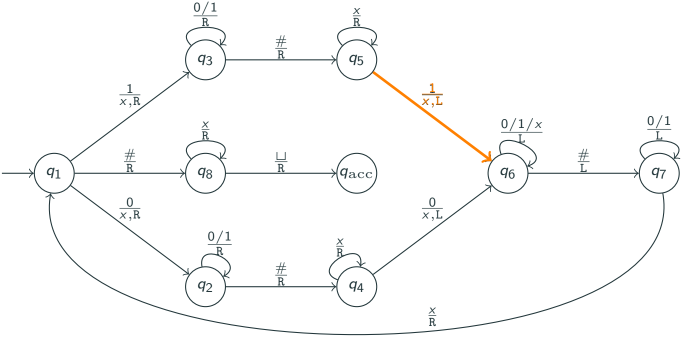
$\triangleleft$	x	x	x	0	1	$\#$	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



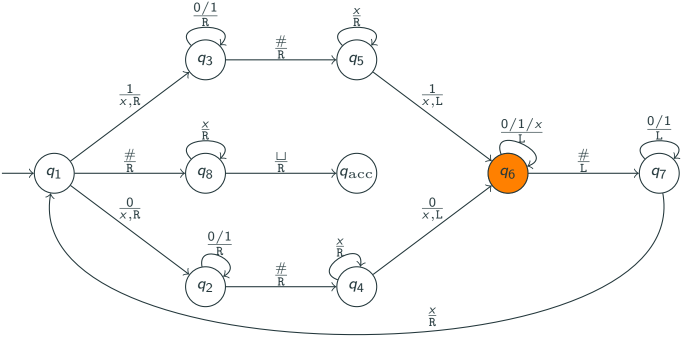
$\triangleleft$	x	x	x	0	1	#	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input $01101\#01101$



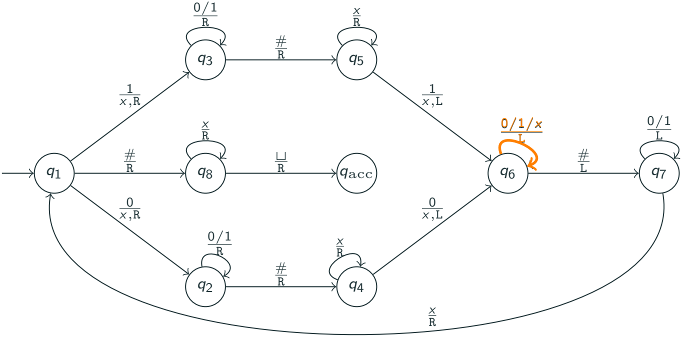
$\triangleleft$	x	x	x	0	1	$\#$	x	x	1	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input $01101\#01101$



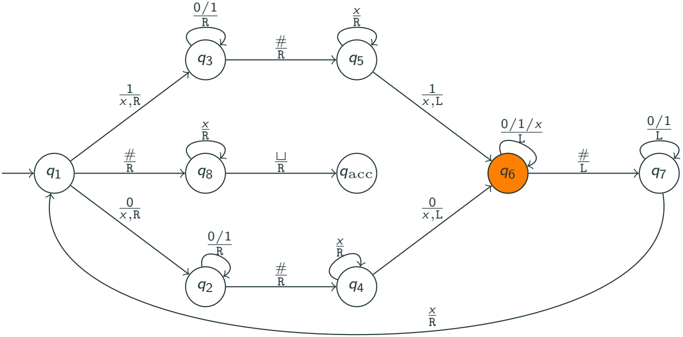
$\triangleleft$	x	x	x	0	1	$\#$	x	x	x	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input $01101\#01101$



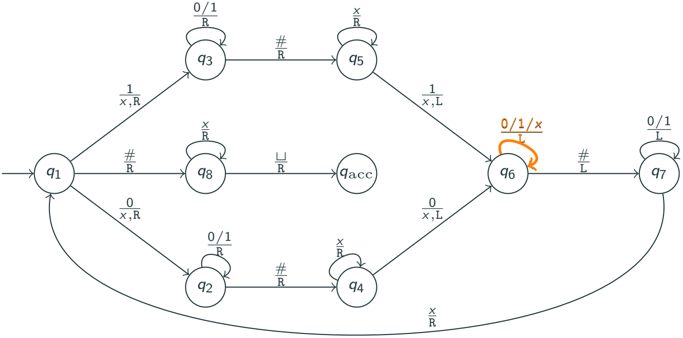
$\triangleleft$	x	x	x	0	1	$\#$	x	x	x	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



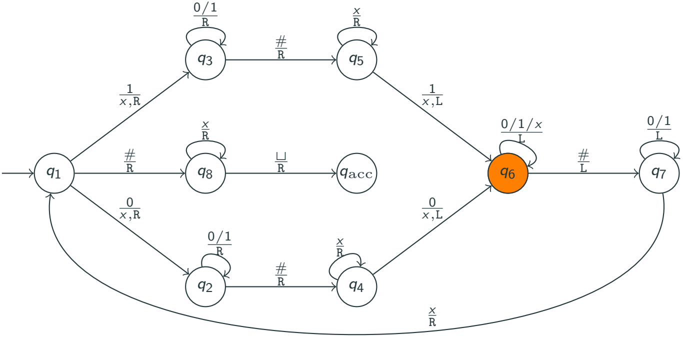
$\triangleleft$	x	x	x	0	1	#	x	x	x	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



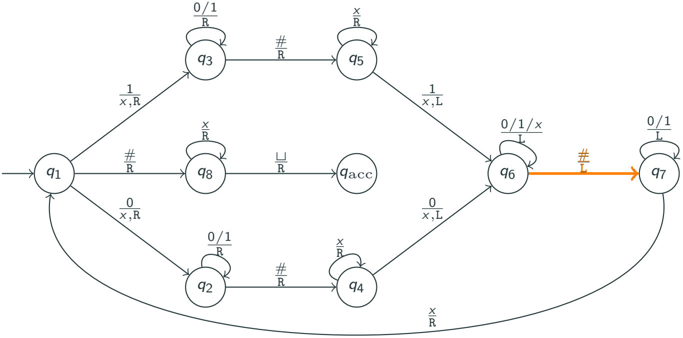
$\triangleleft$	x	x	x	0	1	#	x	x	x	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

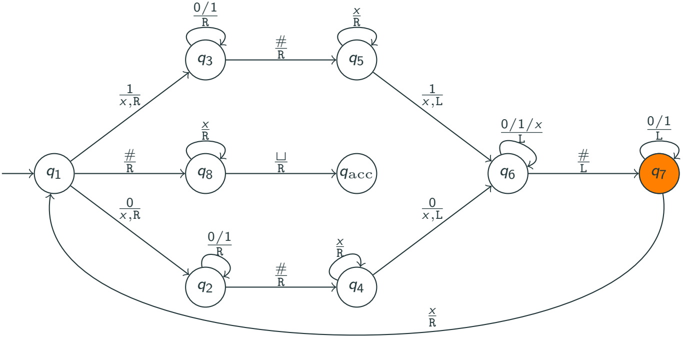


◁	x	x	x	0	1	#	x	x	x	0	1	⊔	⊔	⋯

An example of a Turing machine on input 01101#01101

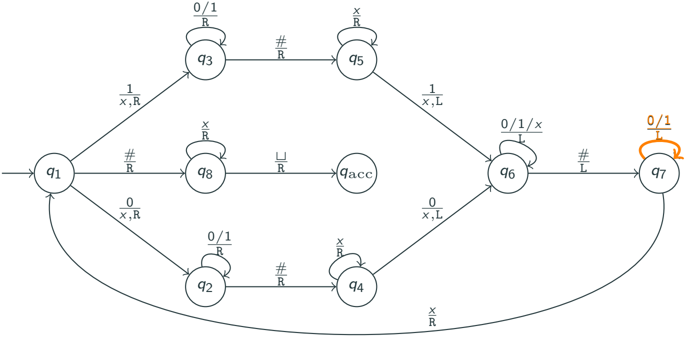


An example of a Turing machine on input 01101#01101

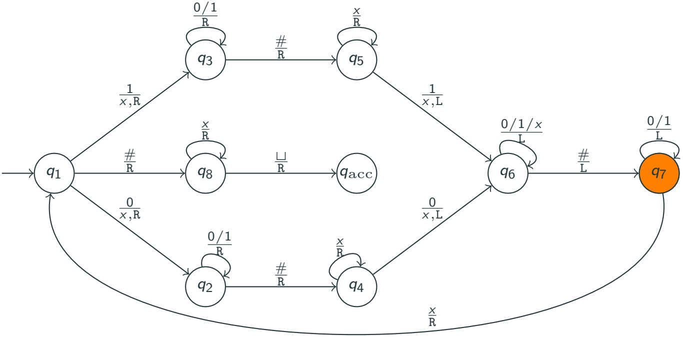


$\triangleleft$	x	x	x	0	1	#	x	x	x	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

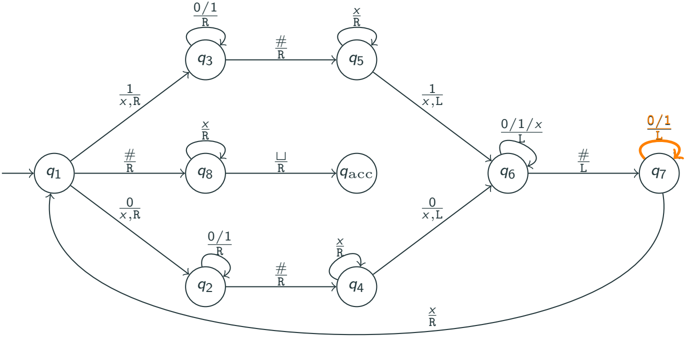


An example of a Turing machine on input 01101#01101

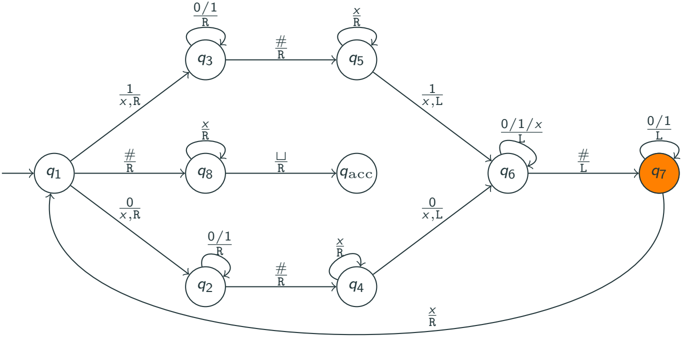


◁	x	x	x	0	1	#	x	x	x	0	1	⊔	⊔	···
				↑										

An example of a Turing machine on input $01101\#01101$

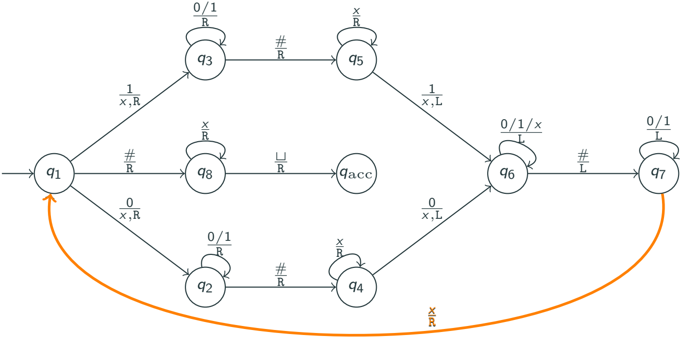


An example of a Turing machine on input 01101#01101



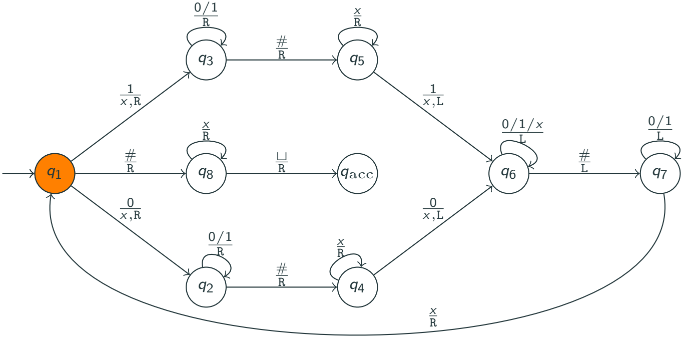
$\triangleleft$	x	x	x	0	1	#	x	x	x	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

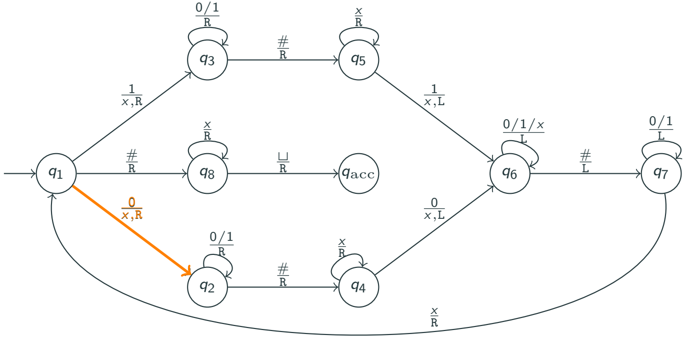


$\triangleleft$	x	x	x	0	1	#	x	x	x	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

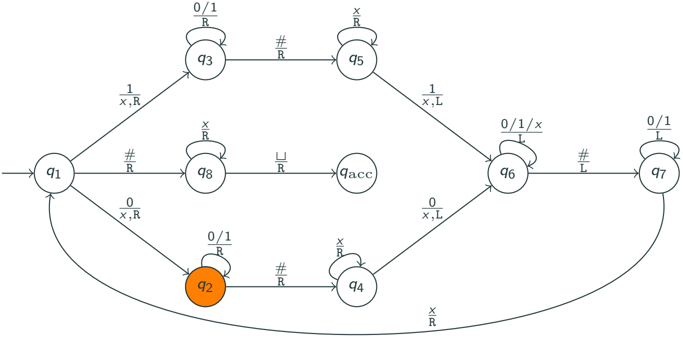


An example of a Turing machine on input 01101#01101



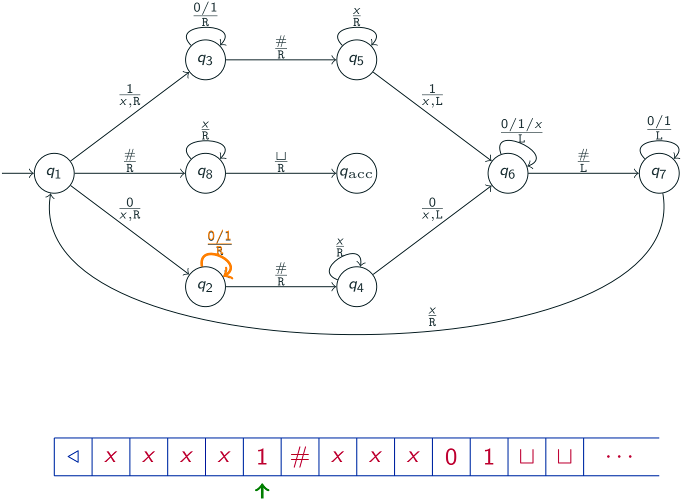
◁	x	x	x	0	1	#	x	x	x	0	1	⊔	⊔	···

An example of a Turing machine on input 01101#01101

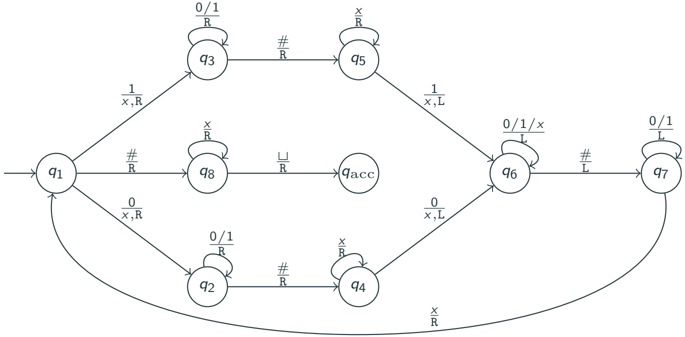


$\triangleleft$	x	x	x	x	1	#	x	x	x	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

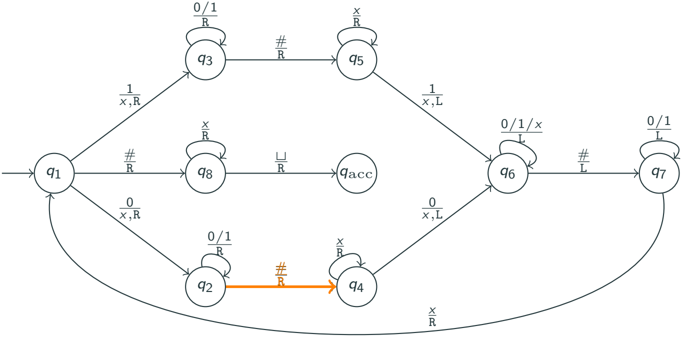


An example of a Turing machine on input 01101#01101



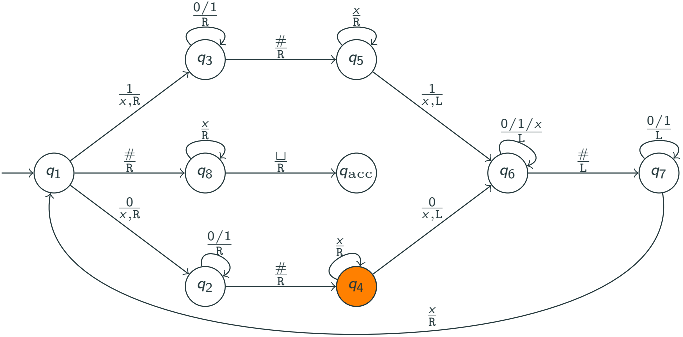
$\triangleleft$	x	x	x	x	1	#	x	x	x	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



$\triangleleft$	x	x	x	x	1	#	x	x	x	0	1	$\sqcup$	$\sqcup$	$\cdots$

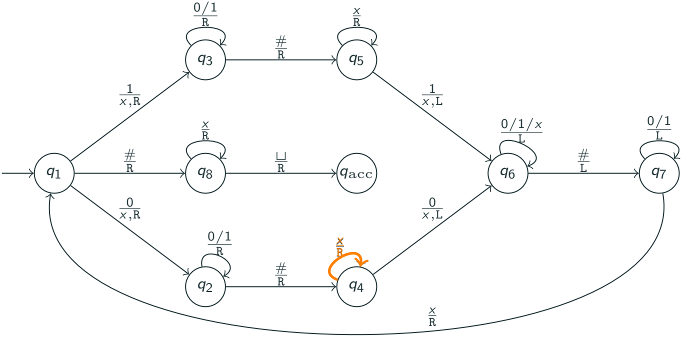
An example of a Turing machine on input 01101#01101



$\triangleleft$	x	x	x	x	1	#	x	x	x	0	1	$\sqcup$	$\sqcup$	$\cdots$

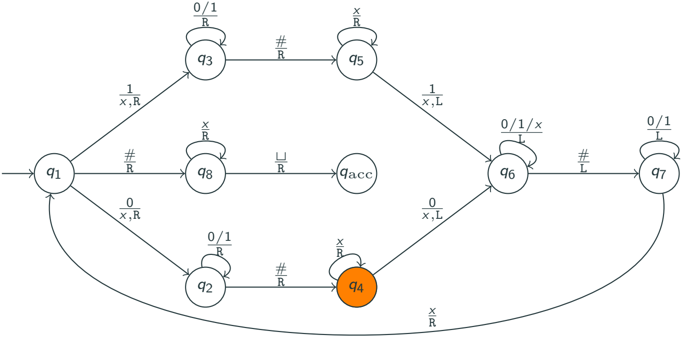
$\uparrow$

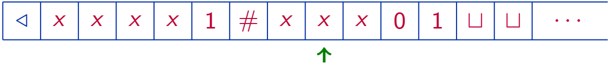
An example of a Turing machine on input 01101#01101



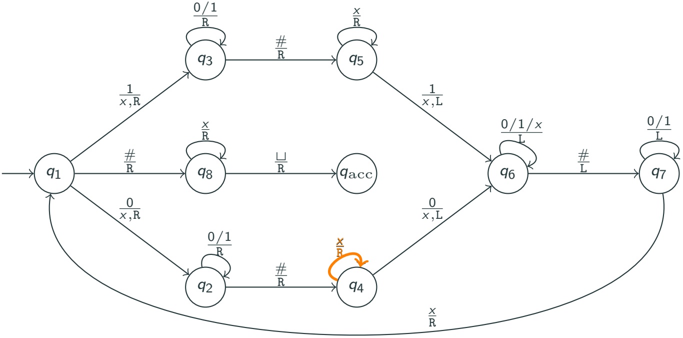
◁	x	x	x	x	1	#	x	x	x	0	1	⊔	⊔	···

An example of a Turing machine on input 01101#01101



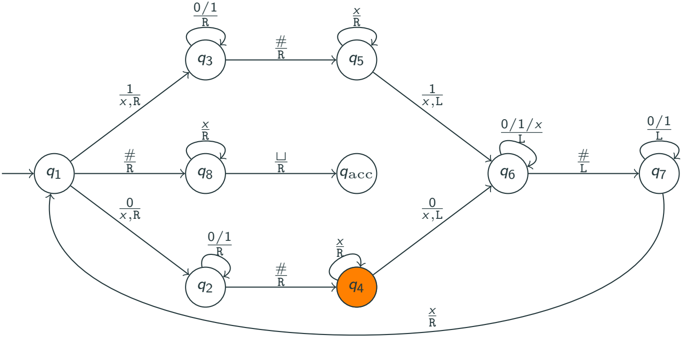


An example of a Turing machine on input 01101#01101



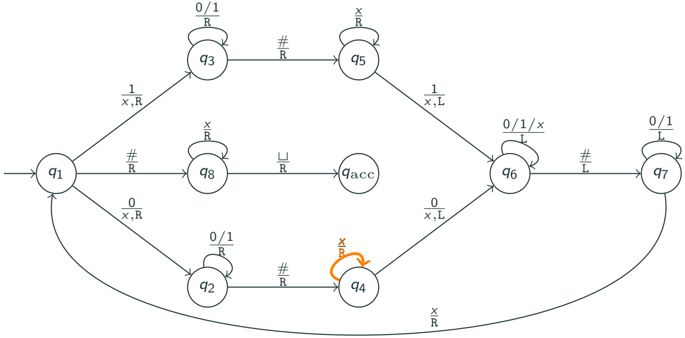
◁	x	x	x	x	1	#	x	x	x	0	1	⊔	⊔	···

An example of a Turing machine on input 01101#01101

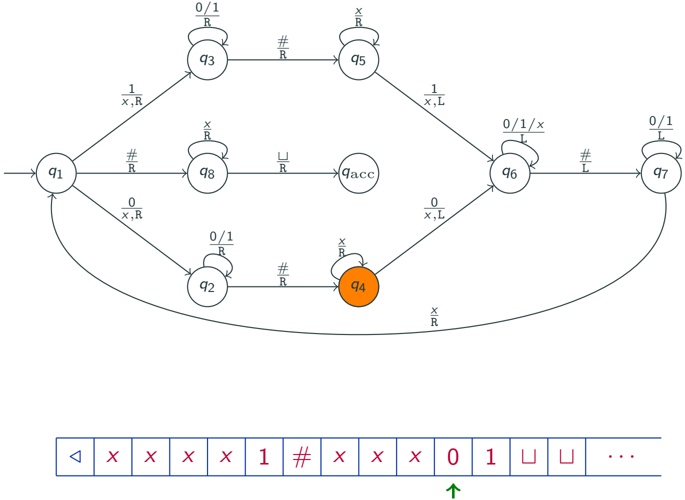


$\triangleleft$	x	x	x	x	1	#	x	x	x	0	1	$\sqcup$	$\sqcup$	$\cdots$

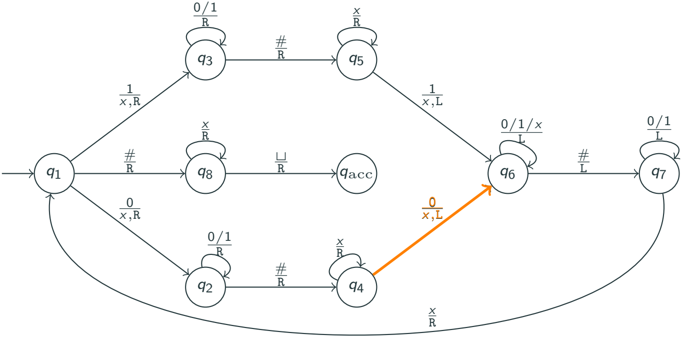
An example of a Turing machine on input 01101#01101



An example of a Turing machine on input 01101#01101

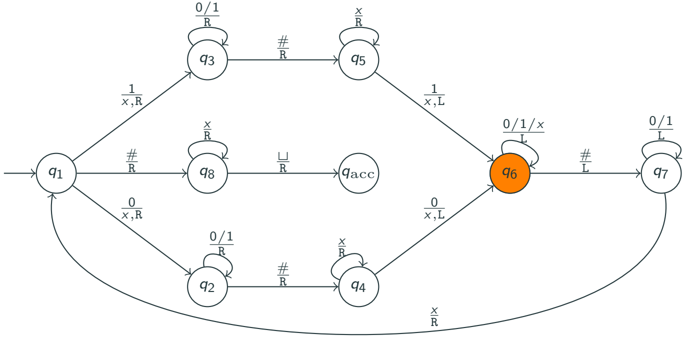


An example of a Turing machine on input 01101#01101



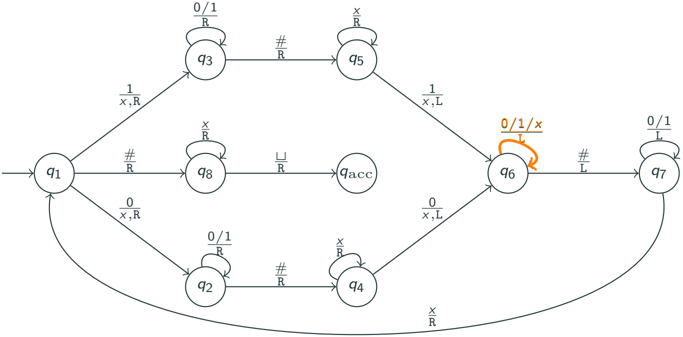
$\triangleleft$	x	x	x	x	1	#	x	x	x	0	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



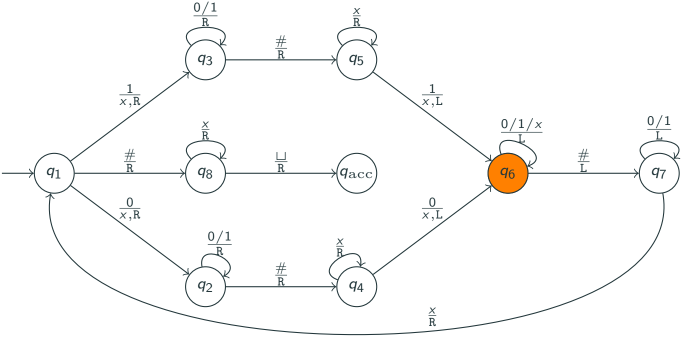
$\triangleleft$	x	x	x	x	1	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



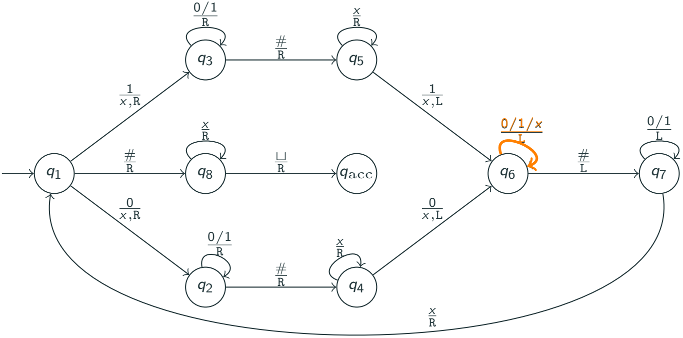
$\triangleleft$	x	x	x	x	1	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



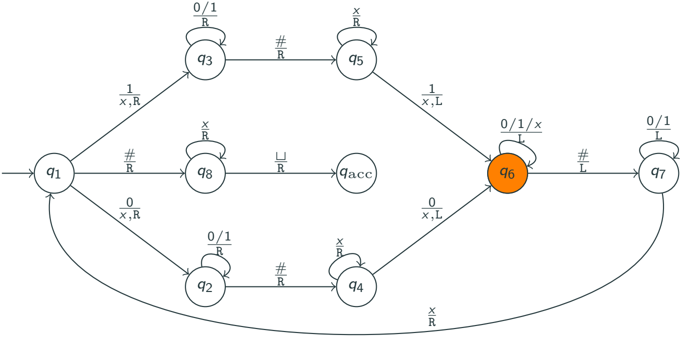
$\triangleleft$	x	x	x	x	1	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



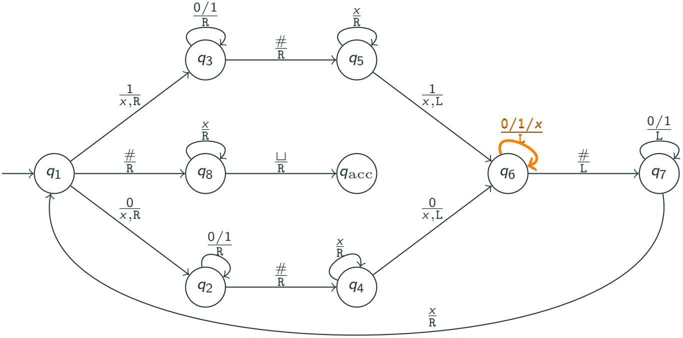
$\triangleleft$	x	x	x	x	1	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



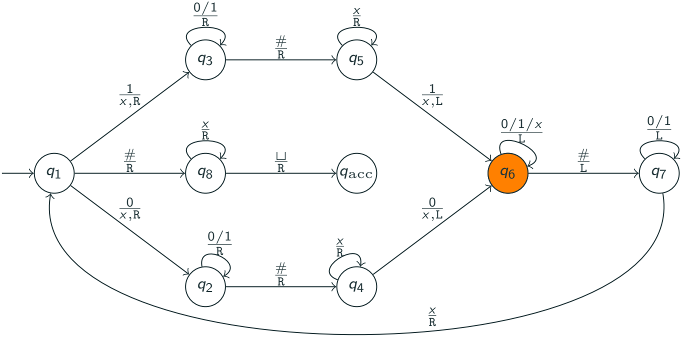
$\triangleleft$	x	x	x	x	1	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



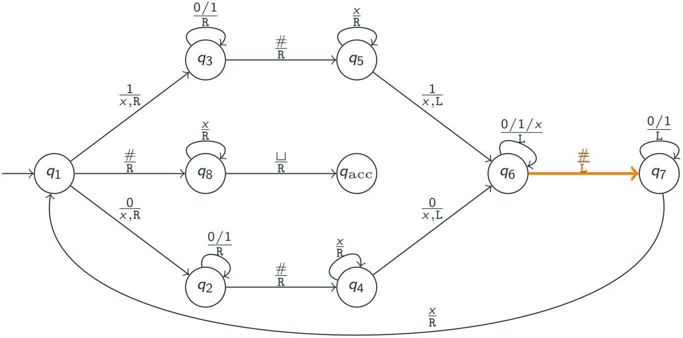
$\triangleleft$	x	x	x	x	1	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



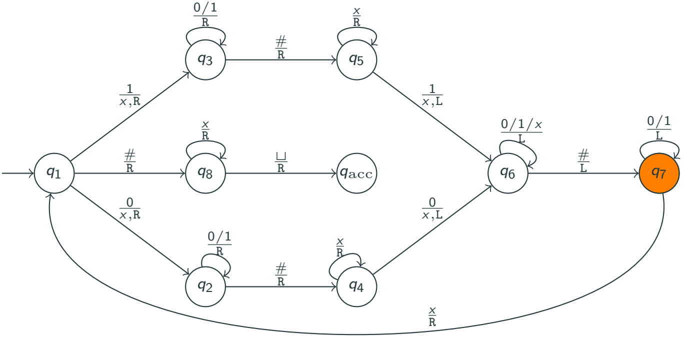
$\triangleleft$	x	x	x	x	1	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



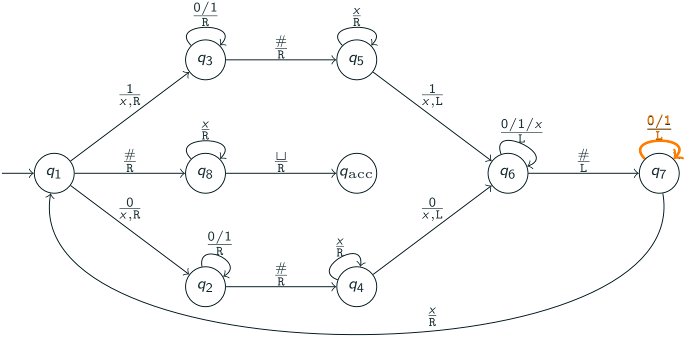
$\triangleleft$	x	x	x	x	1	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



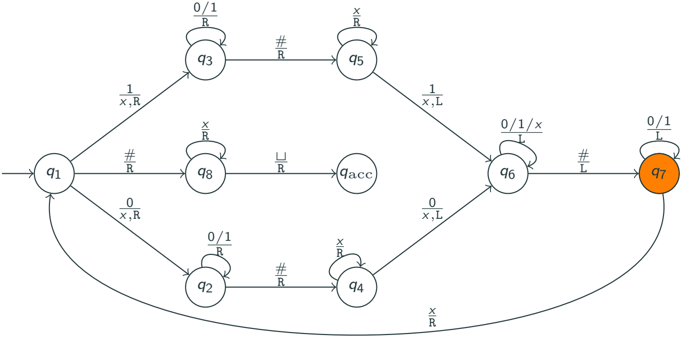
$\triangleleft$	x	x	x	x	1	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$
					$\uparrow$									

An example of a Turing machine on input 01101#01101

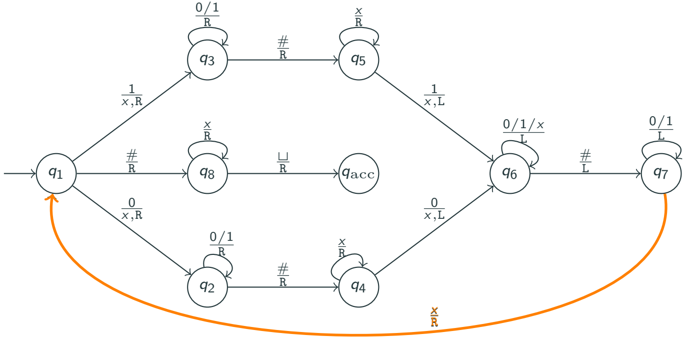


$\triangleleft$	x	x	x	x	1	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

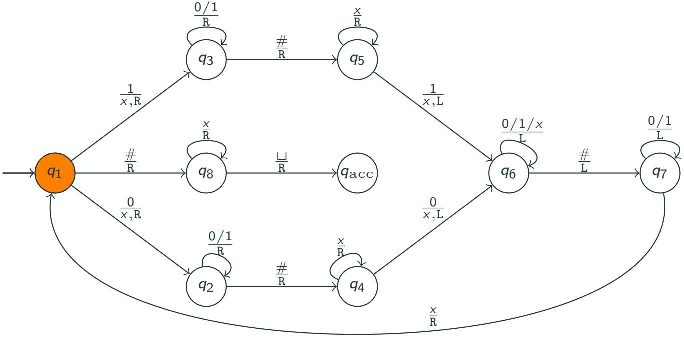


An example of a Turing machine on input 01101#01101



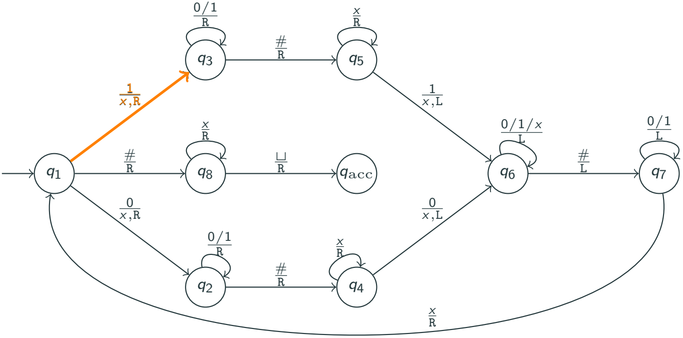
$\triangleleft$	x	x	x	x	1	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



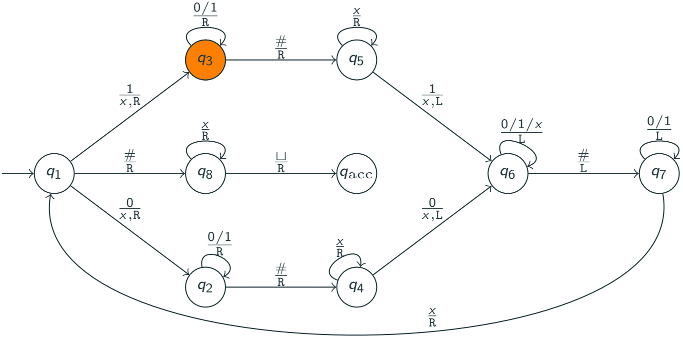
$\triangleleft$	x	x	x	x	1	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

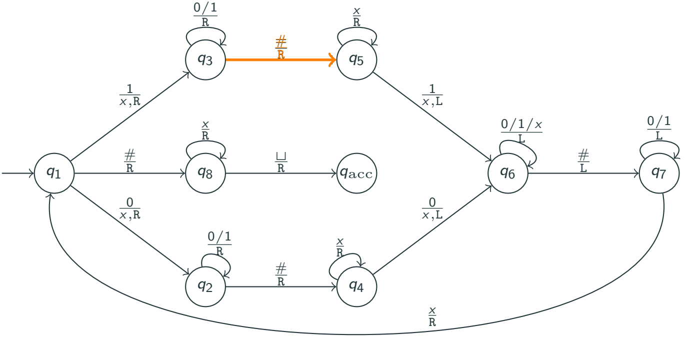


$\triangleleft$	x	x	x	x	1	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

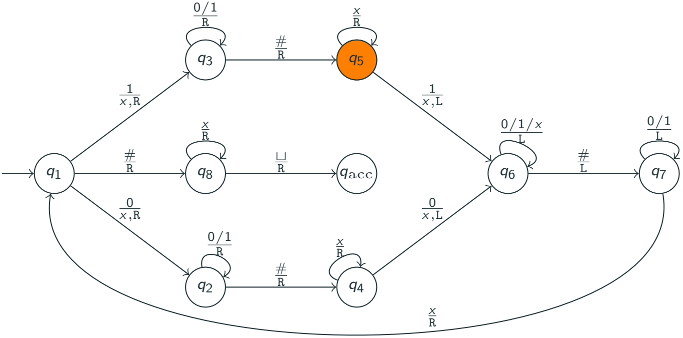


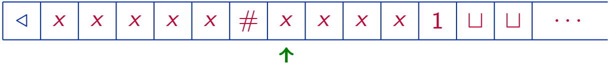
An example of a Turing machine on input 01101#01101



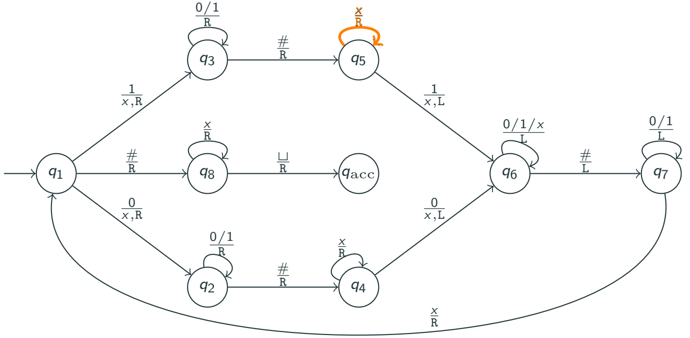
$\triangleleft$	x	x	x	x	x	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



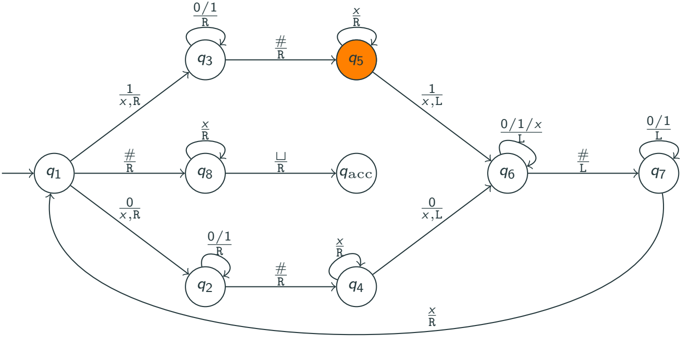


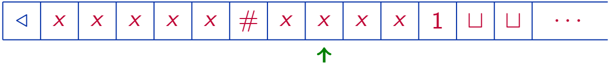
An example of a Turing machine on input 01101#01101



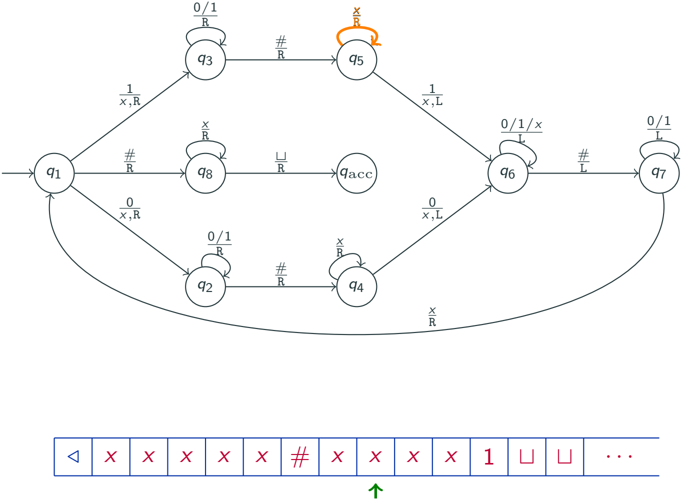
$\triangleleft$	x	x	x	x	x	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

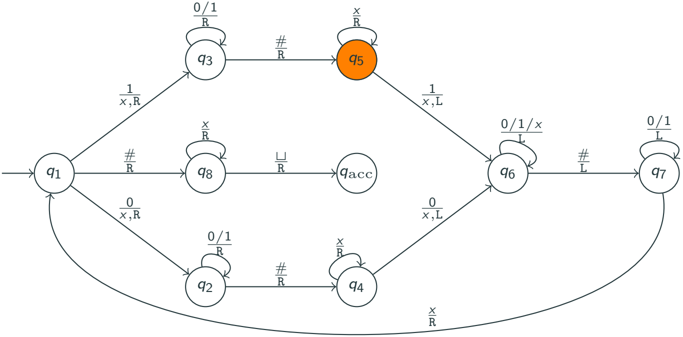




An example of a Turing machine on input 01101#01101

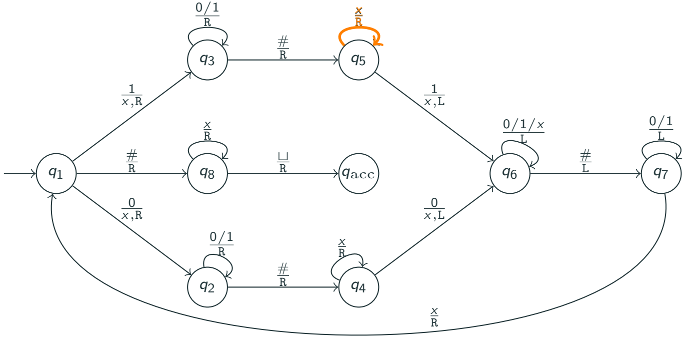


An example of a Turing machine on input 01101#01101

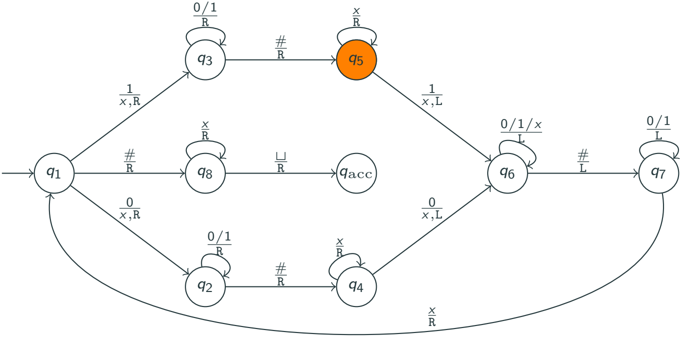


$\triangleleft$	x	x	x	x	x	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

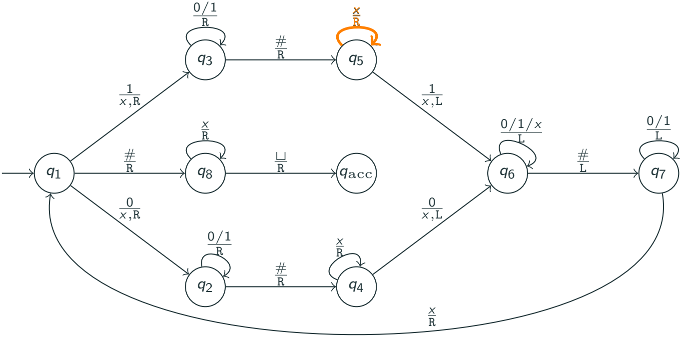


An example of a Turing machine on input $01101\#01101$



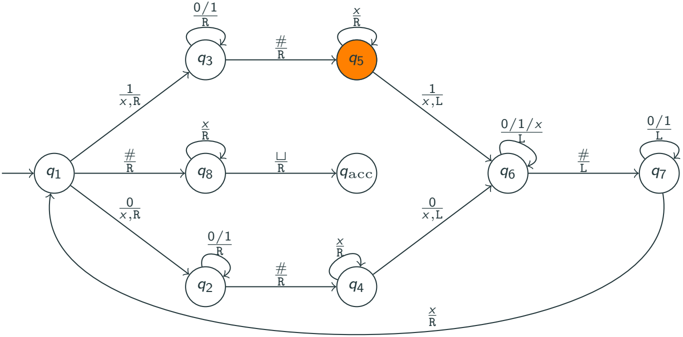
$\triangleleft$	x	x	x	x	x	$\#$	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



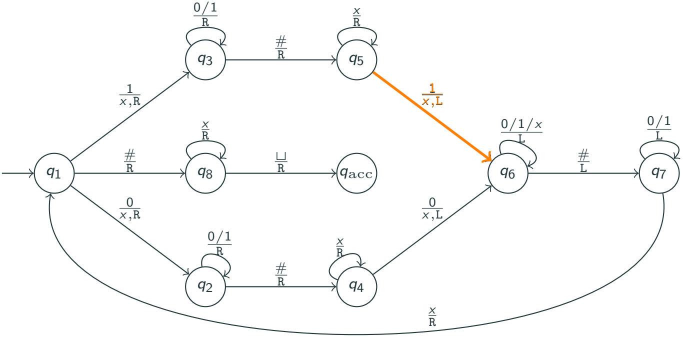
◁	x	x	x	x	x	#	x	x	x	x	1	⊔	⊔	···

An example of a Turing machine on input 01101#01101

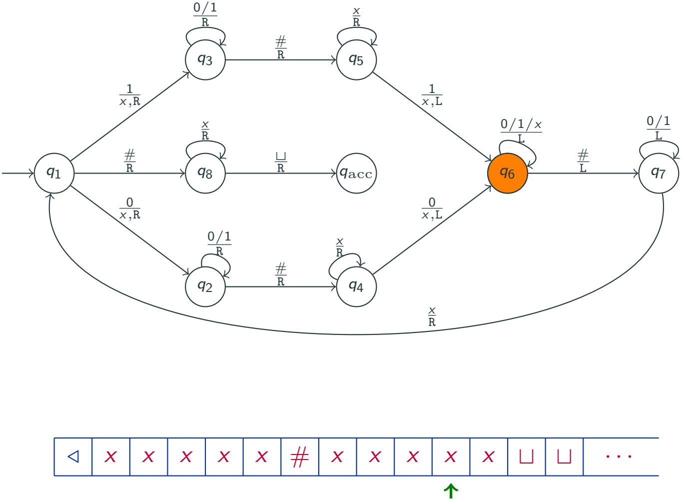


$\triangleleft$	x	x	x	x	x	#	x	x	x	x	1	$\sqcup$	$\sqcup$	$\cdots$

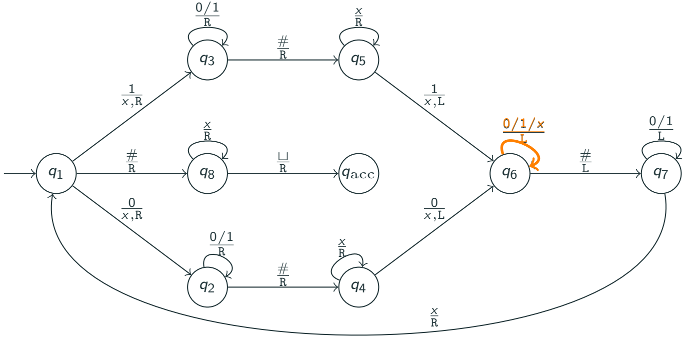
An example of a Turing machine on input 01101#01101



An example of a Turing machine on input $01101\#01101$

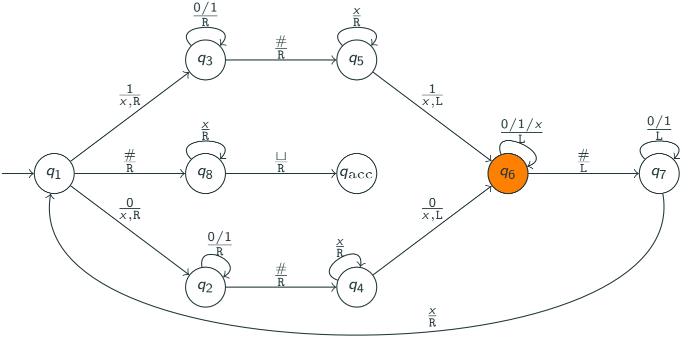


An example of a Turing machine on input $01101\#01101$

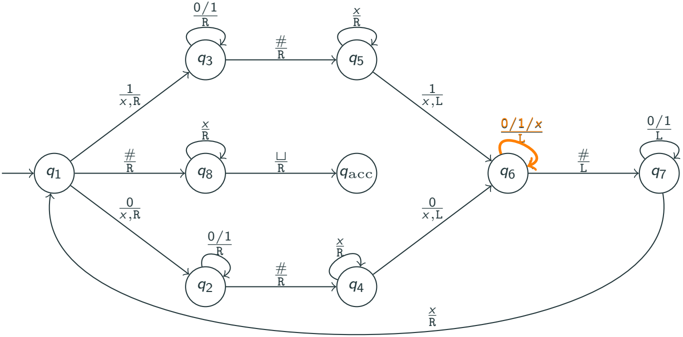


$\triangleleft$	x	x	x	x	x	$\#$	x	x	x	x	x	$\sqcup$	$\sqcup$	$\cdots$

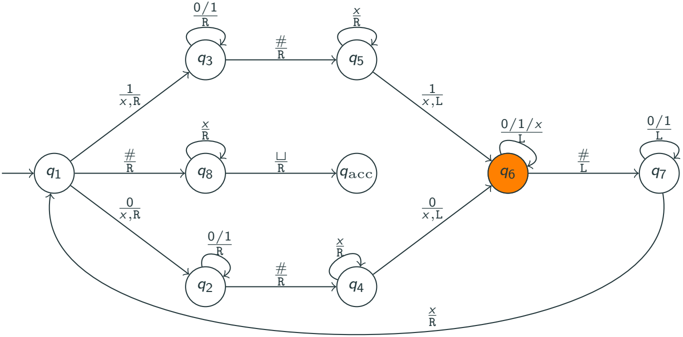
An example of a Turing machine on input 01101#01101



An example of a Turing machine on input 01101#01101

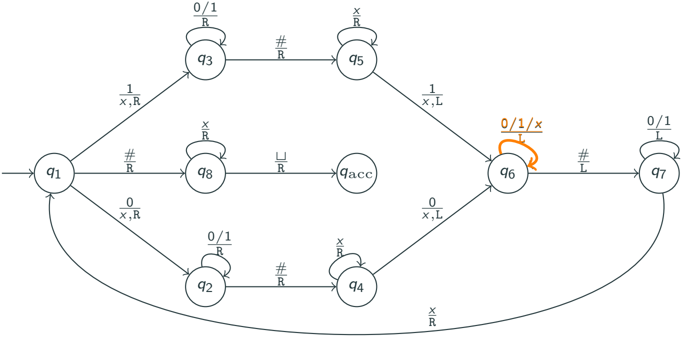


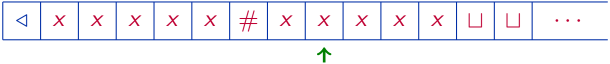
An example of a Turing machine on input $01101\#01101$



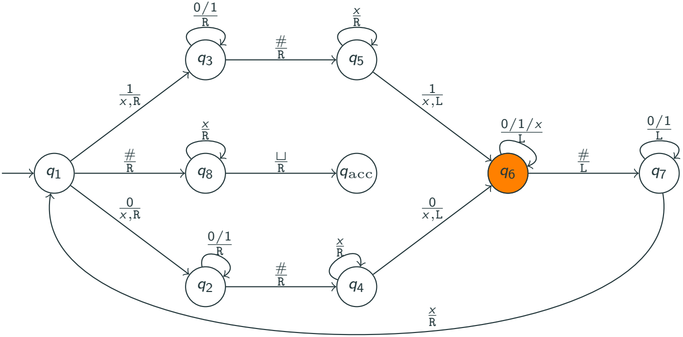
$\triangleleft$	x	x	x	x	x	$\#$	x	x	x	x	x	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input $01101\#01101$



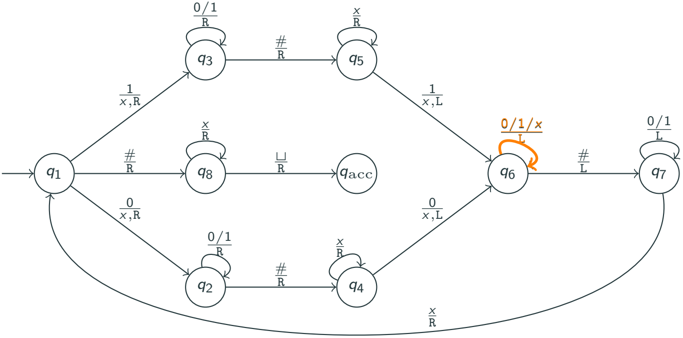


An example of a Turing machine on input $01101\#01101$



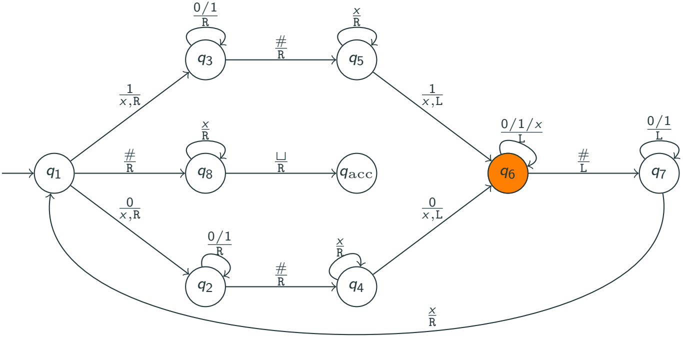
$\triangleleft$	x	x	x	x	x	$\#$	x	x	x	x	x	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101



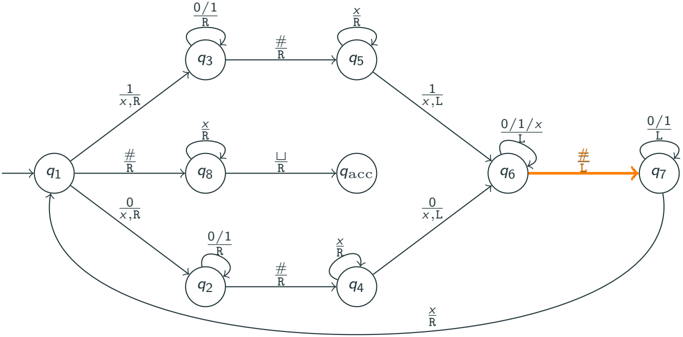
$\triangleleft$	x	x	x	x	x	$\#$	x	x	x	x	x	$\sqcup$	$\sqcup$	$\cdots$
							$\uparrow$							

An example of a Turing machine on input 01101#01101

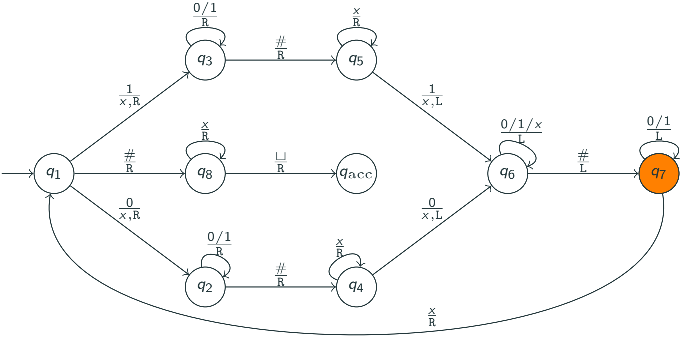


$\triangleleft$	x	x	x	x	x	$\#$	x	x	x	x	x	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

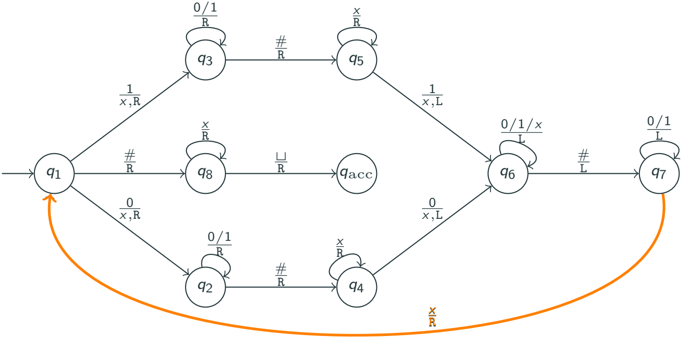


An example of a Turing machine on input $01101\#01101$

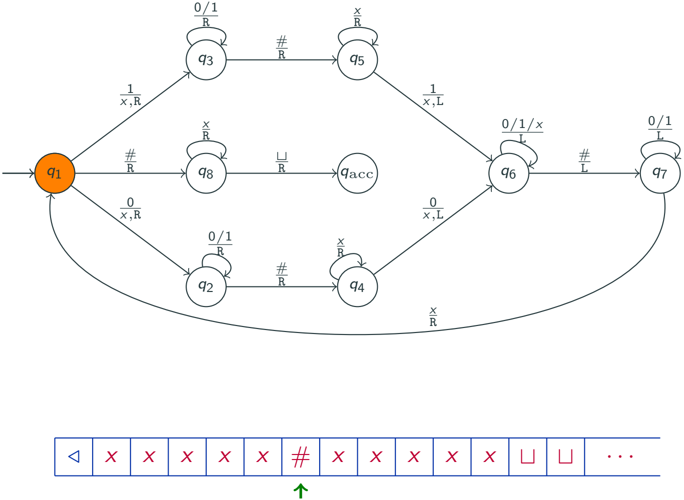


$\triangleleft$	x	x	x	x	x	$\#$	x	x	x	x	x	$\sqcup$	$\sqcup$	$\cdots$

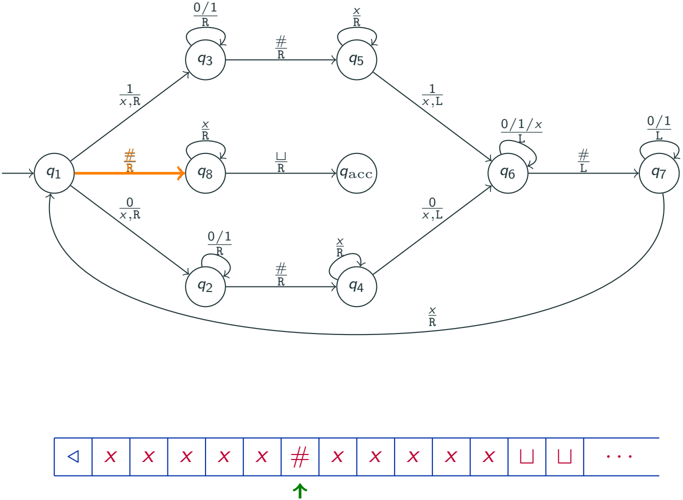
An example of a Turing machine on input 01101#01101



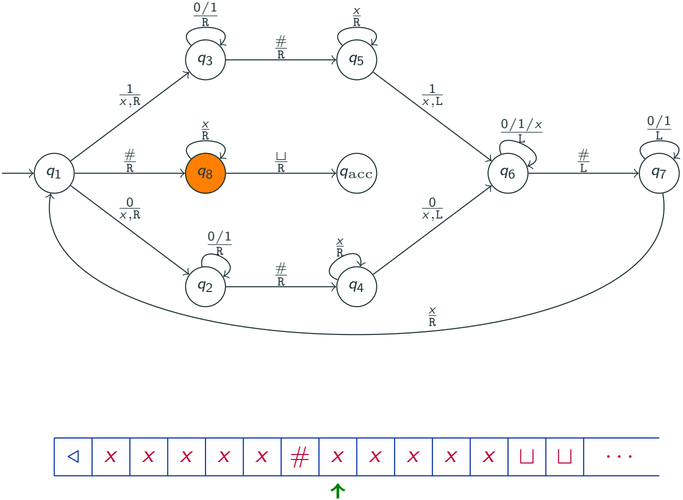
An example of a Turing machine on input 01101#01101



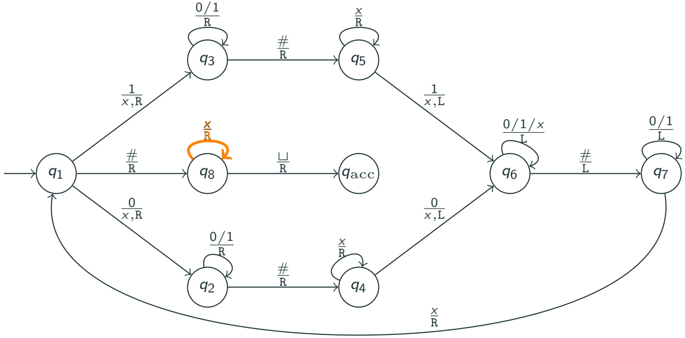
An example of a Turing machine on input 01101#01101

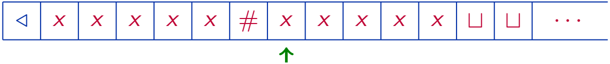


An example of a Turing machine on input 01101#01101

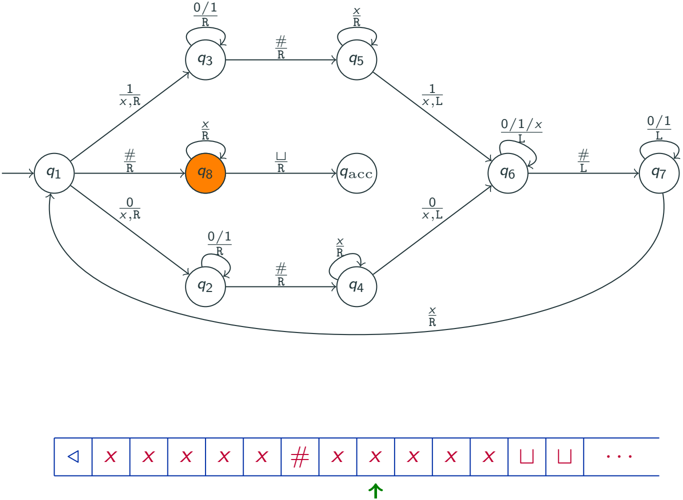


An example of a Turing machine on input 01101#01101

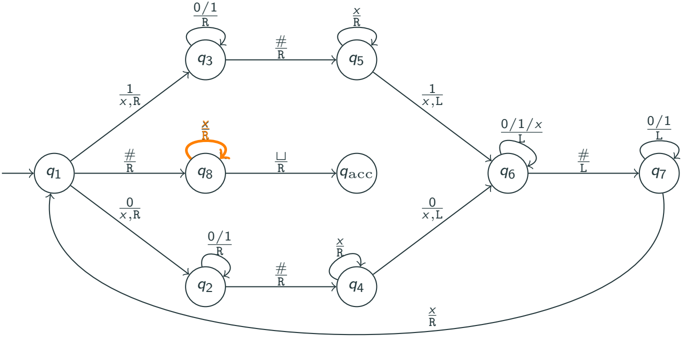


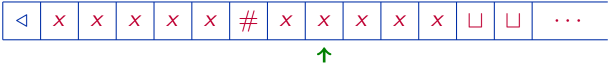


An example of a Turing machine on input 01101#01101

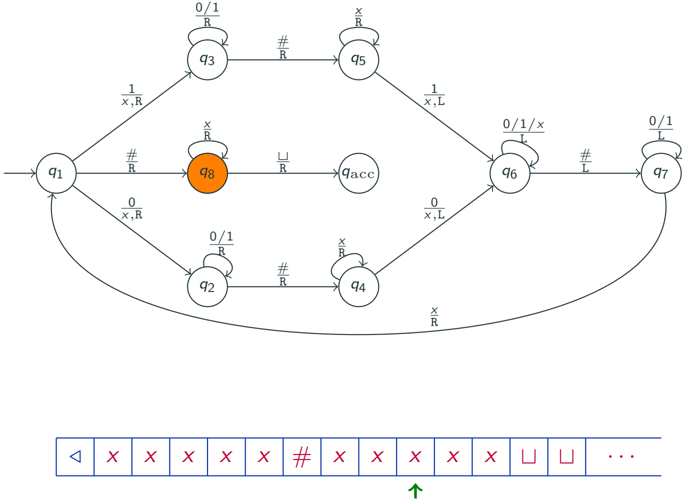


An example of a Turing machine on input 01101#01101

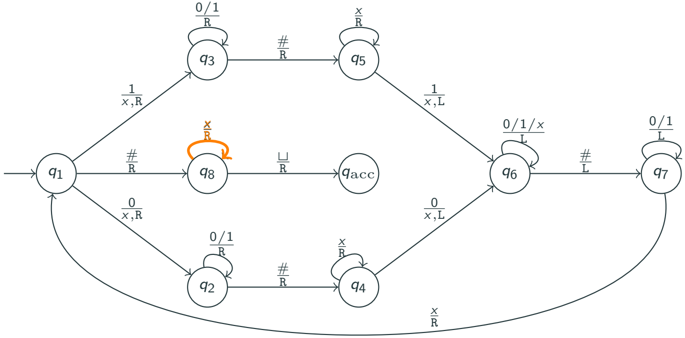




An example of a Turing machine on input 01101#01101

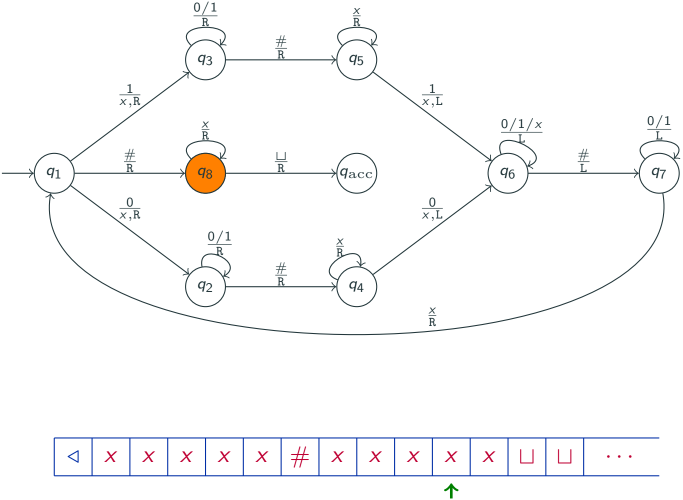


An example of a Turing machine on input 01101#01101

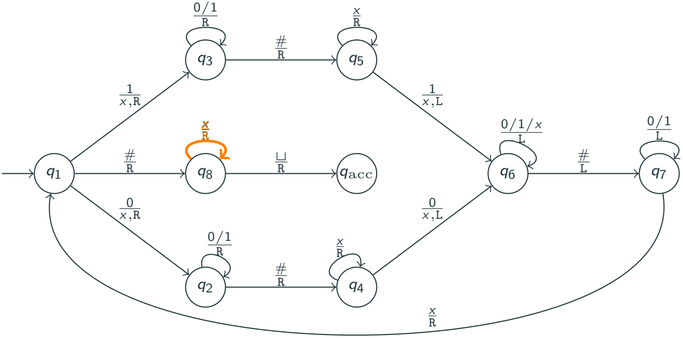


$\triangleleft$	x	x	x	x	x	$\#$	x	x	x	x	x	$\sqcup$	$\sqcup$	$\cdots$
									$\uparrow$					

An example of a Turing machine on input 01101#01101

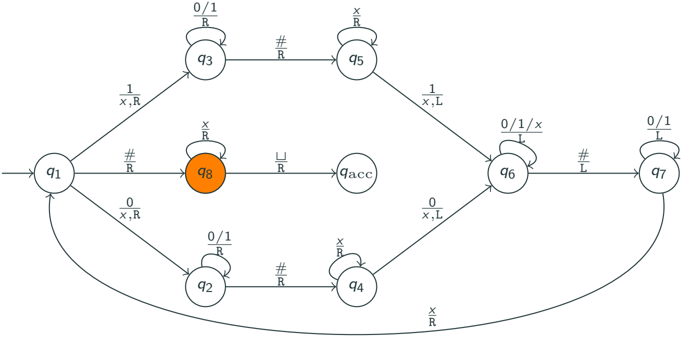


An example of a Turing machine on input 01101#01101

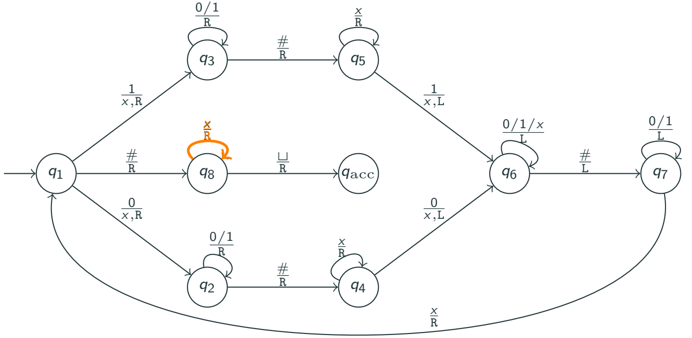


$\triangleleft$	x	x	x	x	x	#	x	x	x	x	x	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input 01101#01101

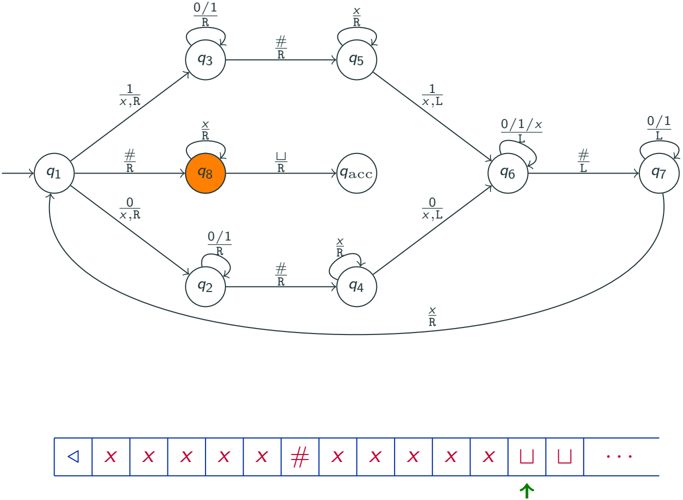


An example of a Turing machine on input 01101#01101

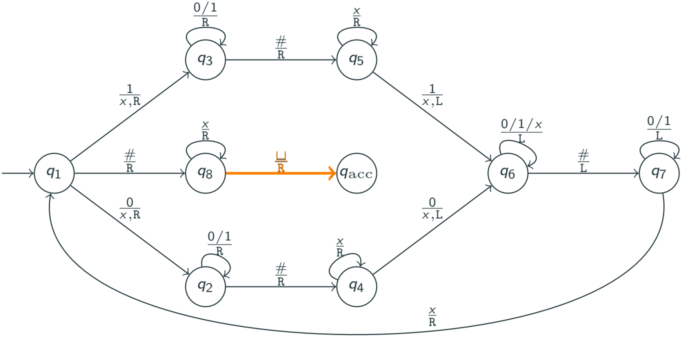


$\triangleleft$	x	x	x	x	x	$\#$	x	x	x	x	x	$\sqcup$	$\sqcup$	$\cdots$

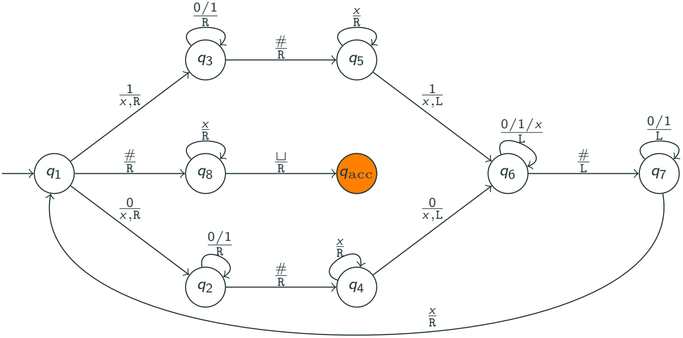
An example of a Turing machine on input 01101#01101



An example of a Turing machine on input 01101#01101

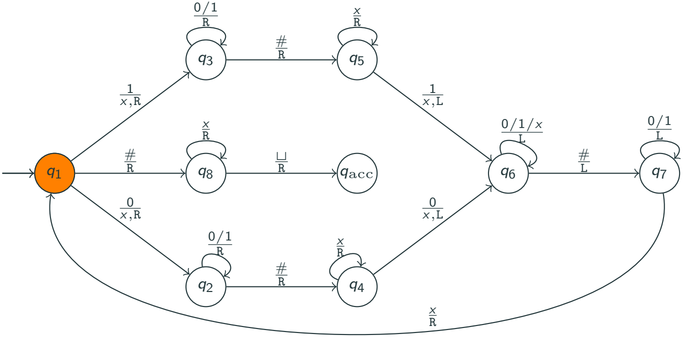


An example of a Turing machine on input 01101#01101



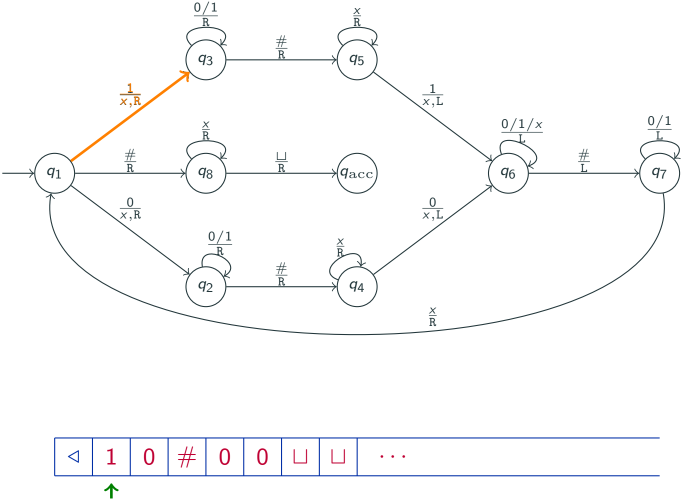
$\triangleleft$	x	x	x	x	x	$\#$	x	x	x	x	x	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input $10\#00$

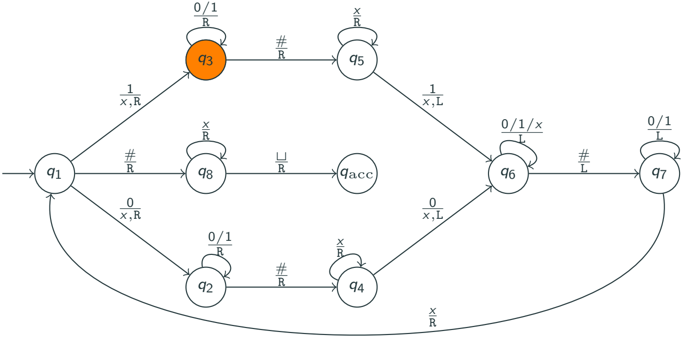


$\triangleleft$	1	0	#	0	0	$\sqcup$	$\sqcup$	$\cdots$

An example of a Turing machine on input $10\#00$

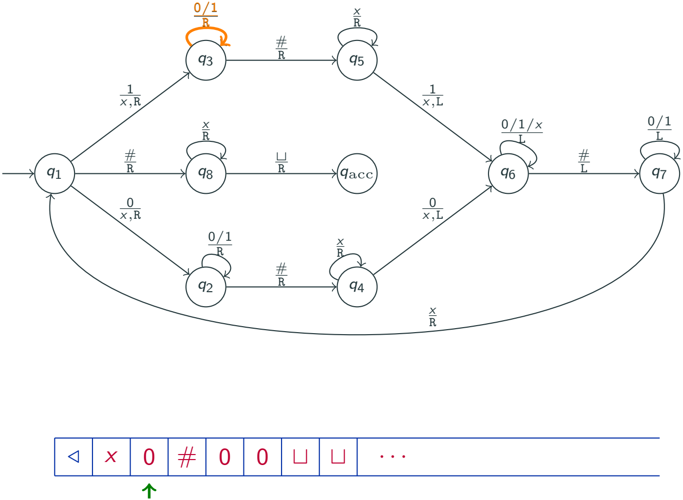


An example of a Turing machine on input $10\#00$

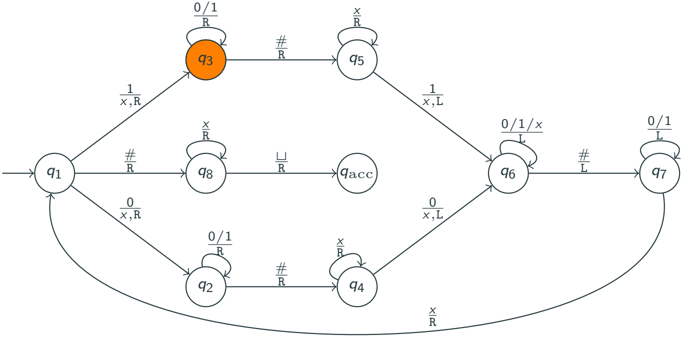


$\triangleleft$	x	0	#	0	0	$\sqcup$	$\sqcup$	$\cdots$

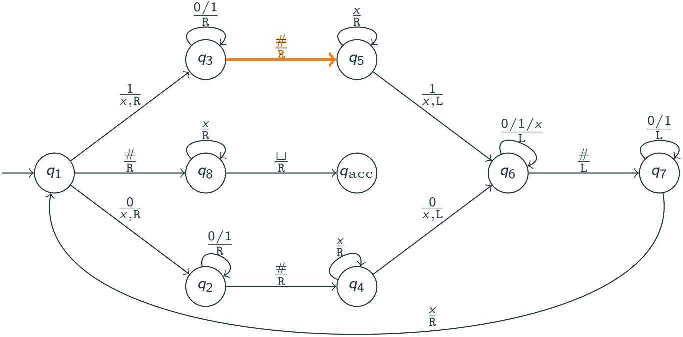
An example of a Turing machine on input $10\#00$



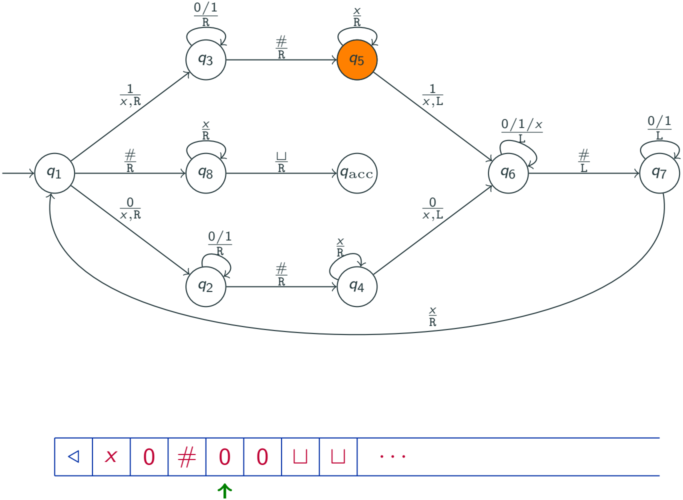
An example of a Turing machine on input $10\#00$



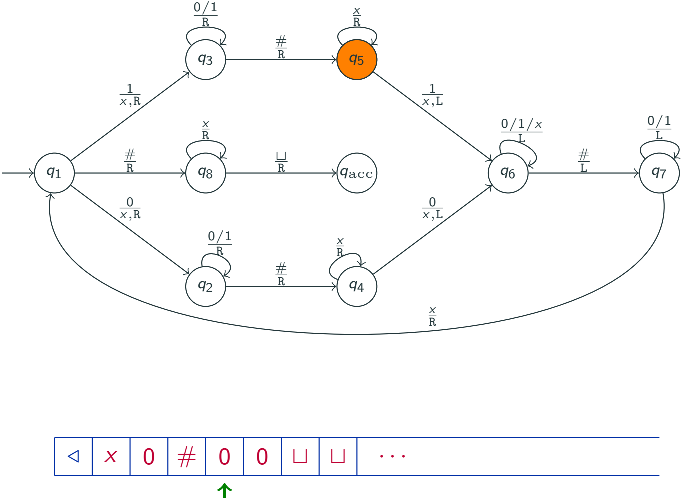
An example of a Turing machine on input $10\#00$



An example of a Turing machine on input $10\#00$

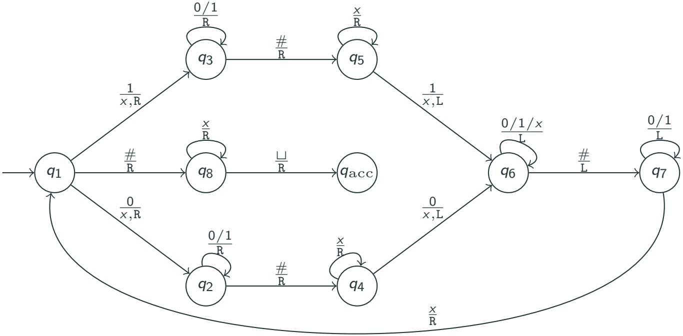


An example of a Turing machine on input $10\#00$



From state q_5, it enters q_{rej}.

An example of a Turing machine



This TM accepts the input word iff it is of the form: $w \# w$ where $w \in \{0,1\}^*$.

The algorithmic meaning of the TM

It works as follows:

$$\triangleleft \; \xleftarrow{\hspace{6em}} w_1 \xrightarrow{\hspace{6em}} \# \xleftarrow{\hspace{6em}} w_2 \xrightarrow{\hspace{6em}}$$

$\uparrow$

The algorithmic meaning of the TM

It works as follows:

$$\begin{array}{l} \triangleleft \;\longleftarrow\!\!\!-\!\!-\!\!-\; w_1 \;-\!\!-\!\!-\!\!\!\longrightarrow \# \longleftarrow\!\!\!-\!\!-\!\!-\; w_2 \;-\!\!-\!\!-\!\!\!\longrightarrow \\ \quad\;\uparrow \end{array}$$

- Read the "first" symbol of w_1, "mark" it and "remember" it.

The algorithmic meaning of the TM

It works as follows:

$$\triangleleft \quad \overset{0}{\longleftarrow}\!\!-\!\!-\!\!-\; w_1 \;\longrightarrow \quad \# \quad \longleftarrow\; w_2 \;\longrightarrow$$
$$\uparrow$$

- Read the “first” symbol of w_1, “mark” it and “remember” it.

The algorithmic meaning of the TM

It works as follows:

- Read the “first” symbol of w_1, “mark” it and “remember” it.

The algorithmic meaning of the TM

It works as follows:

$$\triangleleft \quad \overset{\cancel{0}}{\longleftarrow}\!\!\longleftarrow w_1 \longrightarrow \;\#\; \longleftarrow w_2 \longrightarrow$$

$$\uparrow$$

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).

The algorithmic meaning of the TM

It works as follows:

$$\triangleleft \quad \cancel{0} \longleftarrow w_1 \longrightarrow \# \longleftarrow w_2 \longrightarrow$$

$$\uparrow$$

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).

The algorithmic meaning of the TM

It works as follows:

- Read the “first” symbol of w_1, “mark” it and “remember” it.
- Go to the “first” symbol of w_2 and check if it is the same (as the remembered symbol).

The algorithmic meaning of the TM

It works as follows:

$$\triangleleft \;\overset{\cancel{0}}{\longleftarrow}\!\!\longrightarrow w_1 \longrightarrow \;\#\; \longleftarrow w_2 \longrightarrow$$

$$\uparrow$$

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).

The algorithmic meaning of the TM

It works as follows:

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).

The algorithmic meaning of the TM

It works as follows:

$$\triangleleft \quad \overset{\cancel{0}}{\longleftarrow\!\!-\!\!-\!\!-} \; w_1 \; \longrightarrow \quad \# \quad \longleftarrow \; w_2 \; \longrightarrow$$

$$\uparrow$$

- Read the “first” symbol of w_1, “mark” it and “remember” it.
- Go to the “first” symbol of w_2 and check if it is the same (as the remembered symbol).

The algorithmic meaning of the TM

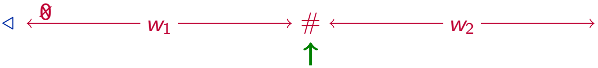
It works as follows:

$$\triangleleft \quad \overset{\cancel{0}}{\longleftarrow\!\!-\!\!-\!\!-} \; w_1 \longrightarrow \# \longleftarrow w_2 \longrightarrow$$

- Read the “first” symbol of w_1, “mark” it and “remember” it.
- Go to the “first” symbol of w_2 and check if it is the same (as the remembered symbol).

The algorithmic meaning of the TM

It works as follows:



- Read the “first” symbol of w_1, “mark” it and “remember” it.
- Go to the “first” symbol of w_2 and check if it is the same (as the remembered symbol).

The algorithmic meaning of the TM

It works as follows:

- Read the “first” symbol of w_1, “mark” it and “remember” it.
- Go to the “first” symbol of w_2 and check if it is the same (as the remembered symbol).

The algorithmic meaning of the TM

It works as follows:

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.

The algorithmic meaning of the TM

It works as follows:

- Read the “first” symbol of w_1, “mark” it and “remember” it.
- Go to the “first” symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, “mark” it.

The algorithmic meaning of the TM

It works as follows:

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.

The algorithmic meaning of the TM

It works as follows:

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.

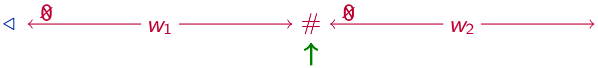
The algorithmic meaning of the TM

It works as follows:

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides.
  The "first" symbol means the "first unmarked" symbol.

The algorithmic meaning of the TM

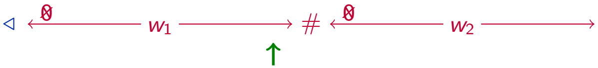
It works as follows:



- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides. The "first" symbol means the "first unmarked" symbol.

The algorithmic meaning of the TM

It works as follows:



- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides.
  The "first" symbol means the "first unmarked" symbol.

The algorithmic meaning of the TM

It works as follows:

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides. The "first" symbol means the "first unmarked" symbol.

The algorithmic meaning of the TM

It works as follows:

- Read the “first” symbol of w_1, “mark” it and “remember” it.
- Go to the “first” symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, “mark” it.
- Repeat until there is no more “unmarked” symbol on both sides. The “first” symbol means the “first unmarked” symbol.

The algorithmic meaning of the TM

It works as follows:

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides. The "first" symbol means the "first unmarked" symbol.

The algorithmic meaning of the TM

It works as follows:

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides. The "first" symbol means the "first unmarked" symbol.

The algorithmic meaning of the TM

It works as follows:

$$\triangleleft \quad \overset{\cancel{0}}{\longleftarrow\!\!-\!\!-} \; w_1 \; \longrightarrow \; \# \; \overset{\cancel{0}}{\longleftarrow\!\!-\!\!-} \; w_2 \; \longrightarrow$$

$$\uparrow$$

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides. The "first" symbol means the "first unmarked" symbol.

The algorithmic meaning of the TM

It works as follows:

- Read the “first” symbol of w_1, “mark” it and “remember” it.
- Go to the “first” symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, “mark” it.
- Repeat until there is no more “unmarked” symbol on both sides.
  The “first” symbol means the “first unmarked” symbol.

The algorithmic meaning of the TM

It works as follows:

- Read the “first” symbol of w_1, “mark” it and “remember” it.
- Go to the “first” symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, “mark” it.
- Repeat until there is no more “unmarked” symbol on both sides. The “first” symbol means the “first unmarked” symbol.

The algorithmic meaning of the TM

It works as follows:

$$\triangleleft \quad \overset{\cancel{0}\ \ \cancel{1}}{\longleftarrow\!\!\!-\!\!\!-\!\!\!-} \; w_1 \; \longrightarrow \quad \# \quad \overset{\cancel{0}}{\longleftarrow\!\!\!-\!\!\!-\!\!\!-} \; w_2 \; \longrightarrow$$

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides. The "first" symbol means the "first unmarked" symbol.

The algorithmic meaning of the TM

It works as follows:

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides.
  The "first" symbol means the "first unmarked" symbol.

The algorithmic meaning of the TM

It works as follows:

- Read the “first” symbol of w_1, “mark” it and “remember” it.
- Go to the “first” symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, “mark” it.
- Repeat until there is no more “unmarked” symbol on both sides.
  The “first” symbol means the “first unmarked” symbol.

The algorithmic meaning of the TM

It works as follows:

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides. The "first" symbol means the "first unmarked" symbol.

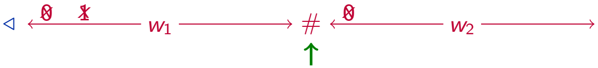
The algorithmic meaning of the TM

It works as follows:

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides.
  The "first" symbol means the "first unmarked" symbol.

The algorithmic meaning of the TM

It works as follows:



- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides. The "first" symbol means the "first unmarked" symbol.

The algorithmic meaning of the TM

It works as follows:

- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides. The "first" symbol means the "first unmarked" symbol.

The algorithmic meaning of the TM

It works as follows:

- Read the “first” symbol of w_1, “mark” it and “remember” it.
- Go to the “first” symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, “mark” it.
- Repeat until there is no more “unmarked” symbol on both sides.
  The “first” symbol means the “first unmarked” symbol.

The algorithmic meaning of the TM

It works as follows:

- Read the “first” symbol of w_1, “mark” it and “remember” it.
- Go to the “first” symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, “mark” it.
- Repeat until there is no more “unmarked” symbol on both sides. The “first” symbol means the “first unmarked” symbol.

The algorithmic meaning of the TM

It works as follows:

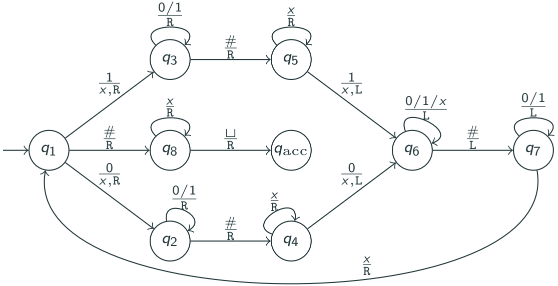
- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides. The "first" symbol means the "first unmarked" symbol.

The algorithmic meaning of the TM

It works as follows:

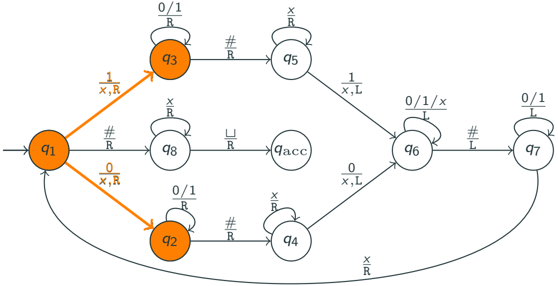
- Read the "first" symbol of w_1, "mark" it and "remember" it.
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).
- If it is different, reject immediately.
- If it is the same, "mark" it.
- Repeat until there is no more "unmarked" symbol on both sides. The "first" symbol means the "first unmarked" symbol.
- When there is no "unmarked" symbol on both sides, accept.

Comparison between the Turing machine and the algorithm, part 1



- Read the "first" symbol of w_1, "mark" it and "remember" it.

Comparison between the Turing machine and the algorithm, part 1

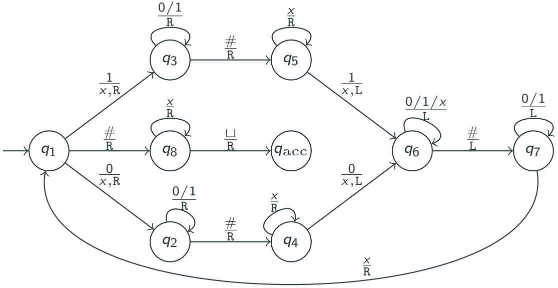


- Read the "first" symbol of w_1, "mark" it and "remember" it.

It goes to q_2 to "remember" that the symbol is 0.

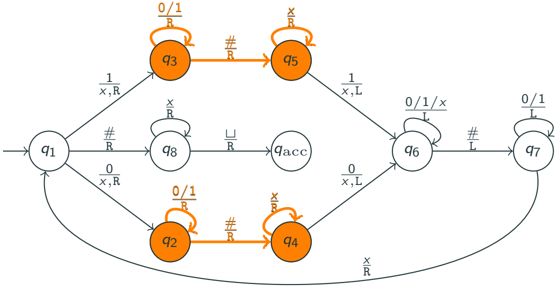
It goes to q_3 to "remember" that the symbol is 1.

Comparison between the Turing machine and the algorithm, part 2



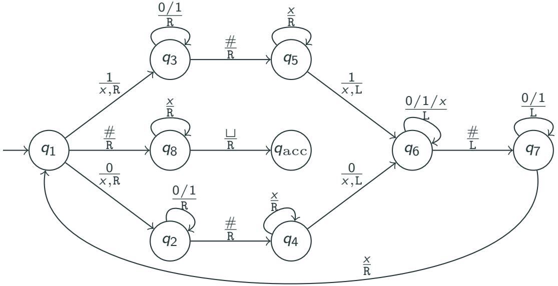
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).

Comparison between the Turing machine and the algorithm, part 2



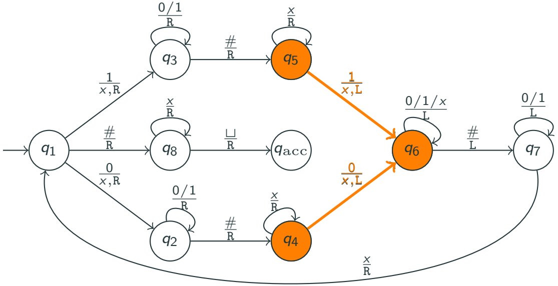
- Go to the "first" symbol of w_2 and check if it is the same (as the remembered symbol).

Comparison between the Turing machine and the algorithm, part 3



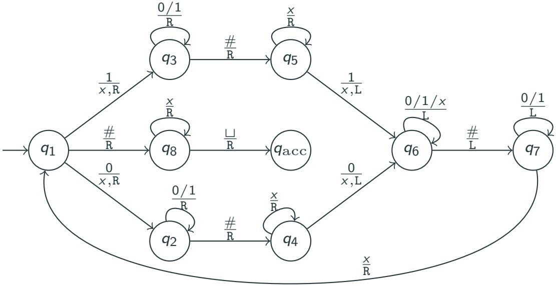
- If it is different, reject immediately.
- If it is the same, “mark” it.

Comparison between the Turing machine and the algorithm, part 3



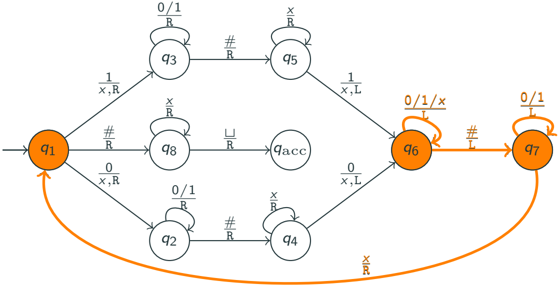
- If it is different, reject immediately.
- If it is the same, “mark” it.

Comparison between the Turing machine and the algorithm, part 4



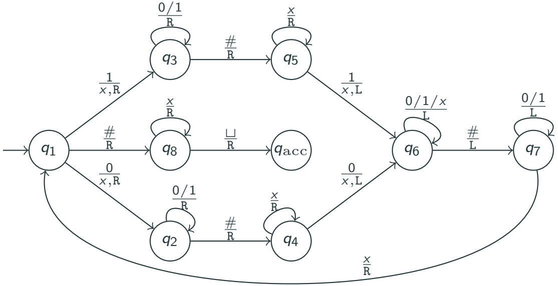
- Repeat until there is no more "unmarked" symbol on both sides.

Comparison between the Turing machine and the algorithm, part 4



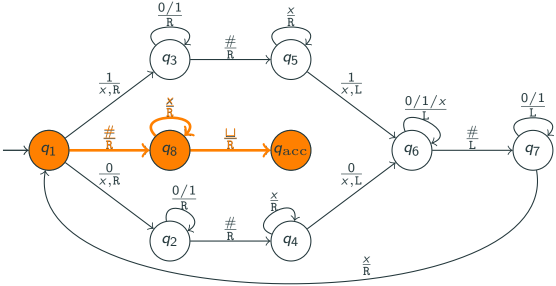
- Repeat until there is no more "unmarked" symbol on both sides.

Comparison between the Turing machine and the algorithm, part 5



- When there is no "unmarked" symbol on both sides, accept.

Comparison between the Turing machine and the algorithm, part 5



- When there is no "unmarked" symbol on both sides, accept.

To conclude:

Universally accepted convention

When we want to describe a Turing machine for a certain problem, it is sufficient to describe an algorithm in some "reasonable" format/pseudo-code.

To conclude:

Universally accepted convention

When we want to describe a Turing machine for a certain problem, it is sufficient to describe an algorithm in some "reasonable" format/pseudo-code.

Usually we only use the formal definition of Turing machine to prove "negative" results, i.e., to prove that certain problems do not have algorithms.

We will see more about this in Lesson 6.

To conclude:

Universally accepted convention
When we want to describe a Turing machine for a certain problem, it is sufficient to describe an algorithm in some "reasonable" format/pseudo-code.

Usually we only use the formal definition of Turing machine to prove "negative" results, i.e., to prove that certain problems do not have algorithms.

We will see more about this in Lesson 6.

Church-Turing thesis
Every "algorithm" is equivalent to a Turing machine.

Table of contents

The notion of "configuration" of a TM $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathbf{acc}}, q_{\mathbf{rej}}, \delta \rangle$

(Def.) A configuration of $\mathcal{M}$ is a string C over $Q \cup \Gamma$ which contains *exactly one symbol* from Q, i.e., a string of the form:

$$\triangleleft a_1 \cdots a_{i-1} \; p \; a_i \cdots a_m$$

The notion of "configuration" of a TM $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathbf{acc}}, q_{\mathbf{rej}}, \delta \rangle$

(Def.) A configuration of $\mathcal{M}$ is a string C over $Q \cup \Gamma$ which contains *exactly one symbol* from Q, i.e., a string of the form:

$$\triangleleft a_1 \cdots a_{i-1} \ p \ a_i \cdots a_m$$

This means the TM is in state p and the content of the tape is:

$$\triangleleft a_1 \cdots a_{i-1} a_i \cdots a_m \ \sqcup \sqcup \sqcup \cdots$$

The head is reading a_i.

The notion of “configuration” of a TM $\mathcal{M} = \langle \Sigma, \Gamma, Q, q_0, q_{\mathbf{acc}}, q_{\mathbf{rej}}, \delta \rangle$

(Def.) A configuration of $\mathcal{M}$ is a string C over $Q \cup \Gamma$ which contains *exactly one symbol* from Q, i.e., a string of the form:

$$\triangleleft a_1 \cdots a_{i-1} \ p \ a_i \cdots a_m$$

This means the TM is in state p and the content of the tape is:

$$\triangleleft a_1 \cdots a_{i-1} a_i \cdots a_m \ \sqcup \sqcup \sqcup \cdots$$

The head is reading a_i.

(Note) We use the symbol p to indicate the state of the TM and the position of the head.

The initial, accepting, rejecting and halting configurations

(Def.) On input word $w \in \Sigma^*$, the *initial* configuration of $\mathcal{M}$ on w is:

$$\triangleleft q_0 w$$

The initial, accepting, rejecting and halting configurations

(Def.) On input word $w \in \Sigma^*$, the *initial* configuration of $\mathcal{M}$ on w is:

$$\triangleleft q_0 w$$

(Def.) A configuration is *accepting*, if it contains q_{acc}.

The initial, accepting, rejecting and halting configurations

(Def.) On input word $w \in \Sigma^*$, the *initial* configuration of $\mathcal{M}$ on w is:

$$\triangleleft q_0 w$$

(Def.) A configuration is *accepting*, if it contains q_{acc}.

(Def.) It is called *rejecting*, if it contains q_{rej}.

The initial, accepting, rejecting and halting configurations

(Def.) On input word $w \in \Sigma^*$, the *initial* configuration of $\mathcal{M}$ on w is:

$$\triangleleft q_0 w$$

(Def.) A configuration is *accepting*, if it contains q_{acc}.

(Def.) It is called *rejecting*, if it contains q_{rej}.

(Def.) A *halting* configuration is either an accepting or a rejecting configuration.

The “run” of a Turing machine

(Def.) For two configurations C and C':

$$C \ \vdash \ C'$$

denotes that C' is the configuration obtained from applying (the applicable) transition on C. (See Note 5 for the detailed definition.)

The “run” of a Turing machine

(Def.) For two configurations C and C':

$$C \ \vdash \ C'$$

denotes that C' is the configuration obtained from applying (the applicable) transition on C. (See Note 5 for the detailed definition.)

(Def.) *The run of $\mathcal{M}$ on w* is the (possibly infinite) sequence:

$$C_0 \ \vdash \ C_1 \ \vdash \ C_2 \ \vdash \ \cdots$$

where C_0 is the initial configuration of $\mathcal{M}$ on w.

The "run" of a Turing machine

(Def.) For two configurations C and C':

$$C \vdash C'$$

denotes that C' is the configuration obtained from applying (the applicable) transition on C. (See Note 5 for the detailed definition.)

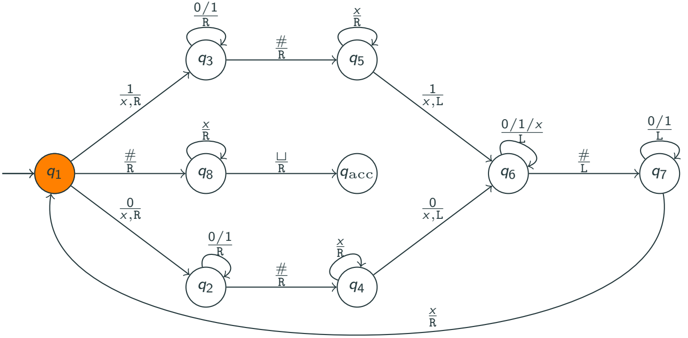
(Def.) *The run of $\mathcal{M}$ on w* is the (possibly infinite) sequence:

$$C_0 \vdash C_1 \vdash C_2 \vdash \cdots$$

where C_0 is the initial configuration of $\mathcal{M}$ on w.

The run is finite when it ends with a halting configuration, i.e., when the TM reaches either q_{acc} or q_{rej}.

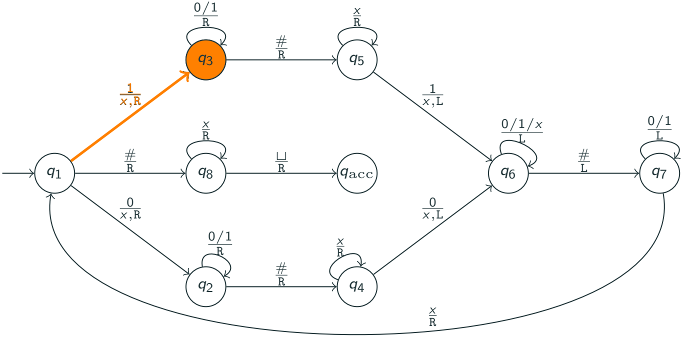
Example: The run of our TM on input 10#00



The run of $\mathcal{M}$ on 10#00:

$\triangleleft q_1 10\#00$

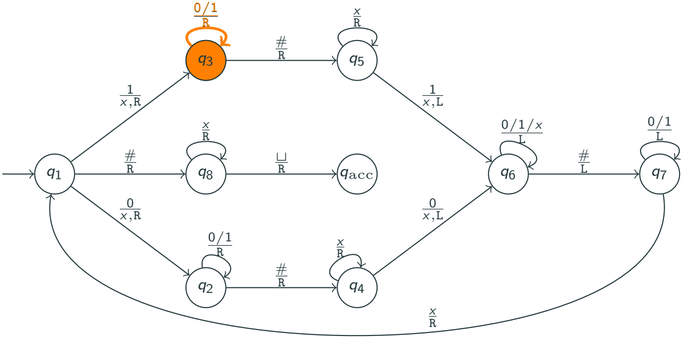
Example: The run of our TM on input 10#00



The run of $\mathcal{M}$ on 10#00:

$\triangleleft q_1 10\#00 \vdash \triangleleft x q_3 0\#00$

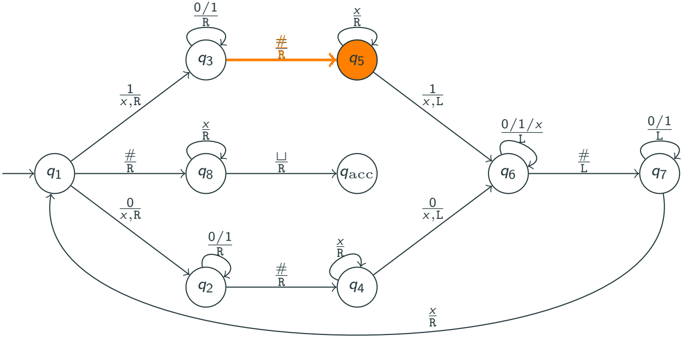
Example: The run of our TM on input $10\#00$



The run of $\mathcal{M}$ on $10\#00$:

$\triangleleft q_1 10\#00 \vdash \triangleleft x q_3 0\#00 \vdash \triangleleft x 0 q_3 \#00$

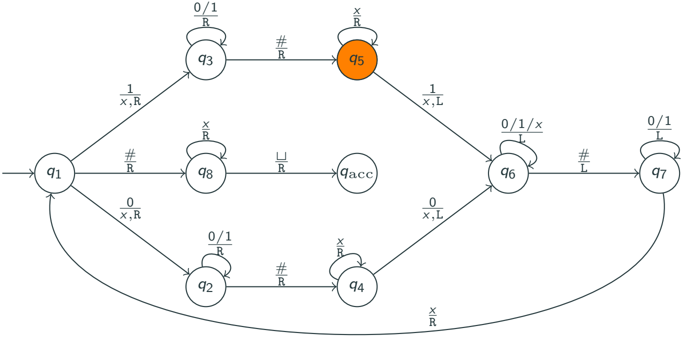
Example: The run of our TM on input $10\#00$



The run of $\mathcal{M}$ on $10\#00$:

$\triangleleft q_1 10\#00 \vdash \triangleleft x q_3 0\#00 \vdash \triangleleft x0 q_3 \#00 \vdash \triangleleft x0\# q_5 00$

Example: The run of our TM on input $10\#00$



The run of $\mathcal{M}$ on $10\#00$:

$\triangleleft q_1 10\#00 \vdash \triangleleft x q_3 0\#00 \vdash \triangleleft x0 q_3 \#00 \vdash \triangleleft x0\# q_5 00 \vdash \triangleleft x0\#0 q_{\rm rej} 0.$

The notion of acceptance and rejection by a Turing machine

(Def.) We say that $\mathcal{M}$ halts on w, if the run of $\mathcal{M}$ on w is finite.

The notion of acceptance and rejection by a Turing machine

(Def.) We say that $\mathcal{M}$ halts on w, if the run of $\mathcal{M}$ on w is finite.

Recall that a finite run implies that it ends in a halting configuration.

The notion of acceptance and rejection by a Turing machine

(Def.) We say that $\mathcal{M}$ halts on w, if the run of $\mathcal{M}$ on w is finite.

Recall that a finite run implies that it ends in a halting configuration.

(Def.) If $\mathcal{M}$ halts in an accepting configuration, i.e., the run ends in an accepting configuration, we say that *$\mathcal{M}$ accepts w*.

The notion of acceptance and rejection by a Turing machine

(Def.) We say that $\mathcal{M}$ halts on w, if the run of $\mathcal{M}$ on w is finite.

Recall that a finite run implies that it ends in a halting configuration.

(Def.) If $\mathcal{M}$ halts in an accepting configuration, i.e., the run ends in an accepting configuration, we say that $\mathcal{M}$ *accepts* w.

(Def.) If it halts in a rejecting configuration, i.e., the run ends in a rejecting configuration, we say that $\mathcal{M}$ *rejects* w.

Recognizable languages

(Def.) We say that $\mathcal{M}$ *recognizes* a language L, if for every input word w:

- if $w \in L$, then $\mathcal{M}$ accepts w,
- if $w \notin L$, then $\mathcal{M}$ does not accept w.

Note that "$\mathcal{M}$ does not accept w" have two meanings: either $\mathcal{M}$ rejects w, or $\mathcal{M}$ does not halt on w.

Recognizable languages

(Def.) We say that $\mathcal{M}$ *recognizes* a language L, if for every input word w:

- if $w \in L$, then $\mathcal{M}$ accepts w,
- if $w \notin L$, then $\mathcal{M}$ does not accept w.

Note that "$\mathcal{M}$ does not accept w" have two meanings: either $\mathcal{M}$ rejects w, or $\mathcal{M}$ does not halt on w.

(Def.) A language L is *recognizable*/*recursively enumerable* (r.e.), if there is a TM $\mathcal{M}$ that recognizes L.

Decidable languages

(Def.) We say that $\mathcal{M}$ *decides* a language L, if for every input word w:

- if $w \in L$, then $\mathcal{M}$ accepts w,
- if $w \notin L$, then $\mathcal{M}$ rejects w.

Note that this implies $\mathcal{M}$ halts on every word $w \in \Sigma^*$.

Decidable languages

(Def.) We say that $\mathcal{M}$ *decides* a language L, if for every input word w:

- if $w \in L$, then $\mathcal{M}$ accepts w,
- if $w \notin L$, then $\mathcal{M}$ rejects w.

Note that this implies $\mathcal{M}$ halts on every word $w \in \Sigma^*$.

(Def.) A language L is *decidable/recursive*, if there is a TM $\mathcal{M}$ that decides L.

Otherwise, it is called *undecidable*.

Decidable languages

(Def.) We say that $\mathcal{M}$ *decides* a language L, if for every input word w:

- if $w \in L$, then $\mathcal{M}$ accepts w,
- if $w \notin L$, then $\mathcal{M}$ rejects w.

Note that this implies $\mathcal{M}$ halts on every word $w \in \Sigma^*$.

(Def.) A language L is *decidable/recursive*, if there is a TM $\mathcal{M}$ that decides L.

Otherwise, it is called *undecidable*.

(Remark) Every decidable language is a recognizable language, but there is a recognizable language that is not decidable. (We will see this language in the next few weeks.)

End of Lesson 5