CS301: Theory of Computation
Section 3.1 - Turing Machines

Daniel White

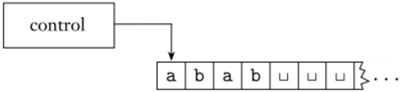
Gettysburg College
Computer Science

Spring 2024

With the main models of computation proposed and developed so far, both finite automata and pushdown automata, many simple tasks are still beyond our capabilities.

e.g. PDAs are more powerful than DFAs, but still can not recognize/compute the prime numbers or the powers of 2*.*

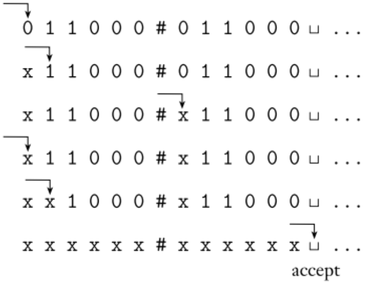
First proposed by Alan Turing in 1936, the **Turing machine** is similar to a DFA, but with *unlimited* and *unrestricted memory*. Such machines can do everything that a real computer can do.



The following list summarizes the differences between finite automata and Turing machines.

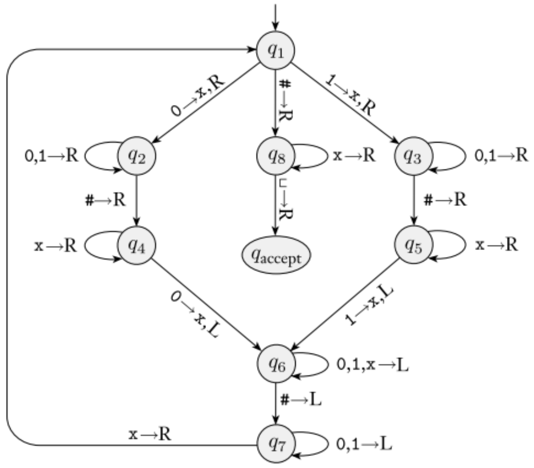
1. A Turing machine can both write on the tape and read from it.
2. The read-write head can move both to the left and to the right.
3. The tape is infinitely long to the right.
4. The special states for rejecting and accepting take effect immediately.

exercise: Convince yourself that a Turing machine, as described so far, can recognize the language $B = \{w\#w \mid w \in \{0,1\}^*\}$.



0 1 1 0 0 0 # 0 1 1 0 0 0 ␣ ...
x 1 1 0 0 0 # 0 1 1 0 0 0 ␣ ...
x 1 1 0 0 0 # x 1 1 0 0 0 ␣ ...
x 1 1 0 0 0 # x 1 1 0 0 0 ␣ ...
x x 1 0 0 0 # x 1 1 0 0 0 ␣ ...
x x x x x x # x x x x x x ␣ ...
accept

exercise: Convince yourself that B is not a CFL. What does this seem to imply?


q_1
0→x,R
#→R
1→x,R
0,1→R
q_2
q_8
x→R
q_3
0,1→R
#→R
⊔→R
#→R
x→R
q_4
q_{accept}
q_5
x→R
0→x,L
1→x,L
q_6
0,1,x→L
#→L
x→R
q_7
0,1→L

Definition 3.3 (pg. 168)

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. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ 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 $q_{\text{accept}} \neq q_{\text{reject}}$.

homework: 3.2abd and 3.5

Running a TM on an input string can result in three possible outcomes. The machine may *accept*, *reject*, or *loop*.

Looping can result from simple to complex behavior never leading to a halt state.

Failure to accept can occur via rejecting or looping. Distinguishing a machine that is looping from one taking a long time is difficult.

Turing machines that halt on all inputs are called **deciders**. Examples of Turing-recognizable, yet not decidable, languages will be presented in future chapters.

Definition 3.5 (pg. 170)

Call a language **Turing-recognizable** if some Turing machine recognizes it.

Definition 3.6 (pg. 170)

Call a language **Turing-decidable**, or simply decidable, if some Turing machine decides it.

$\boxed{\text{homework: 3.15ab and 3.16ab}}$

exercise: Describe a TM, M_2, that decides $A = \{0^{2^n} \mid n \geq 0\}$.

On input string w:

1. Sweep left to right across the tape, crossing off every other 0.
2. If in stage 1 the tape contained a single 0, accept.
3. If in stage 1 the tape contained more than a single 0 and the number of 0s was odd, *reject*.
4. ...

exercise: Convince yourself that $B = \{0^q \mid q \text{ is prime}\}$ is Turing-recognizable. *Hint:* Consider $\{0^n 1^k \mid k \text{ divides } n\}$

exercise: Describe a TM, M_3, which decides the language

$$C = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k \mid i \cdot j = k \text{ and } i, j, k \geq 1\}.$$

On input string w:

1. Scan from left to right to determine whether input string is a member of $\mathtt{a}^+\mathtt{b}^+\mathtt{c}^+$. Reject if not.
2. Return head to the left-hand end of tape.
3. . . .

$\boxed{\text{homework: example 3.12}}$