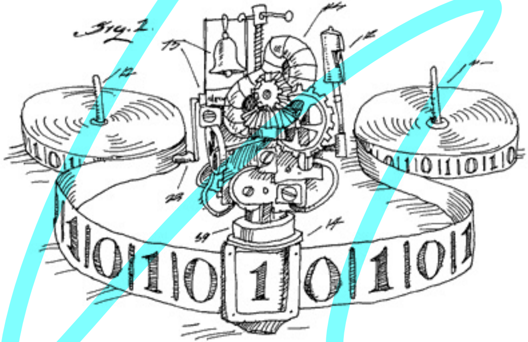
Automata, Languages and Computation

Chapter 8 : Turing Machines

Master Degree in Computer Engineering
University of Padua
Lecturer : Giorgio Satta

Lecture based on material originally developed by :
Gösta Grahne, Concordia University

Turing machines



1. Turing machine (TM) : formal model of computer algorithms that allows the mathematical study of computability
2. Programming techniques for TM : techniques to facilitate the writing of programs for TM
3. TM Extensions : machines that are more complex than TM but with the same computational capacity
4. TM with restrictions : automata that are simpler than TM but with the same computational capacity

Turing machine

In order to mathematically study undecidabilty we need a **simple** formalism to represent programs (Python is **not** suitable)

Historically used formalisms:

- predicate calculus (Gödel, 1931)
- partial recursive functions (Kleene, 1936)
- lambda calculus (Church, 1936)
- Turing machine (Turing, 1936)

Turing machine

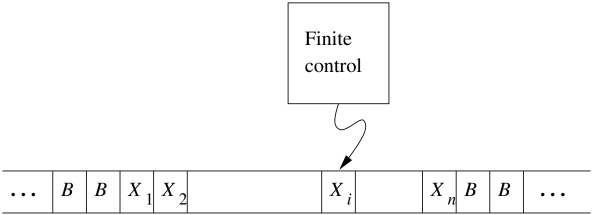
A Turing machine is a finite state automaton with the addition of a **memory tape** with

- sequential access
- unlimited capacity in both tape directions

Differently from the PDA model, input string is initially placed into the auxiliary memory

The Turing machine model allows the study of computability properties such as **undecidabilty** and **intractability**

Turing machine



Informally, a Turing machine performs a move according to its state and the symbol which is read by the tape head

In a single move, a Turing machine

- changes its state
- writes a new symbol in the cell read by the tape head
- moves the tape head to the cell to the right or to the left

Turing machine

A **Turing machine**, MT for short, is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F),$$

where

- Q is a finite set of **states**
- Σ is a finite set of **input symbols**
- Γ is a finite set of **tape symbols**, with $\Sigma \subseteq \Gamma$
- δ is a **transition function** from $Q \times \Gamma$ to $Q \times \Gamma \times \{L, R\}$
- q_0 is the **initial state**
- $B \in \Gamma$ is the **blank symbol**, with $B \notin \Sigma$
- $F \subseteq Q$ is the set of **final states**

Note that the automaton is deterministic, and it has no 'stand' move

Instantaneous descriptions

A TM changes its configuration with each move. We use the notion of instantaneous description (ID) to describe configurations

An **instantaneous description** (ID) of M is a string of the form

$$X_1 X_2 \cdots X_{i-1} q X_i X_{i+1} \cdots X_n$$

where

- q is M's state
- $X_1 X_2 \cdots X_n$ is the "visited" portion of M's tape
- the tape head of M is reading the i-th tape symbol

Computation of a TM

To represent a **computation step** of M we use the binary relation $\underset{M}{\vdash}$ defined on the set of IDs

If $\delta(q, X_i) = (p, Y, L)$, then

$$X_1X_2 \cdots X_{i-1}qX_iX_{i+1} \cdots X_n \underset{M}{\vdash} X_1X_2 \cdots pX_{i-1}YX_{i+1} \cdots X_n$$

If $\delta(q, X_i) = (p, Y, R)$, then

$$X_1X_2 \cdots X_{i-1}qX_iX_{i+1} \cdots X_n \underset{M}{\vdash} X_1X_2 \cdots X_{i-1}YpX_{i+1} \cdots X_n$$

Special cases if the tape head is at the two ends of the written tape

Computation of a TM

To represent the **computations** of M, we use the reflexive and transitive closure of $\underset{M}{\vdash}$, written $\underset{M}{\overset{*}{\vdash}}$

For input string $w \in \Sigma^*$, the initial ID is $q_0 w$

For a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$, an accepting computation has the form

$$q_0 w \underset{M}{\overset{*}{\vdash}} \alpha p \beta$$

with $p \in F$ and $\alpha, \beta \in \Gamma^*$

We will come back to this definition after some examples

Example

Let us specify a TM M with $L(M) = \{0^n 1^n \mid n \geqslant 1\}$

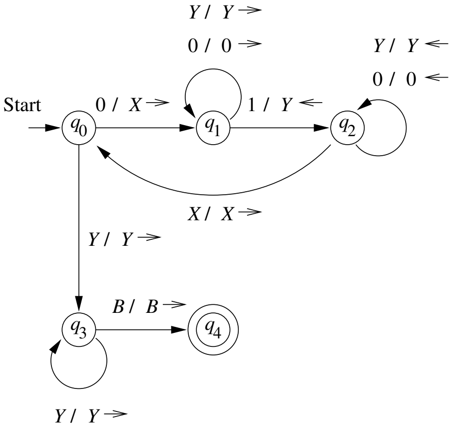
$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$$

The transition function δ is represented by the following table

	0	1	X	Y	B
$\rightarrow q_0$	(q_1, X, R)			(q_3, Y, R)	
q_1	$(q_1, 0, R)$	(q_2, Y, L)		(q_1, Y, R)	
q_2	$(q_2, 0, L)$		(q_0, X, R)	(q_2, Y, L)	
q_3				(q_3, Y, R)	(q_4, B, R)
$\star q_4$					

Example

We can also represent δ by means of the following **transition diagram**



Example

If input w has the form $\mathbf{0}^*\mathbf{1}^*$, then at each ID the tape is of the form $\boldsymbol{X}^*\mathbf{0}^*\boldsymbol{Y}^*\mathbf{1}^*$

M implements the following strategy

- in q_0 it replaces the leftmost 0 with X and moves to q_1
- in q_1 it proceeds from left to right, goes over 0 and Y looking for the leftmost 1, replaces it with Y and moves to q_2
- in q_2 it proceeds from right to left, goes over Y and 0 looking for the rightmost X, and moves back to q_0
- in q_0, if it finds one more 0 it resumes the above cycle, otherwise it moves to q_3
- in q_3 it overrides all of the Y's and accepts if there is no 1

Observe how input string is overwritten during the computation

Example

Given the string input 0011, M performs the following computation (sequence of ID)

$$\begin{aligned}
q_0 0011 &\vdash Xq_1 011 \vdash X0q_1 11 \\
&\vdash Xq_2 0Y1 \vdash q_2 X0Y1 \\
&\vdash Xq_0 0Y1 \vdash XXq_1 Y1 \\
&\vdash XXYq_1 1 \vdash XXq_2 YY \\
&\vdash Xq_2 XYY \vdash XXq_0 YY \\
&\vdash XXYq_3 Y \vdash XXYYq_3 B \\
&\vdash XXYYBq_4 B
\end{aligned}$$

TM with "output"

We have defined a TM as a recognition device. Alternatively, we can use these devices to compute **functions** on natural numbers.

Historically, this was the original definition by A. Turing

We encode each natural number in **unary notation** according to the scheme

$$n =_1 0^n$$

Example

The following TM M computes the **proper subtractor** function

$$m \dot{-} n = max(m - n, 0)$$

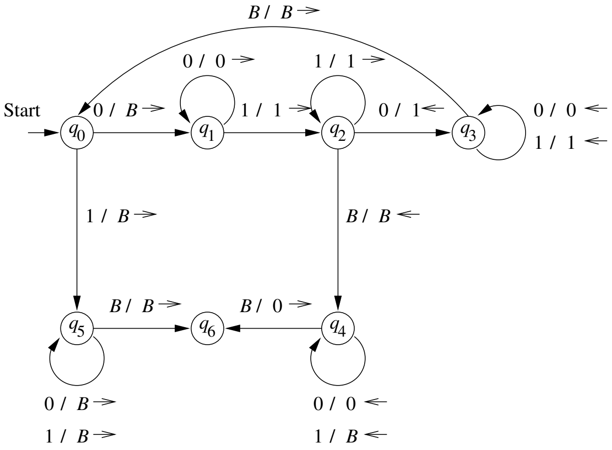
starting with 0^m10^n on its tape and **halting** with $0^{m \dot{-} n}$.

No set of final states for TMs with output

	0	1	B
$\rightarrow q_0$	(q_1, B, R)	(q_5, B, R)	
q_1	$(q_1, 0, R)$	$(q_2, 1, R)$	
q_2	$(q_3, 1, L)$	$(q_2, 1, R)$	(q_4, B, L)
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_0, B, R)
q_4	$(q_4, 0, L)$	(q_4, B, L)	$(q_6, 0, R)$
q_5	(q_5, B, R)	(q_5, B, R)	(q_6, B, R)
$\star q_6$			

Example

The transition diagram is



Example

The TM M performs the following loop

- find the leftmost 0 and replace with B (states q_0, q_3)
- search right for the first 0 placed after symbols 1, and replace it with 1 (states q_1, q_2)

The loop ends in two possible ways

- M cannot find a 0 to the right of the 1's ($m > n$); then M turns all of the 1's into a single 0 followed by B's
- M cannot find a 0 to be replaced by B ($m \leqslant n$); then $m \dot{-} n = 0$ and M replaces all 0's and 1's into B

Notation for TM

We use notational **conventions** similar to those of other automata

- a, b, c, ..., a_1, a_2, ..., a_i, ... input symbols
- X, Y, Z tape symbols
- u, w, x, y, z strings over the input alphabet
- α, β, γ, ..., strings over tape alphabet
- p, q, r, ..., q_1, q_2, ..., q_i, ... states

Exercise

The TM $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_2\})$ has the following transitions :

$$\delta(q_0, 0) = (q_1, 1, R)$$
$$\delta(q_1, 1) = (q_2, 0, L)$$
$$\delta(q_2, 1) = (q_0, 1, R)$$

Specify the computation (ID sequence) of M for input 0100

Exercise

Provide TMs for the following languages by specifying the transition diagram and by briefly explaining the adopted strategy

- $L = \{a^n b^{2n} \mid n \geqslant 1\}$
- $L = \{w \in \{a, b, c\}^* \mid \#_a(w) = \#_b(w) = \#_c(w)\}$
- $L = \{a^n b^{2k} a^n \mid b, k \geqslant 0\}$

Language accepted by a TM

A TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ **accepts** the language

$$L(M) = \{w \mid w \in \Sigma^*,\ q_0 w \overset{*}{\underset{M}{\vdash}} \alpha p \beta,\ p \in F,\ \alpha, \beta \in \Gamma^*\}$$

The class of languages accepted by TMs is called **recursively enumerable** (RE)

This term derives from formalisms that historically preceded TM