Chapter 5
Turing Machines

5.1 Introduction

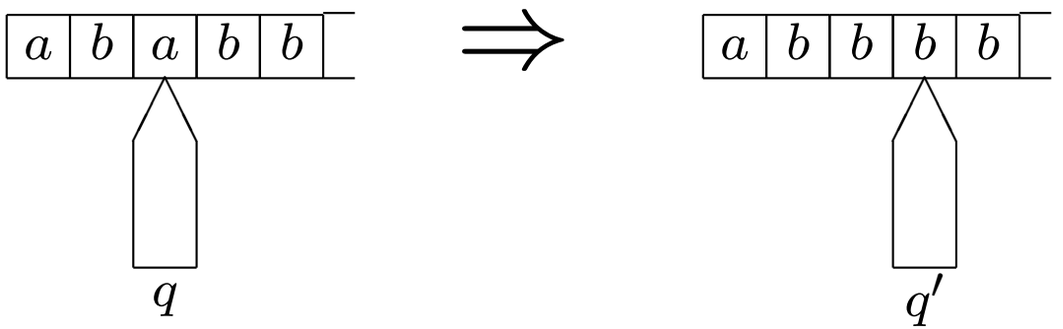
- The language $a^n b^n c^n$ cannot be accepted by a pushdown automaton.
- Machines with an infinite memory, which is not restricted to LIFO access.
- Model of the concept of effective procedure.
- Justification : extensions are not more powerful; other formalizations are equivalent.

5.2 Definition

- Infinite memory viewed as a tape divided into cells that can hold one character of a tape alphabet.
- Read head.
- Finite set of states, accepting states.
- transition function that, for each state and tape symbol pair gives
  - the next state,
  - a character to be written on the tape,
  - the direction (left or right) in which the read head moves by one cell.

Execution

- Initially, the input word is on the tape, the rest of the tape is filled with "blank" symbols, the read head is on the leftmost cell of the tape.
- At each step, the machine
  - reads the symbol from the cell that is under the read head,
  - replaces this symbol as specified by the transition function,
  - moves the read head one cell to the left or to the right, as specified by the transition function.
  - changes state as described by the transition function,
- the input word is accepted as soon as an accepting state is reached.


a b a b b
q
a b b b b
q′

Formalization

7-tuple $M = (Q, \Gamma, \Sigma, \delta, s, B, F)$, where:

- Q is a finite set of states,
- Γ is the tape alphabet,
- $\Sigma \subseteq \Gamma$ is the input alphabet,
- $s \in Q$ is the initial state,
- $F \subseteq Q$ is the set of accepting states,
- $B \in \Gamma - \Sigma$ is the "blank symbol" (#),
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function.

Configuration

The required information is:

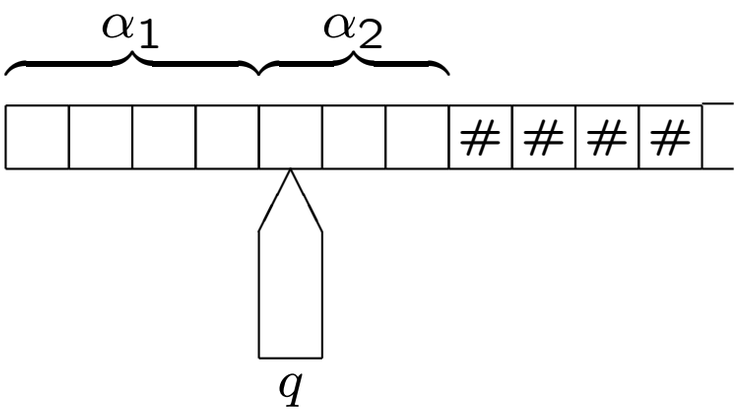
1. The state,
2. the tape content,
3. the position of the read head.

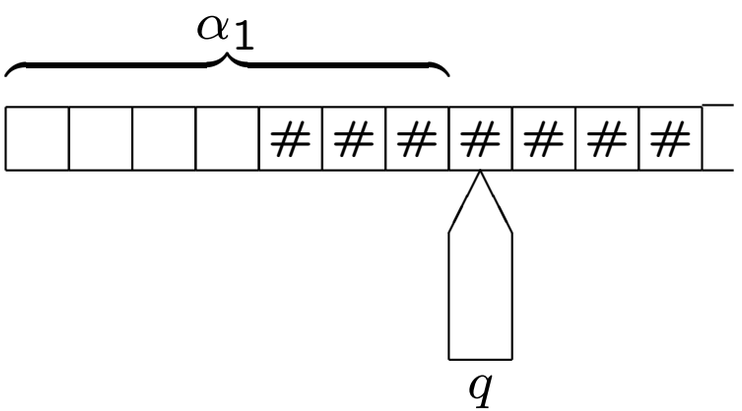
Representation : 3-tuple containing

1. the state of the machine,
2. the word found on the tape up to the read head,
3. the word found on the tape from the read head on.

Formally, a configuration is an element of $Q \times \Gamma^* \times (\varepsilon \cup \Gamma^*(\Gamma - \{B\}))$.

Configurations (q, α_1, α_2) and $(q, \alpha_1, \varepsilon)$.





Derivation

Configuration (q, α_1, α_2) written as $(q, \alpha_1, b\alpha_2')$ with $b = \#$ if $\alpha_2 = \varepsilon$.

- If $\delta(q, b) = (q', b', R)$ we have

$$(q, \alpha_1, b\alpha_2') \vdash_M (q', \alpha_1 b', \alpha_2').$$

- If $\delta(q, b) = (q', b', L)$ and if $\alpha_1 \neq \varepsilon$ and is thus of the form $\alpha_1' a$ we have

$$(q, \alpha_1' a, b\alpha_2') \vdash_M (q', \alpha_1', ab'\alpha_2').$$

Derivation

A configuration C' is derivable in several steps from the configuration C by the machine M ($C \vdash_M^* C'$) if there exists $k \geq 0$ and intermediate configurations $C_0, C_1, C_2, \ldots, C_k$ such that

- $C = C_0$,
- $C' = C_k$,
- $C_i \vdash_M C_{i+1}$ for $0 \leq i < k$.

The language $L(M)$ accepted by the Turing machine is the set of words w such that

$$(s, \varepsilon, w) \vdash_M^* (p, \alpha_1, \alpha_2), \text{ with } p \in F.$$

Example

Turing machine $M = (Q, \Gamma, \Sigma, \delta, s, B, F)$ with

- $Q = \{q_0, q_1, q_2, q_3, q_4\}$,
- $\Gamma = \{a, b, X, Y, \#\}$,
- $\Sigma = \{a, b\}$,
- $s = q_0$,
- $B = \#$,
- δ given by

	a	b	X	Y	$\#$
q_0	(q_1, X, R)	$-$	$-$	(q_3, Y, R)	$-$
q_1	(q_1, a, R)	(q_2, Y, L)	$-$	(q_1, Y, R)	$-$
q_2	(q_2, a, L)	$-$	(q_0, X, R)	(q_2, Y, L)	$-$
q_3	$-$	$-$	$-$	(q_3, Y, R)	$(q_4, \#, R)$
q_4	$-$	$-$	$-$	$-$	$-$

M accepts $a^n b^n$. For example, its execution on $aaabbb$ is

$(q_0, \varepsilon, aaabbb)$	$\vdots$
$(q_1, X, aabbb)$	$(q_1, XXXYY, b)$
$(q_1, Xa, abbb)$	$(q_2, XXXY, YY)$
(q_1, Xaa, bbb)	(q_2, XXX, YYY)
$(q_2, Xa, aYbb)$	$(q_2, XX, XYYY)$
$(q_2, X, aaYbb)$	(q_0, XXX, YYY)
$(q_2, \varepsilon, XaaYbb)$	$(q_3, XXXY, YY)$
$(q_0, X, aaYbb)$	$(q_3, XXXYY, Y)$
$(q_1, XX, aYbb)$	$(q_3, XXXYYY, \varepsilon)$
	$(q_4, XXXYYY\#, \varepsilon)$

Accepted language
Decided language

Turing machine = effective procedure ? Not always. The following situation are possible.

1. The sequence of configurations contains an accepting state.
2. The sequence of configurations ends because either
   - the transition function is not defined, or
   - it requires a left move from the first cell on the tape.
3. The sequence of configurations never goes through an accepting state and is infinite.

In the first two cases, we have an effective procedure, in the third not.

The *execution* of a Turing machine on a word w is the maximal sequence of configurations

$$(s, \varepsilon, w) \vdash_M C_1 \vdash_M C_2 \vdash_M \cdots \vdash_M C_k \vdash_M \ldots$$

i.e., the sequence of configuration that either

- is infinite,
- ends in a configuration in which the state is accepting, or
- ends in a configuration from which no other configuration is derivable.

Decided language: A language L is decided by a Turing machine M if

- M accepts L,
- M has no infinite executions.

Decided Language

Deterministic Automata!

- Deterministic finite automata: the accepted and decided languages are the same.
- Nondeterministic finite automata: meaningless.
- Nondeterministic pushdown automata: meaningless, but the context-free languages can be decided by a Turing machine.
- Deterministic pushdown automata : the accepted language is decided, except if infinite executions exist (loops with only ε transitions).

Other definitions of Turing machines

1. Single *stop state* and a transition function that is defined everywhere. In the stop state; the result is placed on the tape: tape : "accepts" (1) or "does not accept" (0).

2. Two stop states: q_Y and q_N, and a transition function that is defined everywhere.

Recursive and recursively enumerable languages

A language is *recursive* if it is decided by a Turing machine.

A language is *recursively enumerable* if it is accepted by a Turing machine.

The Turing-Church thesis

The languages that can be recognized by an effective procedure are those that are decided by a Turing machine.

Justification.

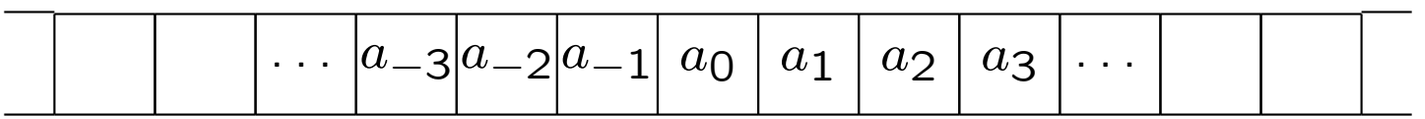
1. If a language is decided by a Turing machine, it is computable: clear.

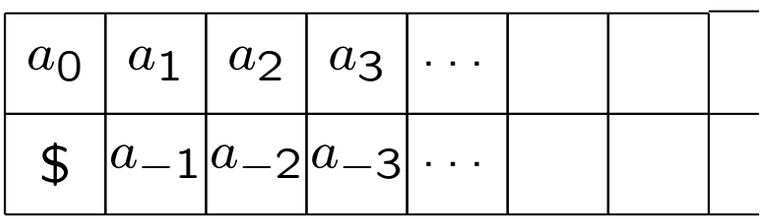
2. If a language is computable, it is decided by a Turing machine:

   - Extensions of Turing machines and other machines.
   - Other models.

Extensions of Turing machines

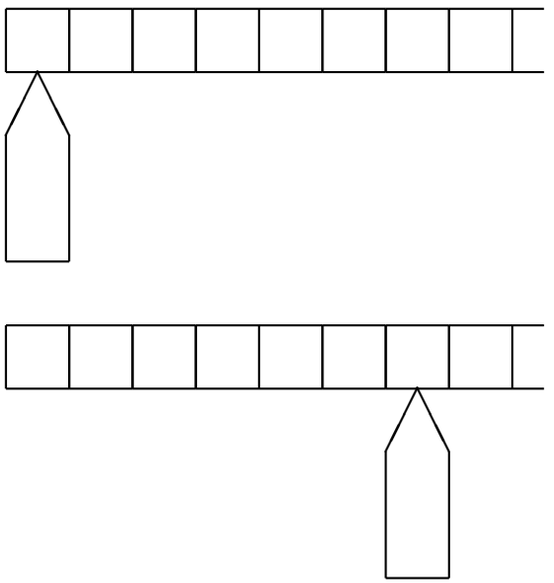
Tape that is infinite in both directions





Multiple tapes

Several tapes and read heads:

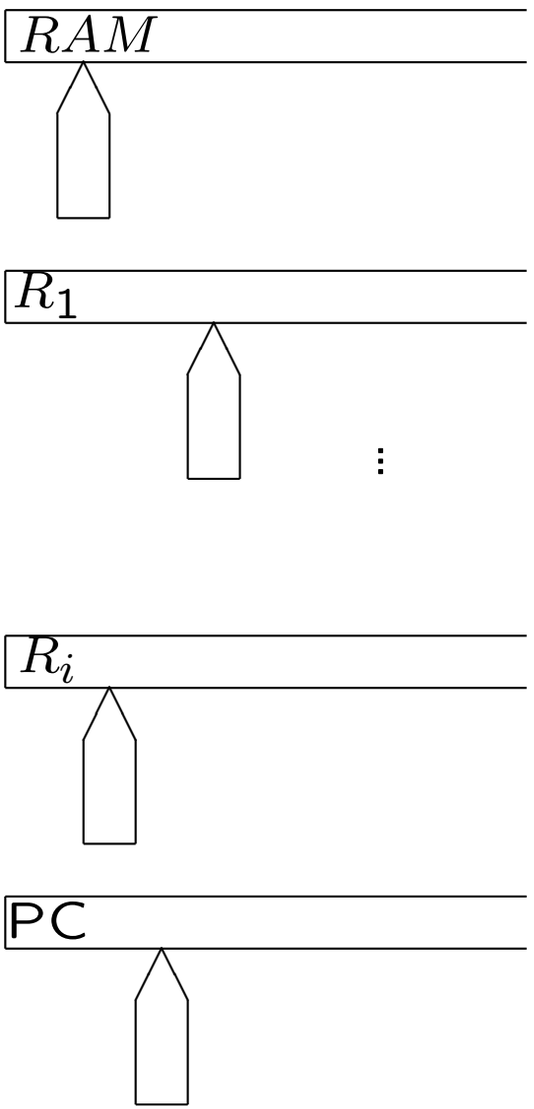


Simulation (2 tapes) : alphabet = 4-tuple

- Two elements represent the content of the tapes,
- Two elements represent the position of the read heads.

Machines with RAM

One tape for the memory, one for each register.



Simulation :

#	0	$*$	v_0	#	1	$*$	v_1	#	...	#	a	d	d	i	$*$	v_i	#

Nondeterministic Turing machines

Transition relation :

$$\Delta : (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R\})$$

The execution is no longer unique.

Eliminating non-determinism

Theorem

Any language that is accepted by a nondeterministic Turing machine is also accepted by a deterministic Turing machine.

Proof

Simulate the executions in increasing-length order.

$$r = \max_{q \in Q, a \in \Gamma} |\{((q, a), (q', x, X)) \in \Delta\}|.$$

Three tape machine:

1. The first tape holds the input word and is not modified.
2. The second tape will hold sequences of numbers less than r.
3. The third tape is used by the deterministic machine to simulate the nondeterministic one.

The deterministic machine proceeds as follows.

1. On the second tape it generates all finite sequences of numbers less than r. These sequences are generated in increasing length order.

2. For each of these sequences, it simulates the nondeterministic machine, the choice being made according to the sequence of numbers.

3. It stops as soon as the simulation of an execution reaches an accepting state.

Universal Turing machines

A Turing machine that can simulate any Turing machine.

- Turing machine M.
- Data for M : M' and a word w.
- M simulates the execution of M' on w.

Turing machine computable functions

A Turing machine computes a function $f : \Sigma^* \to \Sigma^*$ if, for any input word w, it always stops in a configuration where $f(w)$ is on the tape.

The functions that are computable by an effective procedure are those that are computable by a Turing machine