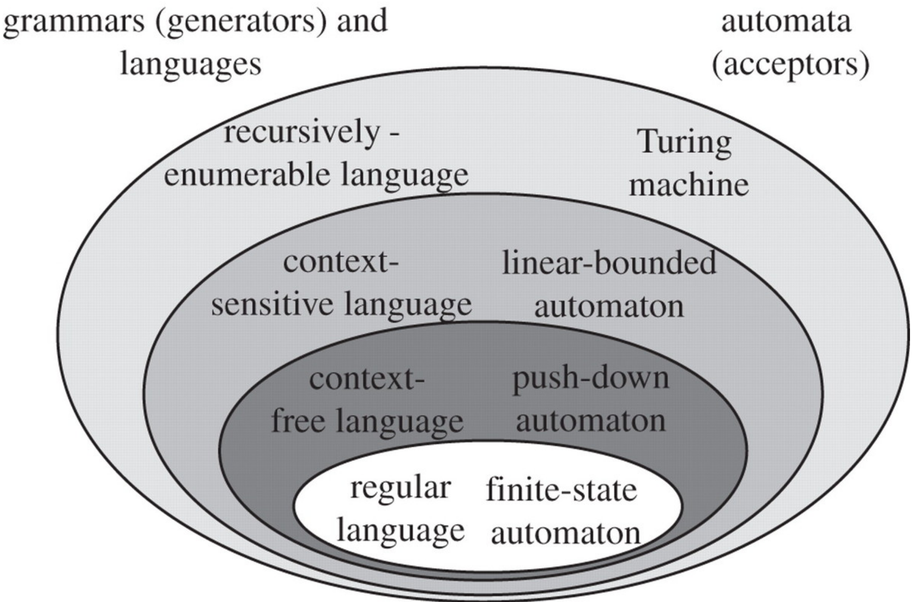
COS210 - Theoretical Computer Science
Turing Machines and the Church-Turing Thesis (Part 1)

Overview


grammars (generators) and languages
automata (acceptors)
recursively - enumerable language
Turing machine
context-sensitive language
linear-bounded automaton
context-free language
push-down automaton
regular language
finite-state automaton

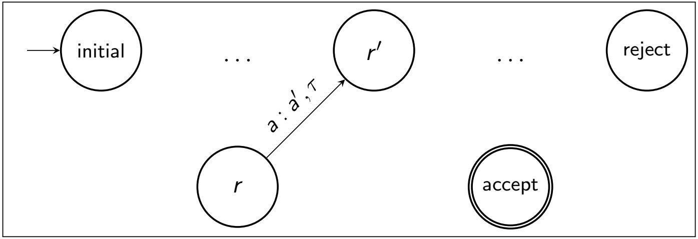
Turing Machines

Originally proposed by Alan Turing in 1936

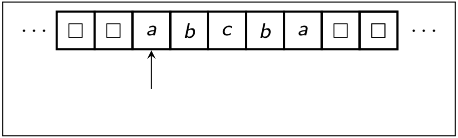
Fundamental to Computer Science. If something cannot be computed with a turning machine it cannot be computed on a computer!

Turing Machine

state control



tape

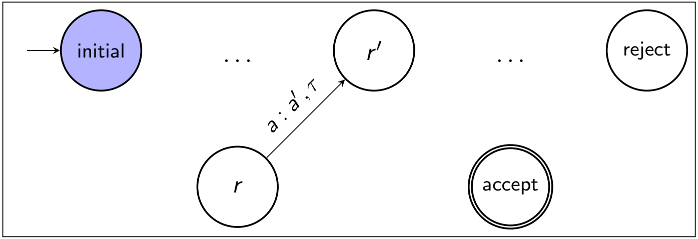


stores the input string

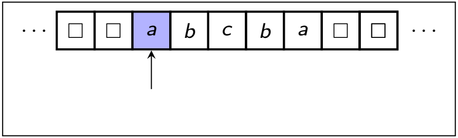
$M = (Q, \Sigma, \Gamma, \delta, q, q_{accept}, q_{reject})$

- set of states Q
- input alphabet Σ
- tape alphabet $\Gamma \supseteq \Sigma$
- transition function δ
- special states $q, q_{accept}, q_{reject}$

state control



tape



Configuration

- tuple of state and symbol at tape head

Initial configuration

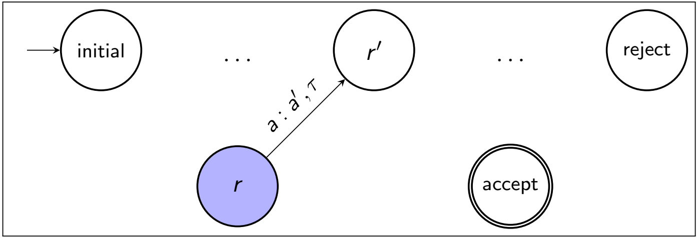
- state control in state q
- tape head at left-most symbol of input string

Accepting configuration

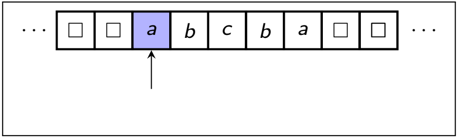
- state control in state q_{accept}

Turing Machine

state control



tape

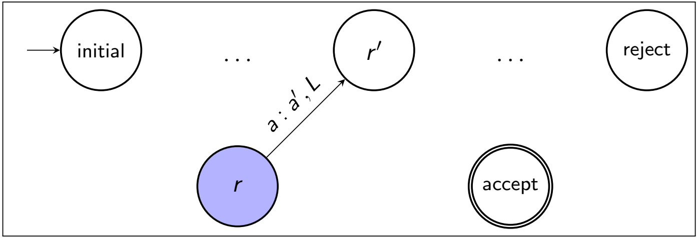


Transitions $r\ a \ \rightarrow\ r'\ a'\ \tau$

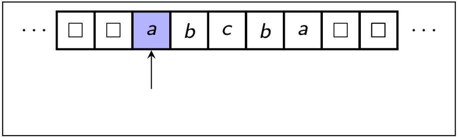
- r current state
- a current symbol at tape head
- r' new state
- a' new symbol that replaces a
- $\tau \in \{R, L, N\}$ move direction of tape head (right, left, no move)

Turing Machine

state control



tape

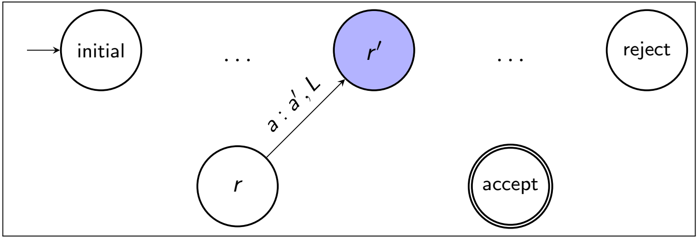


Transitions $r\ a \;\rightarrow\; r'\ a'\ \tau$

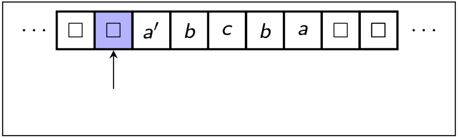
- r current state
- a current symbol at tape head
- r' new state
- a' new symbol that replaces a
- $\tau \in \{R, L, N\}$ move direction of tape head (right, left, no move)

Turing Machine

state control



tape

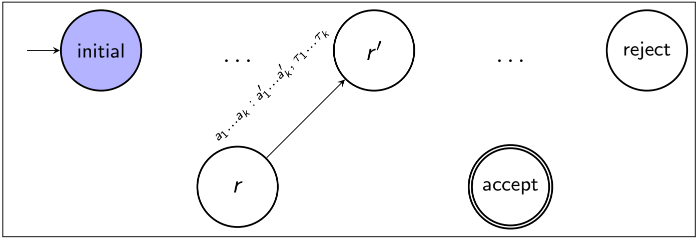


Transitions $r\ a \ \rightarrow\ r'\ a'\ \tau$

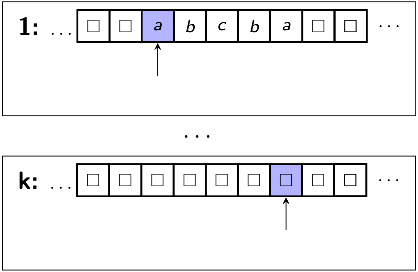
- r current state
- a current symbol at tape head
- r' new state
- a' new symbol that replaces a
- $\tau \in \{R, L, N\}$ move direction of tape head (right, left, no move)

Multi-Tape Turing Machine

state control



tapes 1 to k



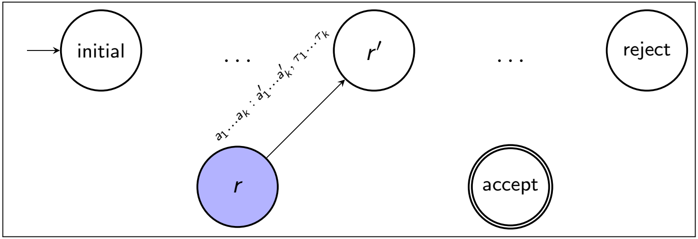
Configuration

- tuple of state and symbol at head of each tape

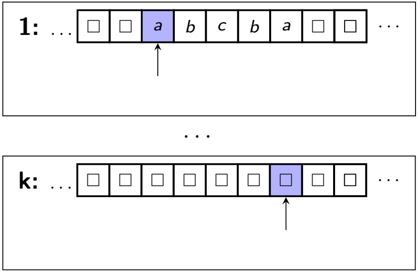
Initial configuration

- state control in state q
- tape head 1 at left-most symbol of
- tapes 2 to k are empty, their heads can be at arbitrary positions

state control



tapes 1 to k



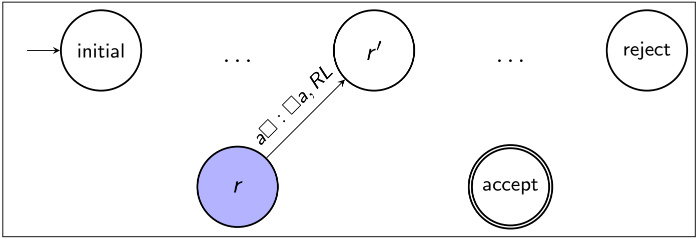
Transitions

$r\ a_1 \dots a_k \ \rightarrow\ r'\ a'_1 \dots a'_k\ \tau_1 \dots \tau_k$

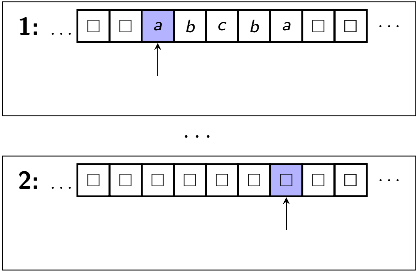
- r current state
- a_i current symbol at head of tape i
- r' new state
- a'_i new symbol that replaces a_i
- $\tau_i \in \{R, L, N\}$ move direction of head of tape i (right, left, no move)

Multi-Tape Turing Machine

state control



tapes 1 and 2



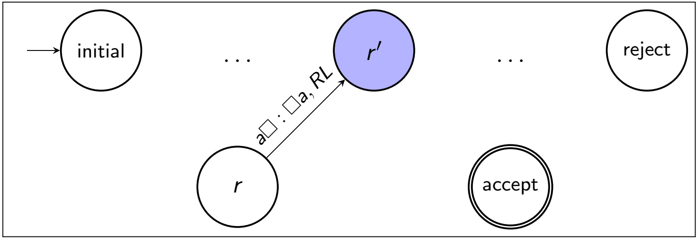
Transitions

$r\ a_1 \dots a_k \ \rightarrow\ r'\ a'_1 \dots a'_k\ \tau_1 \dots \tau_k$

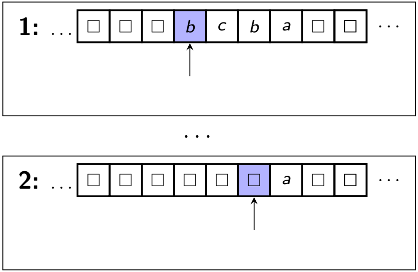
- r current state
- a_i current symbol at head of tape i
- r' new state
- a'_i new symbol that replaces a_i
- $\tau_i \in \{R, L, N\}$ move direction of head of tape i (right, left, no move)

Multi-Tape Turing Machine

state control



tapes 1 and 2



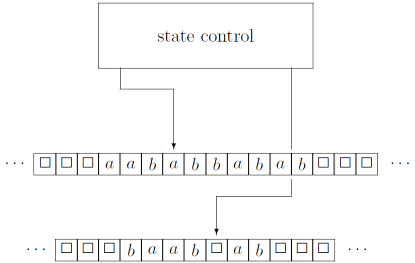
Transitions

$r\ a_1 \dots a_k \ \longrightarrow\ r'\ a'_1 \dots a'_k\ \tau_1 \dots \tau_k$

- r current state
- a_i current symbol at head of tape i
- r' new state
- a'_i new symbol that replaces a_i
- $\tau_i \in \{R, L, N\}$ move direction of head of tape i (right, left, no move)

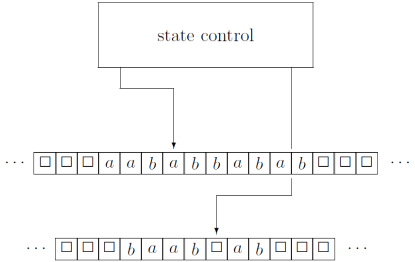
Turing Machines – Informal Summary

- A **single-tape** Turing machine uses only one tape
- A **multi-tape** Turing machine has k tapes, for some fixed $k \geq 1$
- A **tape** is divided into cells, and is **infinite** both to the left and to the right
- Each cell stores a symbol from the tape alphabet Γ
- The tape alphabet contains the blank symbol $\square$
- If a cell contains $\square$, then the cell is **empty**



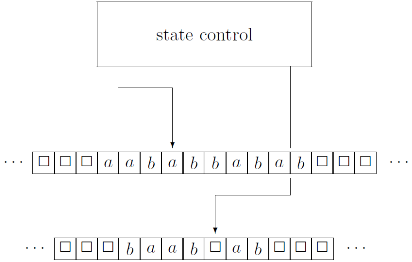
Turing Machines – Informal Summary

- Each tape has a **tape head** which can move along the tape
- By taking a transition a tape head may **move** one position to the **right**, to the **left**, or **remain** at the same position
- The **symbol** at the head of a tape can be **read** and **replaced** by another symbol



Turing Machines – Informal Summary

- A **run** of a Turing machine corresponds to a **sequence of transitions**
- A run starts in the initial configuration, it may terminate the **accept** or **reject** configuration, or it may **never terminate**
- Taking a transition may **update** the current **state**, the **content** of the tapes at the current position of the heads, and the **position** of the heads itself
- Certain parts of a configuration may be also left unchanged by taking a transition



Turing Machines – Definition

Definition

A **deterministic Turing machine** is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q, q_{accept}, q_{reject})$$

where

- Q is a finite set, whose elements are called **states**
- Σ is a finite set, called the **input alphabet**
  the blank symbol $\Box$ is not contained in Σ
- Γ is a finite set, called the **tape alphabet**
  it contains the blank symbol $\Box$, and $\Sigma \subseteq \Gamma$
- q, q_{accept}, and q_{reject} are states called the **start**, **accept**, and **reject** state respectively,
- δ is called the **transition function**, which is a function

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, N\}^k$$

Turing Machines – Transition Function

$$\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, N\}^k$$

Given a current configuration, δ allows to determine the **unique successor configuration**

We use the following **instruction notation** for transitions of Turing machines:

$$r a_1 a_2 \dots a_k \;\rightarrow\; r' a'_1 a'_2 \dots a'_k \tau_1 \tau_2 \dots \tau_k$$

where

- r is the current state, and r' is the changed to state
- $a_1, \dots, a_k$ are the current cell elements the k tapes are at
- $a'_1, \dots, a'_k$ are the new cell elements of the k tapes
- $\tau_1, \dots, \tau_k$ are the moves that each tape head will take (L, R, N)

Turing Machines – Configurations, Runs, Acceptance

Initial configuration:

- The input is a string over the alphabet Σ
- It is stored on the first tape, and the head of this tape is on the leftmost symbol of the input string
- All other $k-1$ tapes are empty

Runs and termination:

- Starting in the initial configuration, the Turing machine takes a sequence of transitions corresponding to the input string
- A run **terminates** at the moment when the Turing machine enters the **accept** state or the **reject** state.

Acceptance:

- The Turing machine accepts the input string if the run over the input terminates in the **accept** state

The language $L(M)$ is the set of all strings over Σ that are accepted by the Turning machine M.

Turing Machine Example: Palindromes

We will construct a Turing machine that accepts the language

$$L = \{w \in \{a, b\}^* : w \text{ is a } \textbf{palindrome}\}$$

A palindrome is a string that reads the **same backward as forward**, e.g.

$$aba, ababa, bbaabb$$

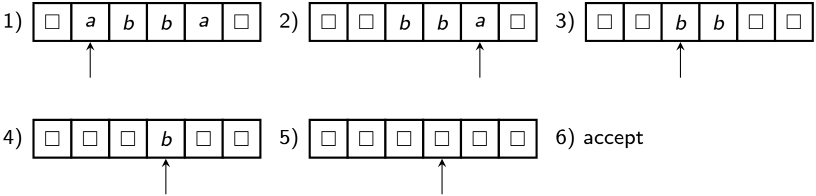
Different Turing machines can solve the problem. We will consider two:

- A **one tape** Turing machine (less efficient)
- A **two tape** Turing machine (more efficient)

One Tape Solution – Idea

- The tape head **reads** the **leftmost symbol** of the input string, **remembers** it by means of a state, and **deletes** it
- Then the tape head moves to the **rightmost symbol** and tests whether it is **equal to** the (already deleted) **leftmost symbol**
  - If **equal**, then the **rightmost symbol is deleted**, the tape head moves to the new leftmost symbol, and the whole **process is repeated**
  - If **not equal**, then the machine *rejects* the input
- The machine **accepts** the input when the **tape becomes empty**

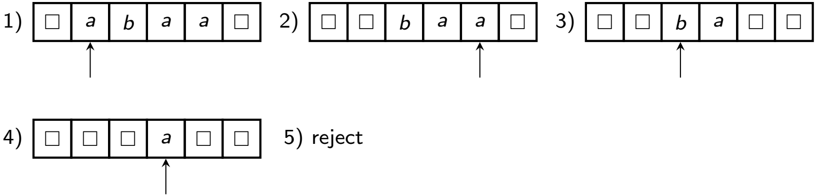
Example 1:



One Tape Solution – Idea

- The tape head **reads** the **leftmost symbol** of the input string, **remembers** it by means of a state, and **deletes** it
- Then the tape head moves to the **rightmost symbol** and tests whether it is **equal to** the (already deleted) **leftmost symbol**
  - If **equal**, then the **rightmost symbol is deleted**, the tape head moves to the new leftmost symbol, and the whole **process is repeated**
  - If **not equal**, then the machine *rejects* the input
- The machine **accepts** the input when the **tape becomes empty**

Example 2:



One Tape Solution – Details

- We use the input alphabet $\Sigma = \{a, b\}$ and the tape alphabet $\Gamma = \{a, b, \Box\}$
- The set Q consists of the following states:

q_0 :	start state; tape head is on the leftmost symbol
q_a :	leftmost symbol was a; tape head is moving to the right
q_b :	leftmost symbol was b; tape head is moving to the right
q'_a :	reached rightmost symbol; test whether it is equal to a, and delete it
q'_b :	reached rightmost symbol; test whether it is equal to b, and delete it
q_L :	test was positive; tape head is moving to the left
q_{accept} :	accept state
q_{reject} :	reject state

One Tape Solution – Details

The transition function δ is defined by the following instructions:

$$q_0 a \to q_a \Box R \qquad q_a a \to q_a a R \qquad q_b a \to q_b a R$$
$$q_0 b \to q_b \Box R \qquad q_a b \to q_a b R \qquad q_b b \to q_b b R$$
$$q_0 \Box \to q_{accept} \qquad q_a \Box \to q_a' \Box L \qquad q_b \Box \to q_b' \Box L$$

$$q_a' a \to q_L \Box L \qquad q_b' a \to q_{reject} \qquad q_L a \to q_L a L$$
$$q_a' b \to q_{reject} \qquad q_b' b \to q_L \Box L \qquad q_L b \to q_L b L$$
$$q_a' \Box \to q_{accept} \qquad q_b' \Box \to q_{accept} \qquad q_L \Box \to q_0 \Box R$$