CM0081 Automata and Formal Languages
§ 8.2 Turing Machines

Andrés Sicard-Ramírez

Universidad EAFIT

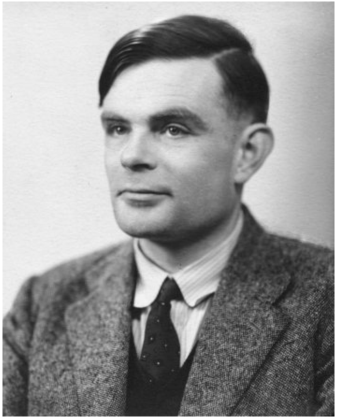
Semester 2024-1

Preliminaries

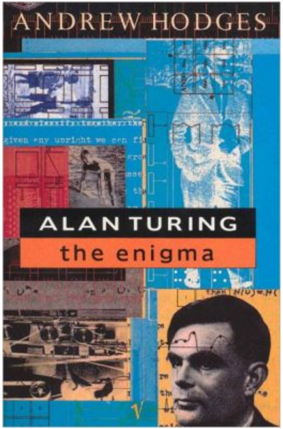
Conventions

- The number and page numbers assigned to chapters, examples, exercises, figures, quotes, sections and theorems on these slides correspond to the numbers assigned in the textbook [Hopcroft, Motwani and Ullman 2007].
- The natural numbers include the zero, that is, $\mathbb{N} = \{0, 1, 2, \dots\}$.
- The power set of a set A, that is, the set of its subsets, is denoted by $\mathcal{P}\,A$.

Introduction

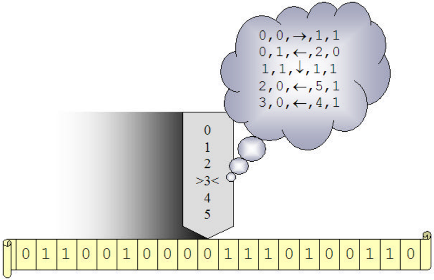


Alan Mathison Turing
(1912 – 1954)



Introduction

- Unbounded tape divided into discrete squares which contain symbols from a finite alphabet.
- Read/Write head.
- Finite set of instructions (transition function).
- Move of a Turing machine:
  From the current state and the tape symbol under the head: change state, rewrite the symbol and move the head one square.



Turing Machines

Definition

A **Turing machine** is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where

Q: A finite set of states

Σ: An alphabet of input symbols

Γ: An alphabet of tape symbols $(\Sigma \subseteq \Gamma)$

$\delta : Q \times \Gamma \to Q \times \Gamma \times D$: A transition (partial) function ($D = \{L, R\}$ set of movements)

$q_0 \in Q$: A start state

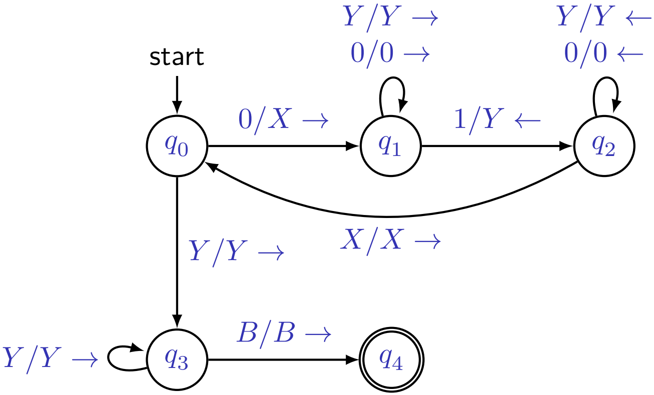
B: The blank symbol $(B \in \Gamma, B \notin \Sigma)$

$F \subseteq Q$: A set of final or accepting states

Transition Diagrams for Turing Machines

Example

Let $\Sigma = \{0, 1\}$ and $\Gamma = \{0, 1, X, Y, B\}$.



Transition Tables for Turing Machines

Example

The machine of the previous example is given by

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

where δ is given by

state	0	1	X	Y	B
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)
q_4	$-$	$-$	$-$	$-$	$-$

Quintuples for Turing Machines

Example

The machine of the previous example is given by

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

where δ is given by

$$\begin{array}{cccc}
q_0, 0, X, R, q_1 & q_1, 0, 0, R, q_1 & q_2, 0, 0, L, q_2 & q_3, Y, Y, R, q_3 \\
q_0, Y, Y, R, q_3 & q_1, 1, Y, L, q_2 & q_2, X, X, R, q_0 & q_3, B, B, R, q_4 \\
 & q_1, Y, Y, R, q_1 & q_2, Y, Y, L, q_2 &
\end{array}$$

Instantaneous Descriptions for Turing Machines

Definition

An **instantaneous description** of a Turing machine is a string

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

where

(i) q is the state of the Turing machine,

(ii) the head is scanning the i-th symbol from the left and

(iii) $X_1 X_2 \cdots X_n$ is the portion of the tape between the leftmost and rightmost non-blank.

Instantaneous Descriptions for Turing Machines

Notation

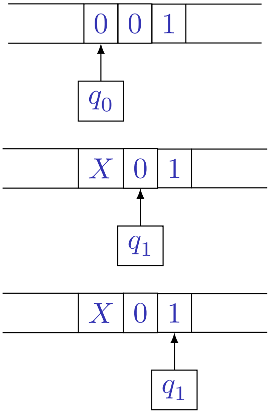
Move of the Turing machine M from an instantaneous description to another is denoted by $\underset{M}{\vdash}$.

Zero o more moves of the Turing machine M are denoted by $\overset{*}{\underset{M}{\vdash}}$.

Instantaneous Descriptions for Turing Machines

Example

For the machine of the previous example we have



$$q_0 001 \underset{M}{\vdash} X q_1 01$$

$$q_0 001 \underset{M}{\vdash} X q_1 01 \underset{M}{\vdash} X0 q_1 1$$

$$q_0 001 \overset{*}{\underset{M}{\vdash}} X0 q_1 1$$

Recursively Enumerable Languages

Definition

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a Turing machine. The **language accepted** by M is defined by

$$\mathrm{L}(M) := \left\{ w \in \Sigma^* \,\middle|\, q_0 w \overset{*}{\underset{M}{\vdash}} \alpha p \beta \right\},$$

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

Recursively Enumerable Languages

Definition

A language L is **recursively enumerable** iff exists a Turing machine M such that $L = \mathrm{L}(M)$.

Recursively Enumerable Languages

Definition

A language L is **recursively enumerable** iff exists a Turing machine M such that $L = \mathrm{L}(M)$.

Example

Let M be the machine described by the previous diagram. Then

$$\mathrm{L}(M) = \{\, 0^n 1^n \mid n \geq 1 \,\}.$$

See the simulation in the course website.

Recursive Languages

Convention

We assume that a Turing machine halts if it accepts.

Recursive Languages

Convention

We assume that a Turing machine halts if it accepts.

What about if a Turing machine does not accept?

Recursive Languages

Convention

We assume that a Turing machine halts if it accepts.

What about if a Turing machine does not accept?

Recall

Recall that a language L is recursively enumerable iff exists a Turing machine M such that $L = \mathrm{L}(M)$.

Recursive Languages

Convention

We assume that a Turing machine halts if it accepts.

What about if a Turing machine does not accept?

Recall

Recall that a language L is recursively enumerable iff exists a Turing machine M such that $L = \mathrm{L}(M)$.

Definition

A language L is **recursive** iff exists a Turing machine M such that

(i) $L = \mathrm{L}(M)$ and

(ii) M always halt (even if it does not accept).

Turing Machine Computable Functions

Definition

A **number-theoretic function** is a function whose signature is

$$\mathbb{N}^k \to \mathbb{N}, \text{ with } k \in \mathbb{N}.$$

Turing Machine Computable Functions

Example

Number-theoretic functions.

$\mathrm{z}(n) = 0$	(**zero** function)
$\mathrm{s}(n) = n + 1$	(**successor** function)
$\mathrm{U}_k^l(n_1, \dots, n_l) = n_k$	(**projection** functions)
$\mathrm{id}(n) = n$	(**identity** function)
$\mathrm{C}_k^l(n_1, \dots, n_l) = k$	(**constant** functions)
$m + n$	(**addition** function)
$m \cdot n$	(**multiplication** function)
m^n	(**exponentiation** function)
$n!$	(**factorial** function)

Turing Machine Computable Functions

Example

Number-theoretic functions.

$$\mathrm{pred}(n) = \begin{cases} 0, & \text{if } n = 0; \\ n - 1, & \text{otherwise}; \end{cases}$$

(**predecessor** function)

$$m \dot{-} n = \begin{cases} m - n, & \text{if } m \geq n; \\ 0, & \text{otherwise}; \end{cases}$$

(**truncated subtraction** function)

$$|m - n| = \begin{cases} m \dot{-} n, & \text{if } m \geq n; \\ n \dot{-} m, & \text{otherwise}; \end{cases}$$

(**absolute difference** function)

Turing Machine Computable Functions

Example

Number-theoretic functions.

$$\mathrm{sg}(n) = \begin{cases} 0, & \text{if } n = 0; \\ 1, & \text{otherwise}; \end{cases}$$

(**signum** function)

$$\overline{\mathrm{sg}}(n) = \begin{cases} 1, & \text{if } n = 0; \\ 0, & \text{otherwise}. \end{cases}$$

(**inverse signum** function)

Turing Machine Computable Functions

Codification of k-tuples of natural numbers

$$\vec{n} := 0^n = \underbrace{0 \cdots 0}_{n \text{ times}}, \qquad \text{for } n \in \mathbb{N};$$

$$\overrightarrow{(n_1, n_2, \dots, n_k)} := \overrightarrow{n_1}\, 1\, \overrightarrow{n_2}\, 1 \cdots 1\, \overrightarrow{n_k}, \qquad \text{for } (n_1, n_2, \dots, n_k) \in \mathbb{N}^k.$$

Turing Machine Computable Functions

Definition

A unary function $f : \mathbb{N} \to \mathbb{N}$ is **Turing machine computable** iff exists a machine $M = (Q, \{0,1\}, \Gamma, \delta, q_0, B)$ (there are not accepting states), such that for all $n \in \mathbb{N}$, from the initial instantaneous description $q_0\vec{n}$ the machine halts with $\overrightarrow{f(n)}$ on its tape, surrounded by blanks.

Turing Machine Computable Functions

Definition

A unary function $f : \mathbb{N} \to \mathbb{N}$ is **Turing machine computable** iff exists a machine $M = (Q, \{0, 1\}, \Gamma, \delta, q_0, B)$ (there are not accepting states), such that for all $n \in \mathbb{N}$, from the initial instantaneous description $q_0\vec{n}$ the machine halts with $\overrightarrow{f(n)}$ on its tape, surrounded by blanks.

Observation

The definition can be extended to functions $f : \mathbb{N}^k \to \mathbb{N}$.

Turing Machine Computable Functions

Example

The truncated subtraction function is Turing machine computable.

$$m \dot{-} n = \begin{cases} m - n, & \text{if } m \geq n; \\ 0, & \text{otherwise}. \end{cases}$$

Initial instantaneous description: $q_0 0^m 1 0^n$

Final information on the tape: $0^{m \dot{-} n}$

See the simulation in the course homepage.

Turing Machine Computable Functions

Example

All the number-theoretic functions in the previous examples are Turing machine computable functions.

Equivalence between Function Computation and Language Recognition

Exercise 8.2.4

- Define the graph of a function $f : \mathbb{N} \to \mathbb{N}$ to be the set of all strings of the form $\left[\vec{n}, \overrightarrow{f(n)}\right]$.

Equivalence between Function Computation and Language Recognition

Exercise 8.2.4

- Define the graph of a function $f : \mathbb{N} \to \mathbb{N}$ to be the set of all strings of the form $\left[\vec{n}, \overrightarrow{f(n)}\right]$.
- A Turing machine is said to compute the function $f : \mathbb{N} \to \mathbb{N}$ if, started with $\vec{n}$ on its tape, it halts (in any state) with $\overrightarrow{f(n)}$ on its tape.

(continued on next slide)

Exercise 8.2.4 (continuation)

Answer the following, with informal, but clear constructions.

Equivalence between Function Computation and Language Recognition

Exercise 8.2.4 (continuation)

Answer the following, with informal, but clear constructions.

1. Show how, given a Turing machine that computes f, you can construct a Turing machine that accepts the graph of f as a language.

Equivalence between Function Computation and Language Recognition

Exercise 8.2.4 (continuation)

Answer the following, with informal, but clear constructions.

1. Show how, given a Turing machine that computes f, you can construct a Turing machine that accepts the graph of f as a language.
2. Show how, given a Turing machine that accepts the graph of f, you can construct a Turing machine that computes f.

Equivalence between Function Computation and Language Recognition

Exercise 8.2.4 (continuation)

Answer the following, with informal, but clear constructions.

1. Show how, given a Turing machine that computes f, you can construct a Turing machine that accepts the graph of f as a language.
2. Show how, given a Turing machine that accepts the graph of f, you can construct a Turing machine that computes f.
3. A function is said to partial if it may be undefined for some arguments. If we extend the ideas of this exercise to partial functions, then we do not require that the Turing machine computing f halts if its input n is one of the natural numbers for which $f(n)$ is not defined.

   Do your constructions for parts (1) and (2) work if the function f is partial? If not, explain how you could modify the constructions to make it work.

Restrictions to Turing Machines

Restrictions

Restrictions to Turing Machines

Restrictions

- Turing machines with semi-unbounded tapes

Restrictions to Turing Machines

Restrictions

- Turing machines with semi-unbounded tapes
- Multi-stack machines

Restrictions to Turing Machines

Restrictions

- Turing machines with semi-unbounded tapes
- Multi-stack machines

Theorem

The previous restrictions are equivalents to Turing machines.

Extensions to Turing Machines

Extensions

Extensions to Turing Machines

Extensions

- Multi-tape Turing machines

Extensions to Turing Machines

Extensions

- Multi-tape Turing machines
- Mutil-dimensional tape Turing machines

Extensions to Turing Machines

Extensions

- Multi-tape Turing machines
- Mutil-dimensional tape Turing machines
- Multi-head Turing machines

Extensions to Turing Machines

Extensions

- Multi-tape Turing machines
- Mutil-dimensional tape Turing machines
- Multi-head Turing machines
- Non-deterministic Turing machines

Extensions to Turing Machines

Extensions

- Multi-tape Turing machines
- Mutil-dimensional tape Turing machines
- Multi-head Turing machines
- Non-deterministic Turing machines
- Subroutines

Extensions to Turing Machines

Extensions

- Multi-tape Turing machines
- Mutil-dimensional tape Turing machines
- Multi-head Turing machines
- Non-deterministic Turing machines
- Subroutines

Theorem

The previous extensions are equivalents to Turing machines.

References

Hopcroft, J. E., Motwani, R. and Ullman, J. D. [1979] (2007). Introduction to Automata Theory, Languages, and Computation. 3rd ed. Pearson Education (cit. on p. 2).