Computability and Complexity

Lecture 1

Introduction
Turing machines
Decidable and recognizable languages

given by Jiri Srba

Theory of Computation

The theory of computation studies

- whether (computability theory), and
- how efficiently (complexity theory)

certain problems can be solved on a computer, or rather on a model of a computer.

Several equivalent models of computational devices can be used:

- Register machines.
- Lambda calculus.
- Simple while programming language (pseudo-code).
- Turing machines.
- ...

We focus on Turing machines.

Course Overview

Computability

- Do algorithmic solutions to problems always exist?
- What are the limitations of computational devices?
- Is there any insight to which problems are algorithmically solvable and which are not?
- Are the unsolvable problems somehow related?

Question: Will a given program ever raise a null pointer exception?

Question: Are two different implementations of some library function equivalent?

Course Overview

Complexity

- How do we measure time/memory requirements of an algorithm?
- How much time/memory is needed to solve a certain problem?
- What problems are efficiently solvable?
- Are there solvable problems which do not have efficient algorithms?

Question: Given a nonnegative integer, is it a prime number?

Question: Does a graph contain contain a Hamiltonian cycle?

Formal Languages — Repetition

- Let Σ be a finite, nonempty set called alphabet.
- A string or word w is a finite sequence of symbols from Σ.
- An empty string is denoted by ϵ.
- A concatenation of strings w_1 and w_2 is a string $w_1 w_2$.
- The length of the string w is denoted by $|w|$.
- The set of all strings over Σ is denoted by Σ^*.
- A language L over Σ is any subset of Σ^*, i.e., $L \subseteq \Sigma^*$.

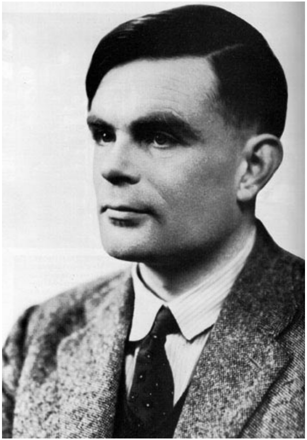
Let L_1 and L_2 be two languages ($L_1, L_2 \subseteq \Sigma^*$).

- $L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ or } w \in L_2\}$
- $L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ and } w \in L_2\}$
- $\overline{L_1} = \{w \in \Sigma^* \mid w \notin L_1\}$
- $L_1.L_2 = \{w \in \Sigma^* \mid w = w_1 w_2 \text{ where } w_1 \in L_1 \text{ and } w_2 \in L_2\}$
- $L_1^* = \{w \in \Sigma^* \mid w = w_1 w_2 \ldots w_k \text{ where } k \geq 0$
  $\text{and each } w_i \in L_1 \text{ for all } 1 \leq i \leq k\}$

Turing Machine

Devised in 1936 by English mathematician Alan Turing.

- computations can be done by writing symbols on sheets of paper
- we have a pen, an eraser, finitely many symbols we can write, and as many sheets of paper as we want
- no thinking is required to execute a computation
- essentially a finite-state automaton with unbounded memory



Turing Machine Formally

Definition

A Turing machine is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$:

- Q is a finite set of states,
- Σ is a finite input alphabet, s.t. $\sqcup \notin \Sigma$
- Γ is a finite tape alphabet, s.t. $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function,
- $q_0 \in Q$ is the start state,
- $q_{accept} \in Q$ is the accept state, and
- $q_{reject} \in Q$ is the reject state, where $q_{accept} \neq q_{reject}$.

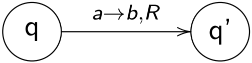
Definition

A Turing machine is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$:

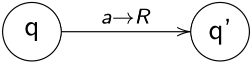
- Q is a finite set of states,
- Σ is a finite input alphabet, s.t. $\sqcup \notin \Sigma$
- Γ is a finite tape alphabet, s.t. $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function,
- $q_0 \in Q$ is the start state,
- $q_{accept} \in Q$ is the accept state, and
- $q_{reject} \in Q$ is the reject state, where $q_{accept} \neq q_{reject}$.

Notation:

$$\delta(q, a) = (q', b, R)$$



$$\delta(q, a) = (q', a, R)$$



Configuration of a Turing Machine (TM)

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ be a TM.
Informally, a configuration consists of

- the current control-state,
- the current tape content, and
- the current head position.

Configuration of a Turing Machine (TM)

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ be a TM.
Informally, a configuration consists of

- the current control-state,
- the current tape content, and
- the current head position.

Definition (Configuration)

A configuration of a TM is a string uqv where

- $u \in \Gamma^*$ is the initial part of the tape,
- $q \in Q$ is the current state,
- $v \in \Gamma^*$ is the final part of the tape and the head points at the first symbol of v.

Remark: if $v = \epsilon$ then the head points at the first blank symbol $\sqcup$ after u, i.e., $uq \equiv uq\sqcup$.

Computation of a Turing Machine

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ be a TM.
Informally, a configuration C_1 yields a configuration C_2 if

- in configuration C_1 the machine M performs one computational step and moves to C_2.

Computation of a Turing Machine

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ be a TM.
Informally, a configuration C_1 yields a configuration C_2 if

- in configuration C_1 the machine M performs one computational step and moves to C_2.

Definition (C_1 yields C_2)

Let $u, v \in \Gamma^*$, $a, b \in \Gamma$, and $q, q' \in Q$. We say that

- $C_1 = uaqbv$ yields $uq'acv = C_2$ if $\delta(q, b) = (q', c, L)$,

Computation of a Turing Machine

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ be a TM.
Informally, a configuration C_1 yields a configuration C_2 if

- in configuration C_1 the machine M performs one computational step and moves to C_2.

Definition (C_1 yields C_2)

Let $u, v \in \Gamma^*$, $a, b \in \Gamma$, and $q, q' \in Q$. We say that

- $C_1 = uaqbv$ yields $uq'acv = C_2$ if $\delta(q, b) = (q', c, L)$, and
- $C_1 = uqbv$ yields $ucq'v = C_2$ if $\delta(q, b) = (q', c, R)$,

Computation of a Turing Machine

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ be a TM.
Informally, a configuration C_1 yields a configuration C_2 if

- in configuration C_1 the machine M performs one computational step and moves to C_2.

Definition (C_1 yields C_2)

Let $u, v \in \Gamma^*$, $a, b \in \Gamma$, and $q, q' \in Q$. We say that

- $C_1 = uaqbv$ yields $uq'acv = C_2$ if $\delta(q, b) = (q', c, L)$, and
- $C_1 = uqbv$ yields $ucq'v = C_2$ if $\delta(q, b) = (q', c, R)$, and
- $C_1 = qbv$ yields $q'cv = C_2$ if $\delta(q, b) = (q', c, L)$.

Acceptance of a String by a TM

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ be a TM.
Informally, a TM M accepts a string $w \in \Sigma^*$ if

- from the initial configuration $q_0 w$ there is a computation which ends in a configuration of the form $u q_{accept} v$.

Acceptance of a String by a TM

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ be a TM.
Informally, a TM M accepts a string $w \in \Sigma^*$ if

- from the initial configuration $q_0 w$ there is a computation which ends in a configuration of the form $uq_{accept}v$.

Definition (M accepts a string w)

A TM M accepts an input string w if there is a sequence of configurations $C_1, C_2, \ldots, C_k$ such that

- $C_1 = q_0 w$ is the initial (start) configuration,
- C_i yields C_{i+1} for all i, $1 \leq i < k$, and
- C_k is an accept configuration (contains q_{accept}).

Similarly, M rejects w if there is such a sequence ending in C_k which is a reject configuration (contains q_{reject}).

Three Possible Outcomes of a TM Computation, Decider

Assume that a TM M computes from the initial configuration $C_1 = q_0 w$ (note that this computation is deterministic).

There are three possible outcomes of running M on w:

1. $C_1, C_2, \ldots, C_k$ ends in accept configuration C_k, or
2. $C_1, C_2, \ldots, C_k$ ends in reject configuration C_k, or

Three Possible Outcomes of a TM Computation, Decider

Assume that a TM M computes from the initial configuration $C_1 = q_0 w$ (note that this computation is deterministic).

There are three possible outcomes of running M on w:

1. $C_1, C_2, \ldots, C_k$ ends in accept configuration C_k, or
2. $C_1, C_2, \ldots, C_k$ ends in reject configuration C_k, or
3. $C_1, C_2, \ldots$ does not terminate (loops).

Three Possible Outcomes of a TM Computation, Decider

Assume that a TM M computes from the initial configuration $C_1 = q_0 w$ (note that this computation is deterministic).

There are three possible outcomes of running M on w:

1. $C_1, C_2, \ldots, C_k$ ends in accept configuration C_k, or
2. $C_1, C_2, \ldots, C_k$ ends in reject configuration C_k, or
3. $C_1, C_2, \ldots$ does not terminate (loops).

Definition (Decider)

A TM M which for any input string w always halts (either in accept or reject configuration) is called a decider.

Definition (Language of M)

The language recognized by a TM M, or simply the language of M is

$$L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}.$$

Recognizable and Decidable Languages

Definition (Language of M)

The language recognized by a TM M, or simply the language of M is

$$L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}.$$

Definition (Recognizable Language)

A language $L \subseteq \Sigma^*$ is recognizable if there exists a TM M such that M recognizes L, i.e., $L = L(M)$.

Recognizable and Decidable Languages

Definition (Language of M)

The language recognized by a TM M, or simply the language of M is

$$L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}.$$

Definition (Recognizable Language)

A language $L \subseteq \Sigma^*$ is recognizable if there exists a TM M such that M recognizes L, i.e., $L = L(M)$.

Definition (Decidable Language)

A language $L \subseteq \Sigma^*$ is decidable if there exists a TM M such that M is a decider and M recognizes L, i.e., $L = L(M)$.

Example

Consider the language $L \stackrel{\text{def}}{=} \{a^n b^n c^n \mid n \geq 0\}$.

Facts:

- L is not regular,
- L is not context-free, but
- L is recognizable and even decidable language.

Exam Questions

- Definition of a Turing machine, configuration, computation, acceptance of a string by a TM.
- Definition of a decider.
- Definition of recognizable and decidable languages.