CS:4330 Theory of Computation

Spring 2018

Computability Theory

Summary

Haniel Barbosa

The University of Iowa

Readings for this lecture

Chapters 3-5 and Section 6.2 of [Sipser 1996], 3rd edition.

A hierachy of languages

- Regular: $a^n b^m$
- Deterministic Context-free: $a^n b^n$
- Context-free: $a^n b^n \cup a^n b^{2n}$
- Turing decidable: $a^n b^n c^n$
- Turing recognizable: A_{TM}

Why TMs?

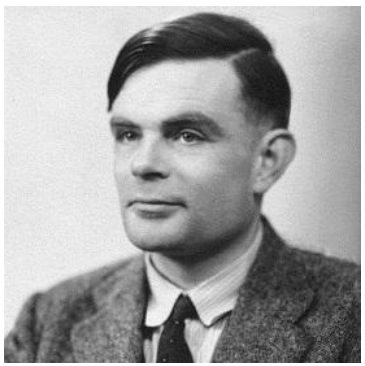
- In 1900: Hilbert posed 23 “challenge problems” in Mathematics

The 10^{th} problem:

> *Devise a process according to which it can be decided by a finite number of operations if a given polynomial has an integral root.*

It became necessary to have a formal definition of “algorithms” to define their expressivity.

Church-Turing Thesis



- ▷ In 1936 Church and Turing independently defined “algorithm”:
  - ▶ λ-calculus
  - ▶ Turing machines
- ▷ Intuitive notion of algorithms = Turing machine algorithms
- ▷ “Any process which could be naturally called *an effective procedure* can be realized by a Turing machine”
- ▷ We now know: Hilbert’s 10^{th} problem is **undecidable**!

Algorithm as Turing Machine

Definition (Algorithm)

An *algorithm* is a decider TM in the standard representation.

- ▷ The input to a TM is always a string.
- ▷ If we want an object other than a string as input, we must first represent that object as a string.
- ▷ Strings can easily represent polynomials, graphs, grammars, automata, and any combination of these objects.

How to determine decidability / Turing-recognizability?

- ▷ Decidable / Turing-recognizable:
  - Present a TM that decides (recognizes) the language
  - If A is *mapping reducible* to B and B is decidable (Turing-recognizable), then A is decidable (Turing-recognizable)
- ▷ Undecidable / Not Turing-recognizable:
  - Prove that no TM decider (no TM) can recognize the language
  - If A is *mapping reducible* to B and B is undecidable (not Turing-recognizable), then B is undecidable (not Turing-recognizable)

Definition

Language A is *mapping reducible* to language B, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \to \Sigma^*$, called a *reduction* from A to B, such that for every $w \in \Sigma^*$,

$$w \in A \Leftrightarrow f(w) \in B$$

Decidability of acceptance problems for DFAs

Theorem

A_{DFA} *is a decidable language.*

Proof idea: construct a TM M that decides A_{DFA}

$M =$"On input string $\langle B, w \rangle$, where B is a DFA and w is a string:

1. Simulate B on w
2. If the simulation ends in an accept state then *accept*; otherwise *reject*."

Acceptance problem for NFAs

Theorem

$A_{\text{NFA}} = \{\langle B, w\rangle \mid B \text{ is an NFA that accepts the string } w\}$ *is a decidable language*

Proof idea: construct a TM N that decides A_{NFA}.

By using the TM M that decides A_{DFA}, N first converts its input NFA to a DFA by the usual technique.

$N =$"On input string $\langle B, w\rangle$, where B is an NFA and w is a string:

1. Convert B to a DFA C
2. Run M on $\langle C, w\rangle$
3. If M accepts, *accept*; otherwise *reject*."

Note

Running M in stage 2 means incorporating M into the design of N as a subprocedure.

Halting problem of TMs

$$A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$$

While A_{DFA} and A_{NFA} are decidable, A_{TM} **is not**.

Theorem

A_{TM} is undecidable.

A_{TM} is however Turing-recognizable. The following TM U recognizes A_{TM}:

$U =$"On input string $\langle M, w\rangle$, where M is a TM and w is a string:

1. Simulate M on input w
2. If M ever enters its accept state, *accept*; if M ever enters its reject state, *reject*."

U is an *universal Turing machine*, which can simulate any other Turing machine from its description.

Halting problem of TMs

$$A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$$

While A_{DFA} and A_{NFA} are decidable, A_{TM} **is not**.

Theorem

A_{TM} is undecidable.

A_{TM} is however Turing-recognizable. The following TM U recognizes A_{TM}:

$U =$"On input string $\langle M, w\rangle$, where M is a TM and w is a string:

1. Simulate M on input w
2. If M ever enters its accept state, *accept*; if M ever enters its reject state, *reject*."

U is an *universal Turing machine*, which can simulate any other Turing machine from its description.

Emptiness problem for TM

Theorem

The language $E_{\mathsf{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } \mathscr{L}(M) = \varnothing\}$ *is undecidable.*

Proof idea: reduction from A_{TM} to E_{TM}

- ▷ Assume that E_{TM} is decidable and let R be its TM decider.
- ▷ Show that a TM S can be constructed using R that decides A_{TM}
- ▷ *Bad idea*: Run R on $\langle M \rangle$. If it accepts then $\mathscr{L}(M) = \varnothing$, so M does not accept w; otherwise, $\mathscr{L}(M) \neq \varnothing$, which does not entail that M accepts w
- ▷ *Good idea*: run R on a modification $\langle M_1 \rangle$ of $\langle M \rangle$ that guarantees that M_1 rejects all strings except w. That is

$$\mathscr{L}(M_1) = \begin{cases} \{w\} & \text{if } w \in \mathscr{L}(M) \\ \varnothing & \text{otherwise} \end{cases}$$

R can test then if $\mathscr{L}(M_1) = \varnothing$ to determine whether $w \in \mathscr{L}(M)$.

Proving E_{TM} is undecidable

The modified machine M_1

$M_1 =$"On input x:

1. If $x \neq w$, *reject*
2. If $x = w$, run M on input w and if M accepts, *accept*."

▷ Note that M_1 has w "hardcoded" as part of its description.

The machine S

$S =$"On input $\langle M, w \rangle$, in which M is a TM and w a string:

1. Construct M_1 from $\langle M, w \rangle$
2. Run R on input $\langle M_1 \rangle$
3. If R accepts, *reject*; if R rejects, *accept*."

▷ If E_{TM} was decidable then A_{TM} would be decidable. Since A_{TM} is undecidable, so is E_{TM}.

Logical Theories

Definition

If $\mathscr{M}$ is a structure, the *theory of* $\mathscr{M}$, written $\mathrm{Th}(\mathscr{M})$, is the collection of true sentences in the language of that structure.

The decision problem associated with a theory $Th(\mathscr{M})$ is to determine whether a given sentence φ belongs to the theory, i.e. whether $\mathscr{M}$ is a model of φ.

A decidable theory

Theorem

$Th(\mathbb{N}, +)$ is decidable.

Proof idea: addition with finite automata

- ▷ Let $\varphi = Q_1 x_1 Q_2 x_2 \ldots Q_l x_l . \psi$, in which $Q \in \{\exists, \forall\}$ and ψ is quantifier-free.
- ▷ Let $\varphi_i = Q_{i+1} x_{i+1} Q_{i+2} x_{i+2} \ldots Q_l x_l . \psi$, for $i = 0..l$. Thus $\varphi_0 = \varphi$ and $\varphi_l = \psi$.
- ▷ The formula φ_i has i free variables. For $a_1, \ldots, a_i \in \mathbb{N}$, we write $\varphi_i[a_1, \ldots, a_i]$ to be the sentence obtained by substituting the constants $a_1, \ldots, a_i$ for the variables $x_1, \ldots, x_i$ in φ_i.
- ▷ The algorithm builds finite automata A_i which recognize the collection of strings representing i-tuples of numbers that make φ_i true.
- ▷ It first builds A_l directly. Then for each $i = l..1$, it uses A_i to build A_{i-1}.
- ▷ Once the algorithm has A_0, it tests whether A_0 accepts ϵ, in which case it shows that φ is true.