Undergraduate Complexity Theory
Lecture 2: Turing Machines

Marcythm

July 8, 2022

1 Lecture Notes

Today we're mainly going to talk about hwo to define an algorithm using Turing Machines.

$$\text{input} \longrightarrow \boxed{\text{algorithm}} \longrightarrow \text{output}$$

Definition 1.1. A language L is a subset of strings $L \subseteq \Sigma^*$.

Fact 1.2. *In general, decision problem $f : \Sigma^* \to \{yes, no\}$ can convert language $L = \{x \in \Sigma^* : f(x) = yes\}$, and conversely, language L convert to decision problem $f(x) = \begin{cases} yes, & x \in L \\ no, & x \notin L \end{cases}$.*

e.g. $\mathsf{IsPrime}$: $\{0,1\}^* \to \{\text{yes}, \text{no}\}$ is equal to language $\mathsf{PRIMES} = \{\langle x \rangle : x \in \mathbb{N}, x \text{ is prime}\}$.

Daily programming languages are fun to program, but hard to formalize.

Ways to formalize "algorithm":

1. Turing Machine ('36)
2. Lambda Calculus ('36)
3. Post Machine ('36)
4. Wang Machine ('50s)
5. P" ('64)

think they as programming language but "machines". Easy to formalize, but annoying to program.

Church-Turing Thesis: Any real-world algorithm can be simulated by (compiled to) Turing Machines.

Definition 1.3. *"computable"* means computable by TMs.

Fact 1.4. *An algo running in time T in C-like pesudocode can be compiled to a TM running in time $\approx T^4$.*

Extended Church-Turing Thesis: Any real-world algorithm can …with at most polynomial slowdown.

Our official model: (1-tape, two-way infinite) Turing Machine.
input alphabet Σ, tape alphabet $\Gamma = \Sigma \cup \{b\}$ where b is the blank symbol. …many definitions about TM.

Definition 1.5. The *computation trace* of M on input X is a sequence of configurations $C_0, C_1, \ldots, C_n$ where C_0 is the initial configuration, C_i yields C_{i+1}, and C_n is an halting configuration.

Definition 1.6. TM M is a *decider* if M on X halt for all $X \in \Sigma^*$.

Definition 1.7. A TM M decides language L iff M is a decider and M accept X iff $X \in L$, otherwise reject.

2 Reading

2.1 Sipser 3.1 (Turing Machines)

1. configuration
2. yield
3. starting / accepting / rejecting / halting configuration
4. $L(M)$: the language of / recognized by M
5. decider
6. Turing recognizable (recursively enumerable language)
7. Turing decidable (recursive language)

2.2 Sipser 3.3 (The Definition of Algorithm)

1. hilbert's 10th problem, history background
2. Church-Turing Thesis
3. 3-level Description of TM: formal description, impl description, high-level description