Lecture #10: First Hard Problems

Lecture Presentation

Comparing "Computability of Languages" and "Computability of Sets"

In CPSC 351 (or 331) you were, very probably, introduced to ***Turing machines with languages:*** These replaced the "halting" state with a pair of "accepting" and "rejecting" states, so that a Turing machine was presented as a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

- M ***accepts*** a string $\omega \in \Sigma^\star$ if M enters an accepting configuration (that is, a configuration where the machine is in state q_{accept}) after making a finite number of moves, when executed on input ω. M ***rejects*** ω if M enters a rejecting configuration (that is, a configuration where the machine is in state q_{reject}) after making a finite number of moves, when executed on input ω. M ***loops on*** ω otherwise.

  As defined here (and, possibly, in CPSC 351 or 331 too), $\delta(q, \sigma)$ is defined for every state $Q \setminus \{q_{\text{accept}}, q_{\text{reject}}\}$ and for every tape symbol $\sigma \in \Gamma$ — so M's computation on the input ω does not halt if M "loops on" this string (as defined here).

- M ***recognizes*** a language $L \subseteq \Sigma^\star$ — so that L is "the language of M" if M accepts every string $\omega \in L$, and M either rejects or loops on every string $\omega \in \Sigma^\star$ such that $\omega \notin L$. M ***decides*** L if M accepts every string $\omega \in L$ and M rejects every string $\omega \in \Sigma^\star$ such that $\omega \in L$. That is, M *decides* L if M *recognizes* L and, furthermore, the execution of M on every string in $\Sigma^\star$ halts.

- A language $L \subseteq \Sigma^\star$ is ***Turing-recognizable*** (which is often shortened to "*recognizable*") if there exists a Turing machine M such that M recognizes L. L is ***Turing-decidable*** (often shortened to "*decidable*") if there exists a Turing machine M such that M decides L.

The goal of this part of the presentation is to better understand how *recognizable* languages, *decidable* languages, *computable* subsets of $\mathbb{N}$ and *computably enumerable* subsets of $\mathbb{N}$ are related.

Relating Languages to Subsets of $\mathbb{N}$

Relating Turing-Decidable Languages to Computable Sets

Relating Turing-Recognizable Languages to Computably Enumerable Sets

A "Parameter Theorem" for Turing Machines

Recall that, for every positive integer n, $\Phi^{(n)} : \mathbb{N}^{n+1} \to \mathbb{N}$ was defined to be the function such that, for all $x_1, x_2, \ldots, x_n, y \in \mathbb{N}$,

$$\Phi^{(n)}(x_1, x_2, \ldots, x_n, y) = f_y(x_1, x_2, \ldots, x_n)$$

where $f_y : \mathbb{N}^n \to \mathbb{N}$ is the (partial) function computed by the $\mathcal{S}$-program $\mathcal{P}$ such that $y = \#(\mathcal{P})$. The preparatory reading also included the following technical result — which relates the various functions $\Phi^{(n)}$ for different values of n.

Parameter Theorem: *For all positive integers n and m there exists a primitive recursive function*

$$S_m^n : \mathbb{N}^{n+1} \to \mathbb{N}$$

such that, for all $x_1, x_2, \ldots, x_m, u_1, u_2, \ldots, u_n, y \in \mathbb{N}$,

$$\Phi^{(m+n)}(x_1, x_2, \ldots, x_m, u_1, u_2, \ldots, u_n, y) = \Phi^{(m)}(x_1, x_2, \ldots, x_m, S_m^n(u_1, u_2, \ldots, u_n, y)).$$

The goal of this document is to describe a similar "Parameter Theorem for Turing Machines" — which one might see, instead of the above "Parameter Theorem", in a version of this course where Turing machines were being used as the basic model of computation instead of $\mathcal{S}$-programs.

Getting Started: Enumerating Turing Machines

A Weak Parameter Theorem for Turing Machines

A Stronger Parameter Theorem for Turing Machines