CIS 511: Theory of Computation **Jan 31, 2017**

Lecture 6: Decidability and Undecidability

Professor Sampath Kannan *Zach Schutzman*

NB: *These notes are from CIS511 at Penn. The course followed Michael Sipser's Introduction to the Theory of Computation (3ed) text.*

Decidable Languages

Definition 6.1 *Recall, a language is* ***decidable*** *if it can be recognized by a Turing Machine which always halts.*

Examples of Decidable languages*:*

- *The language of bipartite graphs:* $\{\langle G\rangle \mid g \text{ is bipartite}\}$
  A TM to decide if a graph is bipartite: use a second tape as a work tape, which will store two arrays. Begin by putting vertex v_1 *on the 'left' side. Iteratively choose a vertex whose label we know, and add its neighbors to the other side. If we ever put a vertex on both sides, reject. If we successfully classify every vertex, accept.*

- *The language of* $3SAT$*:* $\{\langle \Phi\rangle \mid \Phi \text{ is a boolean formula in 3-CNF with a satisfying assignment}\}$
  This is our classic NP-Complete problem. CNF (conjunctive normal form) means that the formula is written as the AND *of several clauses, each containing the* OR *of several literals (variables and/or their complements). 3-CNF is CNF where each clause has at most 3 literals. A TM to decide this language is to enumerate the* 2^n *possibilities for all assignments. If we find one that is true, accept. If we exhaust all possibilities, reject.*

- *Every regular language (and context-free language) is decidable.*

Claim 6.2 *A Turing machine can be designed to simulate any other Turing machine. Essentially, we can describe a Turing machine as a string encoding its 7-tuple.*

Now, we can talk about Turing machines that take another TM as input and performs some computation with respect to that machine. A Turing machine T *can take as input a a machine and a string* $\langle M, w\rangle$*, we can think of* T *like an interpreter and executer and* M *as a program with input* w*.* T *outputs* $M(w)$ *(if such a result can be obtained in finite time).*

At a high level, T *has a work tape that keeps the current configuration of* M*. It starts with the start state of* M *and the input string* w*.* T *consults the transition function of* M *and updates the work tape accordingly. At the end of execution, the work tape contains the output of* M *(with some extra configuration information).*

Definition 6.3 *We call this* T *a* ***Universal Turing Machine****.*

Undecidable Languages

Is there even a language that is undecidable?

Claim 6.4 *Yes!*

Proof:

We use a counting argument to show the existence of an undecidable language (actually, there are uncountably many non-decidable languages). If Σ is a finite alphabet, Σ^* is countably infinite, as it can be enumerated in lexicographic order, for example.

The set of all Turing machines is also countable, as we can encode it as a string over some alphabet, say $\{0, 1, \#\}$. Conversely, we can show $\mathcal{P}(\Sigma^*)$, the set of all subsets of Σ^*, i.e. the set of all languages, is uncountable.

By Cantor's diagonal argument, we can show the set of binary functions on $\mathbb{N}$ is uncountable. This set is in bijection with $\mathcal{P}(\mathbb{N})$. Since there is a bijection $\Sigma^* \longleftrightarrow \mathbb{N}$, and a bijection $\mathcal{P}(\mathbb{N}) \longleftrightarrow \mathcal{P}(\Sigma^*)$ and no bijection $\mathcal{P}(\mathbb{N}) \longleftrightarrow \mathbb{N}$, there is no bijection $\mathcal{P}(\Sigma^*) \longleftrightarrow \mathbb{N}$, because the composition of bijections is a bijection.

Since the set of Turing machines is countable and the set of all languages is uncountable, there must be an uncountable set of languages not decidable (or even recognizable) by some Turing machine.

■