COMS W3261
Computer Science Theory
Lecture 17
Undecidable Problems

Alexander Roth

2014 – 11 – 5

Outline

- Encodings
- Undecidable problems about Turing machines
- Undecidable problems about context-free languages

1 Encodings

- In this course, we have defined a problem as a question of deciding whether a given string is a member of some particular language.
- In formulating questions as membership problems for languages, we may need to encode objects, such as Turing machines and the inputs they process, as strings over some fixed alphabet, such as $\{0, 1\}$. Often the details of the encoding are not important as long as a Turing machine can decode the encoding so we will sometimes use the notation $< M >$ to represent an encoding of a Turing machine M_i without describing in detail how this encoding has been done.
- For example, using this notation, we can represent the universal language L_u as $\{< (M, w) > \mid < M >$ is an encoding of a Turing machine M, $< w >$ is an encoding of an input string w and M accepts $w\}$.
- If it is clear from the context, we will regard strings as the Turing machines and input strings they represent. Thus, we will often write $L_u = \{ (M, w) \mid M$ is a Turing machine and M accepts $w\}$.

2 Undecidable Problems about Turing Machines

- Definte L_e to be $\{M \mid L(M) = \emptyset\}$. In words, L_e is the language consisting of all encoded Turing machines whose language is empty.
  - We can now show that L_e is not recursively enumerable. See HMU Section 9.3.2 for a proof.
- Define L_{ne} to be $\{ M \mid L(M) \neq \emptyset\}$. L_{ne} is the complement of L_e.
  - We can show the L_{ne} is recursively enumerable but not recursive. Again, see HMU Section 9.3.2 for a proof.
  - We have just state that L_{ne} is not recursive. If L_e were recursively enumerable, both it and L_{ne} would be recursive because L_e and L_{ne} are complements of each other. This is the reason why L_e cannot be recursively enumerable.
- The Halting Problem
  - In Alan Turing's original formulation of Turing machines acceptance was just by halting not necessarily by halting in a final state.
  - We can define $H(M)$ for a Turing machine M to be the set of input strings w such that M halts on w in either a final or a nonfinal state.
  - The famous *halting problem* is the set of pairs $\{(M, w) \mid w \text{ is in } H(M)\}$.
  - We can show the halting problem is recursively enumerable but not recursive.

3 Undecidable Problems about Context-free Languages

- We have already shown that it is undecidable whether a context-free grammar is ambiguous. The proof technique we used was to reduce Post's Correspondence Problem to the ambiguity problem for CFG's.
- We can also use this proof technique to show a number of important problems about context-free languages are undecidable.
- Let G and H be CFG's, and let R be a regular expression. The following problems are undecidable:
  1. Is $L(G) \cap L(H) = \emptyset$?
  2. Is $L(G) = L(H)$?
  3. Is $L(G) = L(R)$?
  4. Is $L(G) = \Sigma^*$ for some alphabet Σ?
  5. Is $L(G) \subseteq L(H) = \emptyset$?

6. Is $L(R) \subseteq L(G) = \emptyset$

- Proofs of these results are in HMU Section 9.5.3.