Section 7 Handout

CS 121

November 7, 2010

Today's Topics

- Decidability, Recognizability, Co-Recognizability
- Undecidability
- Reductions

1 Decidability, Recognizability, Co-Recognizability

Quickly, what does it mean for a language to be *decidable*? *Recognizable*? *Co-Recognizable*?

Some useful facts:
A language L is *decidable*, or *recursive* if:

- L is finite, regular, or context-free. (Why?)
- L can be enumerated by a Turing machine in lexicographic order. (Why?)
- L is both recognizable and co-recogizable (Why?)
- $\overline{L}$ is decidable. (Why?)

A language L is *recognizable*, or *recursively-enumerable* (r.e) if:

- L is decidable. (Why?)
- L can be enumerated by a Turing machine. (Why?)

A language L is *co-recognizable*, or *co-r.e* if:

- L is decidable. (Why?)
- There exists a Turing machine that will always reject w if $w \notin L$, but that may either run forever or accept if $w \in L$. (Why?)

Exercise 1.1. *True or false?*

(a) If L is undecidable, then $\overline{L}$ is undecidable.

(b) If $\overline{L}$ is not recognizable, then L is not co-recognizable.

(c) If $\overline{L}$ is r.e and L is not recursive, then L is not r.e.

2 Undecidability

This is probably the most important thing that happens in this class. In fact, its arguably one of the most important mathematical ideas of the last century. Recall the definition of the language $HALT_{TM}$:

$$HALT_{TM} = \{\langle M, w\rangle : M \text{ eventually halts on } w\}$$

From class, we've shown that this language is undecidable. There exists no Turing machine that, given an encoding of another TM M and a string w, can always decide whether or not M would halt on w.
This is big, big news! These Turing machines are the most powerful model of computation weve come up with. As weve seen in lecture, they can carry out any computation that a computer can. And yet, weve found a simple, useful language that no Turing machine can decide.

Exercise 2.1. *Is $HALT_{TM}$ recognizable? Co-recognizable?*

Another common undecidable language that we've seen is:

$$A_{TM} = \{\langle M, w\rangle : M \text{ accepts } w\}$$

Exercise 2.2. *Is A_{TM} recognizable? Co-recognizable?*

Not only that, we saw in class that Rice's Theorem says that *every* nontrivial property of Turing-recognizable languages is undecidable! This means that given a TM M, it is undecidable to tell if L(M) is $\emptyset$, L(M) is regular, L(M) is infinite, etc.
Proof sketch: Reduce L_ϵ to L_P–since L_ϵ is undecidable, so is L_P.

3 Reductions

- We say that L_1 *reduces* to L_2 if we use L_2 to build an algorithm for L_1.
- A function $f : \Sigma_1^* \to \Sigma_2^*$ is **computable** if we can build a TM that halts with $f(w)$ on its tape for every input $w \in \Sigma_1^*$.
- A **mapping reduction** of $L_1 \subseteq \Sigma_1^*$ to $L_2 \subseteq \Sigma_2^*$ is a computable function $f : \Sigma_1^* \to \Sigma_2^*$ such that for any $w \in \Sigma_1^*$, $w \in L_1$ iff $f(w) \in L_2$. If this is so, we write $L_1 \leq_m L_2$.
- If $L_1 \leq_m L_2$, then
  - If L_2 is decidable, recognizable, or co-recognizable, then so is L_1.
  - If L_1 is undecidable, not recognizable, or not co-recognizable, then so is L_2.
- Example reductions:
  - For every Turing-recognizable L, $L \leq_m A_{TM}$.
  - $A_{TM} \leq_m HALT_{TM}$.
  - $HALT_{TM} \leq_m L_\epsilon$.

4 Exercises

Exercise 4.1. *Show that the language* $\{\langle M \rangle : a \in L(M)\}$ *is undecidable.*

Exercise 4.2. *Show that the following language is undecidable:*

$$L = \{\langle M, x \rangle : \textit{At some point in its computation on input } x,\ M \textit{ re-enters its start state.}\}$$

Exercise 4.3. *Using reductions, show that the following language is neither r.e nor co-r.e.*

$$L = \{\langle M \rangle : L(M) \textit{ is regular}\}$$