Undecidable Languages

CS154
Chris Pollett
Apr 17, 2006.

Outline

- Diagonalizing Languages
- The Halting Problem

Diagonalizing Languages

- Another corollary to the Diagonalization Theorem of last lecture is the following:

Corollary. Some languages are not Turing Recognizable.

Proof. Last lecture, we argued the interval (0,1) is uncountable. For the same reason the set of infinite strings over {0,1} is uncountable. We can view such a sequence (0,1, 0, 0 ..) as coding a language over some alphabet. Put a 1 in a position if the *i*th string is in the language and a 0 otherwise. On the other hand, each encoding <M> of a Turing Machine is a finite string over a finite alphabet and we argued last day that the set of finite strings over an alphabet is countable.

The Halting Problem is Undecidable

Theorem. The language A_{TM}={<*M,w*> | *M* is a TM and *M* accepts *w*} is undecidable.

Proof. Suppose *H* is a decider for A_{TM}. Fixing *M* and then listing out encoding of TM's in lex order <M_0>, <M_1>,.. we can view H as giving an infinite binary sequence where we have a 1 in the *i*th slot if <M_i> is in *M*'s language and a 0 otherwise. We will argue if *H* us a decider A_{TM} then there is a decider for the complement of the diagonal of this map. Here's how we do this, let D be the machine:

D="On input <*M*>, where *M* is a TM:

1. Run H on input <*M*, <*M*>>
2. Output the opposite of what *H* outputs; that is, if *H* accepts, then reject. If *H* rejects then accept."

Now consider *D*(<*D*>).Machine D accepts if and only if *H* on input <D, <D> rejects. But *H* on input <D, <D>> reject means that D did not accept input <D>. This is contradictory. A similar argument can be made about if D rejects <D>. So *H* must not exist.

A Specific Non-Turing Recognizable Language I

- Our Corollary on the third slide only shows some Turing unrecognizable language must exist -- it doesn't give us an example.
- We'll use the next theorem to give an example.
- First, call a language **co-Turing recognizable** if its complement is Turing recognized.

Theorem. A language is decidable iff it is Turing-recognized and co-Turing recognized.

Proof. Suppose L is decidable by M. Then it is also Turing Recognized. Further, let $\overline{M}$ be the machine which reject when M accepts and accepts when M rejects. Then $\overline{M}$ recognizes the complement of L. On the other hand, suppose L′ is Turing recognized by M′ and co-Turing recognized by M″. Then Let D be the machines which on input w simulates each of M′ and M″ first for 1 step, then for 2 steps, etc. If M′ ever accepts the D accepts and if M″ ever accepts then D rejects. Since a string is either in L′ or not, one of these two machines must accept eventually, and so then D will decide that string.

A Specific Non-Turing Recognizable Language II

Corollary. $\overline{A}_{TM}$ is not Turing recognized.

Proof. We proved in an earlier lecture A_{TM} is Turing recognized. So if $\overline{A}_{TM}$ were Turing recognized, then A_{TM} would be decidable giving a contradiction with the halting problem being undecidable.