More Undecidable Languages

CS154
Chris Pollett
May 7, 2007.

Outline

- Refresher on A_{TM}
- More undecidable languages

A_{TM} is not Recursive

Theorem. The language A_{TM}= {<*M,w*> | *M* is a TM and *M accepts* on *w*} is not recursive.

Proof. Suppose *A* is a decider for A_{TM}. Fix M_i and consider w's of the form <M_j> for some other TM, M_i. Then listing out encodings of TM's in lex order <M_0>, <M_1>,.. we can create an infinite binary sequence where we have a 1 in the *j*th slot if <M_j> causes M_i to accept and a 0 otherwise. If *A* is a decider A_{TM} then we can consider a variant on the complement of the diagonal of the map f:<M_i> |--> (A(<M_i,<M_0>>), A(<M_i,<M_1>>),..). In particular, we can let D be the machine:

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

- Run A on input <*M*, <*M*>>
- If *A* says Yes, then run forever. If *A* says no, then say halt and accept."

Now consider *D*(<*D*>). Machine D halts if and only if *A* on input <D, <D>> rejects. But *A* on input <D, <D>> rejects means that D did not halt on input <D>. This is contradictory. A similar argument can be made about if D does not halt <D>. Since assuming the existence of *A* leads to a contradiction, hence *A* must not exist. Q.E.D.

Another way to look at this is if you give an *A* which purports to be a decider for A_{TM} then we can give a specific input, <D, <D>>, which is calculated based on *A* on which *A* fails.

A Specific Nonrecursively Enumerable Language I

- Last day we gave a counting argument to show a non recursively enumerable language must exist - our argument though doesn't give a specific example language.
- We'll use the next theorem to give an example.
- First, call a language **co-recursively enumerable** if its complement is recursively enumerable.

Theorem. A language is decidable iff it is recursively enumerable and co-recursively enumerable.

Proof. Suppose L is decidable by M. Then it is also r.e. Further, let $\overline{M}$ be the machine which reject when M accepts and accepts when M rejects. The $\overline{M}$ 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-Recursively Enumerable Language II

Corollary. $\overline{A}_{TM}$ is not r.e.

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

Reducibility

- We next consider what other problem are undecidable.
- Our approach to showing languages are undecidable will be to use a notion called **reducibility**.
- A **reduction** r is a mapping from possible inputs I_A to a problem A, **instances** of A, to instances of problem B, with the property that $I_A \in A$ if and only if $r(I_A) \in B$.
- If the reduction can be computed by a TM, i.e., a **Turing reduction**, then if B is decidable then A will be too. Conversely, if A is not decidable, then B also won't be decidable.

Example

- Let $HALT_{TM}$ = {<*M,w*> | *M* is a TM and *M* halts on input *w*}.

Theorem. $HALT_{TM}$ is undecidable.

Proof. Suppose *H* decides $HALT_{TM}$. From *H* we can construct a machine *S* which decides A_{TM} as follows:

S =" On input <*M, w*> an encoding of a TM *M* and a string w:

1. We build a new string <M´,w> where M´ is a machine which simulates M until M halts (if it does) and if M accept M´ accepts. Otherwise, if M reject that M´ moves right forever one, square at a time. The map <M,w> --> <M´,w> is well defined enough that it can be computed by a TM. This is our reduction.
2. We then ask our decider for H if <M´,w> is in H. If H accepts we accept and if H rejects we reject."

- Since the only way M´ on input w halts is if M accepts w, we know <M,w> is in A_{TM} iff <M´,w> is in $HALT_{TM}$. So if H was a decision procedure for $HALT_{TM}$, then S would be a decision procedure for A_{TM}. As we know there is no decision procedure for A_{TM} we know that the supposed H can't exist.

A Problem about Regular Languages

- Even problems about regular languages can sometimes be hard. Let:
  $Regular_{TM}$ = {<M> | M is a TM and L(M) is a regular language}.

Theorem. $Regular_{TM}$ is undecidable.

Proof. Suppose R decides $Regular_{TM}$. Then the following machine decides A_{TM}:

S="On input <*M,w*>, where *M* is a TM and *w* is a string:

1. Construct the following machine M_2:

   M_2 = "On input *x*:

   - If *x* has the form 0^n1^n, accept.
   - If *x* does not have this form, run *M* on input *w* and accept if *M* accepts *w*."

   // So if *M* accepts *w*, then M_2 accepts all strings; otherwise, M_2 only accepts strings of the form 0^n1^n.

2. Run *R* on input <M_2>.
3. If *R* accepts, accept; otherwise, if *R* rejects, reject."