Diagonalization

CS154
Chris Pollett
May 2, 2007.

Outline

- Diagonalization
- The Halting Problem is Undecidable

Introduction

- Recall last day we considered the language:

  A_{TM}={<M,x> | M is the encoding of a TM which when run on input x accepts}.

- We gave a last day a procedure for a TM to recognize this language (this is what a Universal TM does) and we said that there is no procedure for a TM to decide this language.
- Today, we are going to prove this second statement.
- Before we do let's define a language to be **recursive enumerable** if there is some some TM which recognizes the language.
- Define a language to be **decidable** or **recursive** if there is some TM which decides the language.
- So we have shown A_{TM} is recursively enumerable and we'd like to show it is not decidable. To do this we need a slight digression…

Sizes of Sets

- In the 1870's Georg Cantor was interested in figuring out when two sets are of the same size.
- In particular, he was worried about infinite sized sets.
- He argued two sets A, B should be said to be of the same size if there is a one-to-one, onto function (a **bijection**) between them.
- Recall **one-to-one** means $a \neq b$ implies $f(a) \neq f(b)$ and **onto** means for every element b in B, there is some a in A such that $f(a) = b$.
- For example the map $f(k)=2k$ is a bijection between the integers and the even integers.
- A set is said to be **countable** if there is a bijection between it and a subset of the naturals. Otherwise, a set is said to be uncountable.
- For example, the rational numbers and the set of finite strings over are {0,1} are countable. (will doodle on board why, but also see book).

Diagonalization

- Suppose f is a one-to-one function from a countable set A={a(0), a(1), a(2), …} to sequences of elements over some set B of size at least 2, such that the length of the sequence f(a(i)) is at least i.
- For example,

  f(a(0)) = (1, 0, 1)

  f(a(1)) = (0, 0, 0)

  f(a(2)) = (0, 1, 1)
- Let $f(a(i))_j$ denote the jth element of the sequence f(a(i)).
- The diagonal of this function is the function of f is the sequence $d(f)=(f(a(0))_0, f(a(1))_1, f(a(2))_2,\ldots)$.
- So in this case d(f) = (1, 0, 1).
- Call a sequence d'(f) a **complement** of the diagonal if $d'(f)_i$ is always different from $d(f)_i$.
- For example, for the f above a possible d'(f) is (0, 1, 0).
- The following theorem is an easy consequence of our definition.

Theorem (Diagonalization Theorem) If f satisfies the first bullet above then it does not map any element to a complement of its diagonal.

Example Use of the Diagonalization Theorem

Corollary. A countable set A is not the same size as its P(A).

Proof. Let f:A --> P(A) be a supposed bijection. Since A is countable, we have some function a(k) to list out its elements a(0), a(1), a(2), …An element {a(2), a(5), ..}∈P(A) can be view as an binary sequence (0, 0, 1, 0, 0, 1, …) where we have a 1 if a(i) is in P(A) and a 0 otherwise. So f satisfies the Diagonalization theorem. A complement of the diagonal for f will still be in P(A) but not mapped to by f.

- A set which is not countable is **uncountable**.
- Let **N** be the natural numbers. So P(**N**) is uncountable.

Non Recursively Enumerable Languages

Another corollary to the Diagonalization Theorem is the following:

Corollary. Some languages are not recursive enumerable.

Proof. The set of infinite sequences over {0,1} is uncountable, as we just indicated in the last proof there is a bijection between this set and P(N). On the other hand, each encoding <M> of a Turing Machine is a finite string over a finite alphabet and we argued earlier today that the set of finite strings over an alphabet is countable.

A_{TM} is not Recursive

Theorem. The language A_{TM}= {<*M*,*w*> | *M* is a TM and *M* halts 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 halt 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 H on input <*M*, <*M*>>
- If *H* says Yes, then run forever. If *H* 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.