Sizes of infinite sets

A function from set A to B is *1-1* if it never maps two elements of A to the same element of B. It is *onto* if for every $b \in B$ there is an $a \in A$ such that $f(a) = b$.

With a 1-1 onto function $f : A \to B$ we can pair every element in A with every element in B. A set is *countable* if it is finite or it can be paired with the natural numbers.

e.g. the even numbers, rational numbers.

Paradoxes

- The Paradox of the Liar: "This sentence is not true."

- The Barber's Paradox: The barber cuts the hair of everyone in the town who doesn't cut his or her own hair.

Cantor's Theorem

- The proof is an example of a diagonalization argument.
- Let S be a countably infinite set, say $S = \{x_1, x_2, \ldots\}$. $\mathcal{P}(S)$ is the set of all subsets of S. This set is infinite but not countably infinite, i.e., it is *uncountably infinite*.
- Proof: Suppose to the contrary that $\mathcal{P}(S)$ is countable. Let f be a 1-1 and onto function from N to $\mathcal{P}(S)$. Define the set $T = \{x_i \mid x_i \notin f(i)\}$.
- Since T is in $\mathcal{P}(S)$ there must be some j such that $f(j) = T$.
- Is x_j in T?
- Neither Yes nor No. Hence our assumption, that $\mathcal{P}(S)$ was countable is wrong.

A counting argument

- Theorem: There are languages which are not Turing recognizable.
- Say $\Sigma = \{0, 1\}$, so every binary string can be assigned a distinct natural number by just putting a 1 in the front – the set of binary strings is thus countably infinite.
- The set of all possible languages over $\{0, 1\}$ is the power set of the set of binary strings. By Cantor's Theorem, this set is uncountably infinite.
- The set of TM's is countable because it can be described by a finite string over a finite alphabet – every TM can be encoded by a unique binary string.
- Since each Turing machine accepts one language, there are only countably infinite Turing recognizable languages.
- Hence, since there are an uncountable number of languages, there are languages which are not recognized by any TM.
- Can we show an *explicit* language which is not Turing recognizable? We will start by showing a language that is not decidable.

The Acceptance Problem is undecidable

Theorem: $A_{TM} = \{\langle M, w\rangle \mid M \text{ is a } TM \text{ and } M \text{ accepts } w\}$ is undecidable.

Proof: Assume it's decidable and show a contradiction. Let H be a TM which decides A_{TM}, i.e., H is halting, and H accepts input $\langle M, w\rangle$ iff M accepts w

We construct a new TM D which uses H as a subroutine. D takes as input any TM description $\langle M\rangle$ and simulates H on $\langle M, \langle M\rangle\rangle$ When H halts, D enters the opposite final state. So

- D accepts $\langle M\rangle$ if M rejects $\langle M\rangle$
- D rejects $\langle M\rangle$ if M accepts $\langle M\rangle$

Now, what happens when D is given $\langle D\rangle$ as input? D accepts $\langle D\rangle$ iff D rejects $\langle D\rangle$!

This is a contradiction. Therefore D and H can't exist.

Can view as a diagonalization argument in a table.

A_{TM} is Turing recognizable

Recall that we are assuming a standard way of encoding the pair $\langle M, w\rangle$ as a string

Given this input, we want to *simulate* the computation of M. Use a 4-tape TM, which we call U.

- Tape 1 stores the string encoding $\langle M, w\rangle$, (the input to U)
- Tape 2 stores the simulated tape of M.
- Tape 3 stores the state of M
- Tape 4 is scratch.

Steps of the simulation

- Examine the code to make sure it's for a legitimate TM. If not, halt without accepting.
- Initialize the second tape by putting w on it
- Place the start state 1 on tape 3. Move the head of the second tape to the leftmost simulated cell.
- To simulate a move:
  - Based on the state on tape 3, and symbol scanned on tape 2, search through the description of M on tape 1 until we find the appropriate transition.
  - Update the contents of tape 2, and the state on tape 3, based on this transition
- If M has no transition that matches the symbol being read, U halts.
- If M enters an accepting state, U accepts.

A Turing Unrecognizable Language

A language is *co-Turing recognizable* if its complement is Turing recognizable.

Theorem: If a language is Turing-recognizable *and* co-Turing recognizable, then it is decidable.

Proof: Run the TM for recognizing the language and the TM for recognizing its complement in parallel. If the former accepts, accept. If the latter accepts, reject.

Corollary: The complement of A_{TM} is not Turing recognizable.

Decidable languages and their complements

From the preceding slides, we see that there are languages which are Turing-recognizable, but whose complements are not Turing recognizable. What about decidable languages?

Theorem: If a language L is decidable, so is its complement.

Proof: Let M be the halting TM which accepts L.

- Change accepting states to nonaccepting states.
- Make a new accepting state and a transition to it from every nonaccepting state labeled with every tape symbol such that there was no transition out of that state with that label (in the old machine).