Computable Sets

KR Chowdhary
Professor & Head
Email: kr.chowdhary@acm.org

Department of Computer Science and Engineering
MBM Engineering College, Jodhpur

March 19, 2013

Computable sets

- **Definition:** *A* set $A \subseteq \mathbb{N}$ is computable if there is a computer program that, on input n, decides whether $n \in A$.
- **Church-Turing thesis:** This definition is independent of the programing language chosen.
- **Examples:** The following sets are computable:
  1. The set of even numbers.
  2. The set of prime numbers.
  3. The set of stings that correspond to well-formed programs.
- Recall that any finite object can be encoded by a natural number(e.g. an algorithm, a program, a large prime number, a large composite number, code of a Turing machine, etc).

Examples of non-computable sets

- **The word problem:** Consider the groups that can be constructed with a finite set of generators and a finite set of relations between the generators. The set of pairs (set-of-generators, relations), of *non-trivial* groups is not computable. For example, the generators can be grammar's set, and relations can be acceptability/rejection relation between grammar and strings.
- **Simply connected manifolds:** The set of finite triangulations of *simply connected* manifolds is not computable.
- **The Halting problem:** The set of programs that halt, and don't run for ever, is not computable.

Basic definitions

- Given sets $A, B \subseteq \mathbb{N}$ we say that "A is computable in B", and we write $A \leq_T B$, if there is a computable procedure that can tell whether an element is in A or not using B as an oracle.
- We say that A is Turing equivalent to B, and we write $A \equiv_T B$ if $A \leq_T B$ and $B \leq_T A$.
- **Example:** The following sets are Turing equivalent.
  1. The set of pairs (set-of-generators, relations), of non-trivial groups;
  2. The set of finite triangulations of simply connected manifolds;
  3. The set of programs that halt.

What is a computable set

- We fix some model of computation (typically Turing machines).
- Fix a set $A \subseteq \mathbb{N}$.
- **Definition:** A is computable iff there is a finite program (for the model of computation) that gets as input any number n and will output a correct yes/no answer to the question "$n \in A$?".
- **Intuition:** Finite information effectively describes the set completely. Example: The set of prime numbers.
- **Finite description:** Program that describes a procedure that tries to factor the number, and outputs "no" if it finds prime factors, and "yes" if it doesn't.

The Kolmogorov complexity of a set

- Again, fix a set $A \subseteq \mathbb{N}$.
- We identify it with an infinite sequence of 0's and 1's.
- We want to quantify how difficult it is to describe A (in our computational model).
- Most of the time, this will require infinitely much information.
- So instead: Look at parts of A and investigate how much information we need to describe those: $A|n := A \cap \{0, ..., n-1\}$.
- **Example:**

  Any computable set is of minimal complexity.

  Sets that correspond to sequences with "few regularities" have high complexity.

Traceable sets

- Two ways of looking at sets: Computability and compressibility.
- We Look at something called "traceability".
- That is a notion that describes that a set is "nearly computable".
- We will see: Correspond to some notion of "quite well compressible".

Motivation of traceability

- Assume we have some function f, and we can compute it (i.e., there is a program), but only with some external information that we need from a set A. We write $f \leq_T A$.
- Assume that in addition A is computable (i.e., there is a program).
- We can build together both programs into one program that directly computes f.
- In other words: If A is computable then all f that are computable in A are computable, too.
- Next, we want to model "close to computable": Stipulate that all $f \leq_T A$ are "close to computable".
- **Intuition:** A is so easy that it contains so little information that we cannot use it to compute anything too complicated.
- f "close to computable" means: We cannot necessarily compute f, but can, given n, generate a small list of potential values of $f(n)$, including the correct value.