Algorithm Analysis and Design (CS1.301)

Monsoon 2021, IIIT Hyderabad
21 August, Saturday (Lecture 2)

Countability and Computability

We wish to show that there are problems which cannot be solved computationally. We note that computational solutions (or programs) are countably infinite in number, while there is an uncountably infinite number of problems.

We can assume that there is an injective encoding from computer programs to finite-length bitstrings. Then, if the set of finite-length bitstrings (*i.e.*, $\{0,1\}*$) is countable, so is the set of computational solutions. We can prove this by noting that the set of binary strings has an ordering, the *short-lex order* – ε, `0`, `1`, `00`, `01`, `10`, and so on – which can be mapped to the natural numbers in that order. Finding a formal bijection from this intuition is straightforward.

A problem is encoded, as we know, as a *set* of all possible bitstrings. Therefore any subset of $\{0,1\}*$ can be considered a problem, *i.e.*, the number of problems is $\mathcal{P}(\{0,1\}*)$. This set is uncountable (since the powersets of countably infinite sets are uncountable), which completes our proof.

Another way to prove the uncountability of $\mathcal{P}(\{0,1\}*)$ is by the diagonalisation argument. Assume that there are countably many languages. Now, consider the language which contains the i^{th} string iff L_i does not contain it, and vice versa. This language cannot coincide with any of the languages already enumerated; therefore there cannot be such an enumeration.

A common example of a problem which is not computationally solvable is the *Program Equivalence Problem* – to identify if two programs solve the same problem or not.

The Church-Turing Hypothesis

Turing Machines

What we informally call an algorithm is simply a Turing machine. A Turing machine is consists of an infinite *tape*, a read-write *head* and a set of states.

Formally, a TM is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where

- Q is the set of states.
- Σ is the input alphabet (does not contain a blank)
- Γ is the tape alphabet (contains a blank)
- δ is the transition function $Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
- q_0 is the start state.
- q_{accept} is the accept state.
- q_{reject} is the reject state.

Most programming languages are Turing-complete; thus if no C program exists for a problem, then no Turing machine (hence no algorithm) exists for it.