Advanced Topics in Computability

1 Diagonalisation

Definition: Countable

A set S is countable if there is a one-to-one correspondence between S and the set of natural numbers $\mathbb{N}$

2 Cantor's Proof

Proposition: The set of reals in the interval (0,1) is uncountable

Proof: A real number A in (0,1) is an (infinite) decimal expansion: $A = 0.a_1a_2a_3...$

Assume, for the sake of contradiction, there is a one-to-one correspondence between the real interval (0,1) and $\mathbb{N}$, i.e. all the reals in (0,1) can be ordered in a sequence

$$A_1, A_2, A_3...$$

We will construct a real number which is not in the sequence

3 Cantor's diagonal argument

Denote $A_i = 0.a_1^i a_2^i a_3^i....$ and put the sequence in the following rectangular table

$$\begin{array}{lccccccccc}
A_1 = & 0 & . & a_1^1 & a_2^1 & a_3^1 & \dots & \dots & \cdots \\
A_2 = & 0 & \cdot & a_1^2 & a_2^2 & a_3^2 & \dots & \dots & \cdots \\
A_3 = & 0 & \cdot & a_1^3 & a_2^3 & a_3^3 & \dots & \dots & \cdots \\
 & \vdots & & & & \ddots & & & \\
A_i = & 0 & . & a_2^i & a_3^i & \dots & a_i^i & \cdots & \cdots \\
\vdots & & & & & & \ddots & & \\
\vdots & & & & & & & \ddots & \\
 & & & & & & & & \ddots
\end{array}$$

Construct a new number $B = 0.b_1b_2b_3...$ by taking

$$b_i = \begin{cases} a_i^i + 1 & \text{if } a_i^i < 9 \\ 0 & \text{if } a_i^i = 9 \end{cases}$$

Now, B is a real number in (0,1) which is not in the table above, as $b_i \neq a_i^i$ for every i

4 Halting problem by diagonalisation

The set of all strings over a finite alphabet is countable - order them by length first and order the ones of the same length in lexicographic order

$$\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, \dots$$

Therefore, the set of all Turing machines is countable, too. Put all TMs vs all inputs in an infinite table.

$HALT\ (M,w)$	w_0	w_1	$\cdots$	w_i	$\cdots$	w_j	$\cdots$
M_0	h_{00}	h_{01}					
M_1	h_{10}	h_{11}					
$\vdots$			$\ddots$			$\vdots$	
M_i			$\cdots$	h_{ii}	$\cdots$	$h_{ij} = \begin{cases} 1 & M_i \text{ halts on } w_j \\ 0 & \text{otherwise} \end{cases}$	$\cdots$
$\vdots$					$\ddots$	$\vdots$	

With the help of HALT machine, we created a TM M tat everywhere disagrees with the diagonal

5 The class of Nice machines

A set of Turing machines $\mathcal{N}$ has a Universal machine $U_{\mathcal{N}}(i,w)$ if

1. For every machines $N \in \mathcal{N}$, there is a number n such that $N(w) = U_{\mathcal{N}}(n,w)$ or all inputs w
2. For every number n, the machine $U_{\mathcal{N}}(n,.) \in \mathcal{N}$

Definition: Nice machines

The class of "nice" machines $\mathcal{N}$ is the set of all TMs that terminate on every input

Proposition: The class of "nice" machines $\mathcal{N}$ does not have a universal machine

Proof: Assume that there is a universal function $U_{\mathcal{N}}(i,w)$. Diagonalise: consider the machine M defined by

$$M(w_i) = \neg U_{\mathcal{N}}(i, w_i)$$

for all i

M itself is a nice machine, so there much be a number n such that $M(w) = U_{\mathcal{N}}(n,w)$ for all inputs w. In particular, for $w = w_n$ we would have that

$$M(w_n) = U_{\mathcal{N}}(n, w_n)$$

However, but by the construction of M we have that

$$M(w_n) = \neg U_{\mathcal{N}}(n, w_n)$$

which is a contradiction

6 Self-Reference

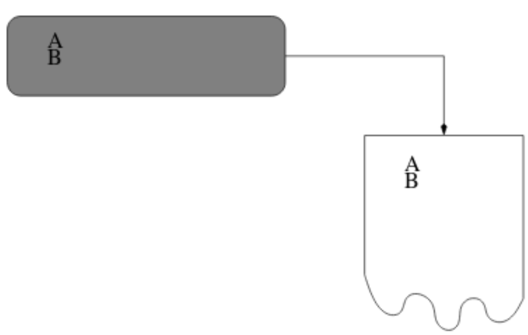
We want a program (Turing machine) that ignores the input and produced its own source code (description) as output.

Definition: Quine

A program that generates a copy of its own source code as its complete output

7 Solution by mutual recursion

A quine that consists of two parts: A followed by B. A prints out B in a straightforward way, and then B prints out A using the output that has just been produced by A.



8 m-reducibility

Definition. Let A and B be languages over the same alphabet Σ. A is a many-to-one reducible to B (write $A \leqslant B$) if there is a Turing machine F that terminates on every input $u \in \Sigma^*$, and such that

$$A\{u \in \Sigma^* | F(u) \in B\}$$

Informally: checking $u \in A$ is no harder than checking $w \in B$

8.1 Properties of m-reducibility

Proposition. Suppose $A \leqslant B$

1. If B is Turing-decidable, so is A
2. If B is Turing-recognisable, so is A
3. If $A \leqslant B$ and $B \leqslant C$, then $A \leqslant C$

Definition. Denote $A \equiv B$ to mean that $A \leqslant B$ and $B \leqslant A$

Informally: A and B are equally difficult

9 m-completeness

Definition. A language A is m-complete if

1. A is Turing-recognisable
2. For every Turing-recognisable language B, $B \leqslant A$

Informally: If A is m-complete then A is as hard as any other Turing-recognisable language

Corollary If A is m-complete and $A \leqslant B$, then B is m-complete

Definition - The Halting language H consists of the words $\langle M \rangle \circ w$ (over some fixed alphabet) such that the Turing machine M terminates on w

Theorem H is M complete

Proof: Generic reduction. Pick any Turing-recognisable language A. It is recognised by some machine M_A. Reduce it to H by mapping any word w onto the word $\langle M_A \rangle \circ w$. It is obvious that the reduction is computable and $w \in A$ iff $\langle M_A \rangle \circ w \in H$

Definition: H_0 is the "diagonal" of H, i.e. the language $\langle M \rangle \circ \langle M \rangle$ such that M terminates on $\langle M \rangle$

Theorem: H_0 is m-complete

Proof: Reduction from H. Given a word $\langle M \rangle \circ w$, create a Turing machine $N_{M,w}$ that simulates M on w (and note that it ignores the input) - this can be done using a universal Turing machine. Now, $N_{M,w}$ terminates on any input iff M terminates on w. In particular $N_{M,w}$ terminates on $\langle N_{M,w}$ iff M terminates on w

10 Oracle Turing Machine and t-reducibility

Definition

1. An oracle for a language A is a black-box that takes a word w as an input and instantly (and correctly) replies if $w \in A$
2. An oracle Turing machine M, denotes by M^A is a Turing machine that has an additional capability of making calls to an oracle for the language A

Definition: A language A is t-reducible to a language B is A is decidable by some oracle Turing machine M^B

Theorem: If $A \leqslant_t B$ and B is Turing-decidable, then A is Turing-decidable