3P17/4P17 COMPUTABILITY 2014-2015 Lecture Handout 7

Encoding general objects

We have said that for $N \in \mathbb{N}_0$, $\langle N \rangle$ is the minimal length binary representation of N.

In general, we may wish to encode objects that are not numbers. This would involve choosing some encoding scheme that makes use of some finite alphabet Σ. Under that scheme, an object O gets encoded into a string $\langle O \rangle \in \Sigma^*$. For multiple objects, we will write $\langle O_1, O_2, \ldots O_k \rangle$.

We need not define precisely what the encoding scheme is, as long as it is "reasonable", and reasonable schemes will result in finite length encodings for finite objects.

Thus, in our example of the Turing machine M_{A+B}, the input was $\langle A, B \rangle$ which was encoded as $\langle A \rangle * \langle B \rangle$.

Note, in particular, that it is possible to define encoding schemes to encode Turing machines. Hence, under some encoding scheme, a Turing machine M gets encoded into a finite-length string $\langle M \rangle$. (In fact, we have been encoding Turing machines into finite length strings all along, using the Latin and Greek alphabets, Arabic numerals, (and), *, etc).

Countability, Cantor's diagonalisation method

Two sets A, B have the same cardinality $|A| = |B|$ if there is a bijective mapping between them.

The cardinality of $\mathbb{N}$ is denoted $\aleph_0$ ("aleph-naught"). That is, $|\mathbb{N}| = \aleph_0$.

A set A is **countable** if it is finite or has the same cardinality as $\mathbb{N}$. If it is not countable, it is called **uncountable**.

It can be shown that $|\mathbb{Z}| = |\mathbb{Q}| = \aleph_0$.

Cantor's diagonalisation method is a way of proving some sets are not countable. It can be used to prove $|\mathbb{N}| \neq |\mathbb{R}|$. We shall use it to show the following:

Theorem 1. *Let S be a set such that $|S| = \aleph_0$. Then the power set $\mathcal{P}(S)$ of S is uncountable.*

Proof. Since S has the same cardinality as $\mathbb{N}$, we can write S as $\{f(1), f(2), \ldots\}$ where f is some bijection (which must exist) mapping $\mathbb{N}$ to S. Then a subset S' of S can be specified by infinite binary sequence $b_1 b_2 \ldots$ where each $b_i \in \{0, 1\}$. Specifically, $b_i = 1$ if and only if $f(i) \in S'$. Clearly, each subset of S maps to a unique infinite binary sequence and each infinite binary sequence specifies exactly one subset of S. Therefore, there is a bijection between the elements of $\mathcal{P}(S)$ and the set of all infinite binary sequences $\mathcal{B}$. If $\mathcal{P}(S)$ is countable, then there must also be a bijection $F : \mathbb{N} \to \mathcal{B}$.

We will show that there is some infinite binary sequence $b^* \in \mathcal{B}$ such that $\forall n \in \mathbb{N}$, $F(n) \neq b^*$. This will give us the contradiction we seek.

For any infinite binary string b, let $[b]_i$ be the i'th binary symbol in b. We Construction of b^* as follows: $[b^*]_i = 1 - [F(i)]_i$. That is, the i'th symbol of b^* is 1 if the i'th symbol of $F(i)$ is 0, and is 0 if the i'th symbol of $F(i)$ is 1.

for example, if F was as follows:

n	F(n)
1	`0110001...`
2	`1101001...`
3	`1111111...`
4	`1010010...`
⋮	⋮

then b^* would start $b^* = 1001\ldots$

Now suppose $F(n) = b^*$ for some $n \in \mathbb{N}$. Then $[b^*]_n = 1 - [b^*]_n$, and we have a contradiction. □

Cantor s diagonalisation method takes its name from the part of the proof above where a string is constructed by reading the diagonal of the table and choosing a different symbol.

Corollary 2. *Some languages are not Turing-recognisable*

Proof. For a finite alphabet Σ, $|\Sigma^*| = \aleph_0$, and a language over Σ is merely a subset of Σ^*. Therefore, the collection of all languages is the power set of Σ^*, which is uncountable.

On the other hand, every Turing machine M can be encoded into a finite string $\langle M \rangle$ over some encoding scheme alphabet Σ_e. Since Σ_e^* is countable, it means that the set of possible Turing machines is also countable. Since a Turing machine recognises exactly one language, there must be languages not recognised by Turing machines. □

We reiterate from the definition given in Handout 1, that any Turing machine M recognises exactly one language $L(M)$, the set of *all* strings over Σ that M halts with accept on (possibly $L(M) = \emptyset$). If L' is a *strict* subset of $L(M)$, then M does **not** recognise L', even though M will halt on every member of L'.