CA320 - Computability & Complexity

Decidability

Dr. David Sinclair

Diagonalisation

How do we compare the sizes of two infinite set?

For example comparing the set of natural numbers $\{1, 2, 3, \ldots\}$ with the set of even natural numbers $\{2, 4, 6, \ldots\}$ is one of these bigger that the other?

To answer this we need the concept of a *correspondence*. A *correspondence* is a function $f : A \to B$ that is:

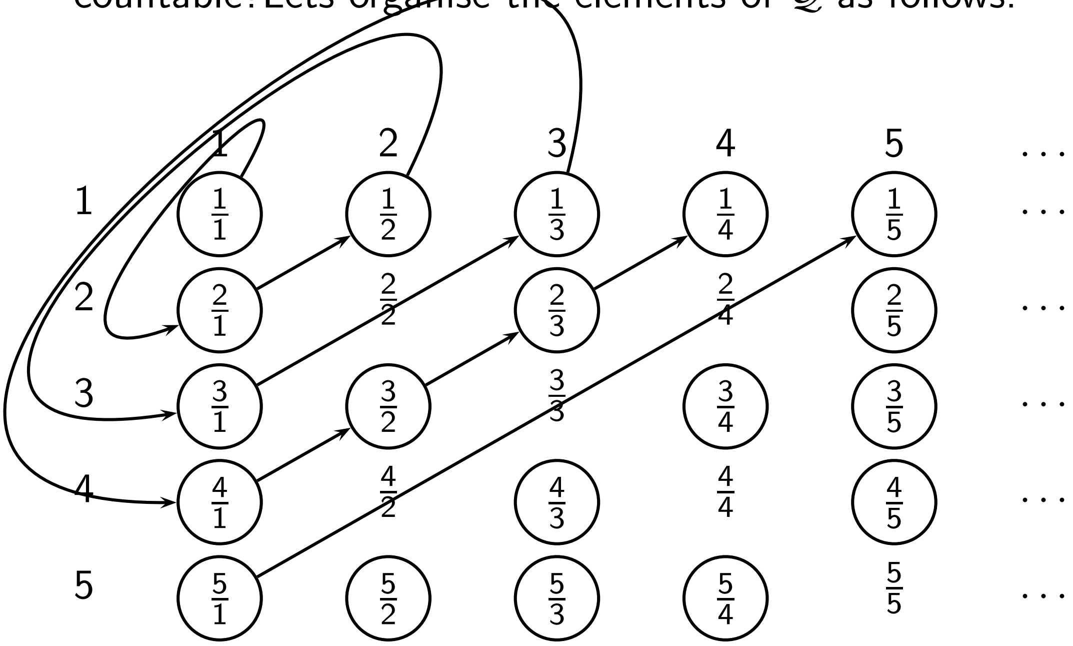
- **one-to-one**, that is it never maps two different elements of A to the same element of B; and
- **onto**, that is $\forall c \in B$, $\exists a \in A$ such that $f(a) = b$.

There is a *correspondence*, $f(n) = 2n$, between $A = \{1, 2, 3, \ldots\}$ and $B = \{2, 4, 6, \ldots\}$, so both sets have the same size.

Diagonalisation (2)

A set is countable if it is either finite or has the same size as $\mathcal{N}$.

Is the set $\mathcal{Q} = \{\frac{m}{n} | m, n \in \mathcal{N}\}$, the set of positive rational numbers, countable?Lets organise the elements of $\mathcal{Q}$ as follows:



Diagonalisation (3)

Each "bottom-left to upper right" diagonal is finite in size and hence we can list all the distinct elements of $\mathcal{Q}$. Since there is a correspondence between $\mathcal{Q}$ and $\mathcal{N}$, $\mathcal{Q}$ is countable.

Is the set $\mathcal{R}$, the set of real numbers, countable?

Theorem

The set $\mathcal{R}$ is uncountable.

Proof.

We will assume a correspondence f exists and show this generates a contradiction, hence no correspondence really exists and $\mathcal{Q}$ is uncountable. We will do this by constructing $x \in \mathcal{R}$ and showing that it cannot be paired with anything in $\mathcal{N}$.

Diagonalisation (4)

Proof (contd.)

Suppose a correspondence f exists. Here is a hypothetical example

n	$f(n)$
1	$0.\underline{1}4159\ldots$
2	$0.5\underline{5}555\ldots$
3	$0.12\underline{3}45\ldots$
4	$0.500\underline{0}0\ldots$
$\vdots$	$\vdots$

Construct x as follows. Let the i-th fractional digit of x not be equal to the i-th fractional digit of $f(i)$.
e.g. $x = 0.4641\ldots$
Hence $\forall i \in \mathcal{N},\ x \neq f(i)$.

Since we have created an x that cannot be paired with any element of $\mathcal{N}$ our assumption that a correspondence existed is false and $\mathcal{R}$ is uncountable. $\square$

This is an example of the diagonalisation techniques developed by Georg Cantor in 1873.

Undecidability and the Halting Problem

Theorem

The language $A_{TM} = \{\langle M, w\rangle | M$ *is a Turing machine and* M *accepts* $w\}$ *is undecidable.*

Proof.

Let's assume a Turing machine H is a decider for A_{TM}.

$$H(\langle M, w\rangle) = \begin{cases} \textit{accept}, & \text{if } M \text{ accepts } w \\ \textit{reject}, & \text{if } M \text{ halts and does not accept } w \end{cases}$$

Let D be a Turing machine that uses H as follows.

D(M) = **1.** Run H on input $\langle M, M\rangle$.
2. Output the opposite of what H produces.

Therefore,

$$D(M) = \begin{cases} \textit{accept}, & \text{if } M \text{ halts and does not accept } M \\ \textit{reject}, & \text{if } M \text{ accepts } M \end{cases}$$

Undecidability and the Halting Problem (2)

Proof (contd.)

If we run D on its own description we get:

$$D(D) = \begin{cases} \textit{accept}, & \text{if } D \text{ halts and does not accept } D \\ \textit{reject}, & \text{if } D \text{ accepts } D \end{cases}$$

This contradiction implies that the initial assumption that A_{TM} is decidable is false. □

This is just another version of the diagonalisation argument. Consider the table $H(M_i, M_j)$ with some fictional values.

	M_1	M_2	M_3	...	D	...
M_1	**accept**	reject	accept	...	accept	...
M_2	accept	**reject**	accept	...	accept	...
M_3	accept	reject	**accept**	...	reject	...
$\vdots$						
D	reject	accept	reject	...	?	...
$\vdots$						

Undecidability and the Halting Problem (3)

Theorem

The language $HALT_{TM} = \{\langle M, w\rangle | M$ is a Turing machine and M halts on input $w\}$ is undecidable.

Proof.

Assume that R is a Turing machine that decides $HALT_{TM}$. Using R we can construct a TM S that decides A_{TM}. S operates as follows on the input $\langle M, w\rangle$ where M is a Turing machine and w is a string.

1. Run R on $\langle M, w\rangle$.
2. If R rejects $\langle M, w\rangle$, then reject.
3. If R accepts, then simulate M on w until it halts.
4. If M accepts, then return accept else return reject.

But since A_{TM} is undecidable then the initial assumption that there exists a Turing machine that decides $HALT_{TM}$ must be false. □

Reducability

Rather than using a diagonalisation proof to show the undecidability of a language L_1 we can map/reduce the language to a language L_2 whose decidability is already known.

The language L_1 is *mapping reducible* to language L_2, written $L_1 \leq L_2$, if there is a computable function $f : \Sigma^* \to \Sigma^*$ where for every w,

$$w \in L_1 \Leftrightarrow f(w) \in L_2$$

If $L_1 \leq L_2$ and L_2 is decidable, then L_1 is decidable.

If $L_1 \leq L_2$ and L_2 is undecidable, then L_1 is undecidable.

Other Undecidable Problems

Some examples of:

- Given a Turing machine M, is there any string at all on which M halts?
- Given a Turing machine M, does M halt on every input?
- Given two Turing machines M_1 and M_2, do they halt on the same input strings?
- Given a Turing machine M, is the language M accepts regular? Is it context-free? Is it Turing- decidable?
- Does a particular line (transition) in a program (machine) get executed?
- Does a program contain a computer virus?