5CCS2FC2: Foundations of Computing II

The Limits of Computation

Week 10

Dr Christopher Hampson

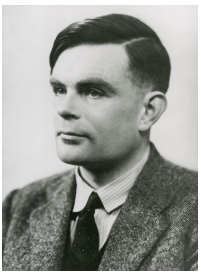
Department of Informatics

King's College London

Decidable and Undecidable Languages

Theorem (The Church-Turing Thesis) Any language that can be effectively computed by some finite process can be recognised by a Turing machine.

Theorem (The Church-Turing Thesis) Any language that can be effectively computed by some finite process can be recognised by a Turing machine.



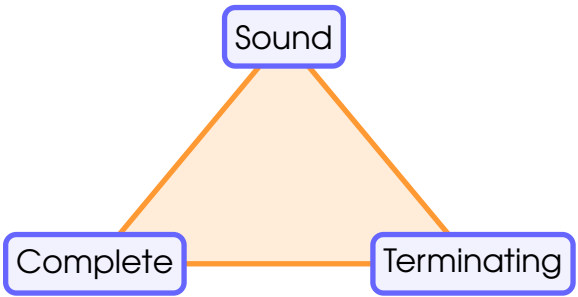
- **Some Notation**

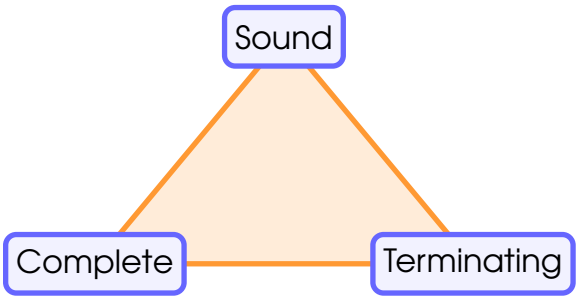
$$M(w) := \begin{cases} 1 & \text{if } M \text{ accepts } w \\ 0 & \text{if } M \text{ rejects } w \\ \uparrow & \text{if } M \text{ does not halt on } w \end{cases}$$

Decidability and Undecidability

- **Decidable Languages**
  - A language L is said to be **decidable** if there is *some* machine M that is:
    - **Sound** — If $M(w) = 1$ then $w \in L$
    - **Complete** — If $w \in L$ then $M(w) = 1$
    - **Terminating** — $T_M(w) < \infty$ is finite for all $w \in \Sigma^*$

Decidability and Undecidability





Is Every Language Decidable?

Decidability and Undecidability

Universal Turing Machines

Universal Turing Machines

- **Encoding tuples as strings**
  - We can encode **tuples** $(w_0, w_1, \ldots, w_n)$ of (non-empty) strings over an expanded alphabet $\Sigma \cup \{\#\}$

$$\langle w_0, w_1, \ldots, w_n \rangle \quad := \quad \# \, w_0 \, \# \, w_1 \, \# \, \ldots \, \# \, w_n \, \#$$

Another possible encodings might include **transcribing** each character using a binary code such as

$0 \mapsto 00 \qquad 1 \mapsto 01 \qquad \# \mapsto 11$

Universal Turing Machines

- **Encoding (finite) functions as strings**
  - We can encode **finite function** $f : A \to B$, where A and B are finite sets as a tuple of tuples

$$\langle f \rangle \quad := \quad \langle\, \langle a_1, f(a_1) \rangle, \langle a_2, f(a_2) \rangle, \ldots, \langle a_n, f(a_n) \rangle \,\rangle$$

(effectively encoding f as a $2 \times n$ matrix)

Universal Turing Machines

- **Encoding Turing Machines as strings**
  - We can even encode **Turing Machines** as strings over an expanded alphabet:

$$\mathbf{code}(M) \quad := \quad \langle q_{\mathbf{init}}, q_{\mathbf{accept}}, q_{\mathbf{reject}}, \langle \delta \rangle \rangle$$

Universal Turing Machines

- **Universal Turing Machine**
  - A **Universal Turing Machine** is a Turing Machine $\mathcal{M}_u$ that takes as input a pair $\langle \mathbf{code}(M), w \rangle$, with the property that

$$\begin{aligned} \mathcal{M}_u \text{ accepts } \langle \mathbf{code}(M), w \rangle &\iff M \text{ accepts } w \\ \mathcal{M}_u \text{ rejects } \langle \mathbf{code}(M), w \rangle &\iff M \text{ rejects } w \\ \mathcal{M}_u \text{ halts on } \langle \mathbf{code}(M), w \rangle &\iff M \text{ halts on } w \end{aligned}$$

Universal Turing Machines

- **Universal Turing Machine**
  - A **Universal Turing Machine** is a Turing Machine $\mathcal{M}_u$ that takes as input a pair $\langle \mathbf{code}(M), w \rangle$, with the property that

$$\mathcal{M}_u \left(\langle \mathbf{code}(M), w \rangle \right) = M(w)$$

A Universal Turing machine acts like a **compiler** or an **interpreter** that reads the *software* $\mathbf{code}(M)$, and simulates the operation of M on the input w. It will accept if M accepts, reject if M rejects and get struck if M gets stuck.

Universal Turing Machines

A Universal Turing machine acts like a **compiler** or an **interpreter** that reads the *software* $\mathbf{code}(M)$, and simulates the operation of M on the input w. It will accept if M accepts, reject if M rejects and get struck if M gets stuck.

End of Slides!