# The Church-Turing Thesis

- We have introduced various abstract computation models, focusing on language recognition and decidability.

- It is commonly believed that all reasonable computational models can be simulated by Turing machines.

- **Church-Turing Thesis**

  > *The intuitive notion of an algorithm corresponds to computability by a Turing machine.*

- Specifically, a language decided in any reasonable computation model is decided by a Turing machine.

- The Church-Turing thesis is not a mathematical theorem, but provides a precise mathematical definition of algorithmic computation, and hence a framework for the formal study of computability.

- We will show that certain problems are not algorithmically solvable in this sense.

# Universal Turing Machines

- We have seen many examples of Turing machines that were designed to solve specific problems and in that sense represent algorithms.

- It is also possible to design general-purpose Turing machines that can simulate any other Turing machine.

- Let $\langle M \rangle$ denote the *encoding* of a Turing machine $M$, $\langle w \rangle$ the encoding of a string $w$, $\langle M, w \rangle$ the encoding of $M$ and $w$, etc.

- A *universal Turing machine* $U$, for input $\langle M, w \rangle$, where $M$ is a Turing machine and $w$ a string, simulates the computation of $M$ for input $w$.

- A universal Turing machine can be designed with three tapes, where
  - The first tape contains the encoding of $M$,
  - The second tape contains the encoding of the current tape contents of $M$,
  - The third tape contains the encoding of the current state of $M$.

# Decidable Problems

- The following problems about finite automata are decidable:
  - **Acceptance problem for DFA's:**
    Given a DFA $M$ and a string $w$, is $w$ accepted by $M$?
  - **Acceptance problem for NFA's:**
    Given an NFA $M$ and a string $w$, is $w$ accepted by $M$?
  - **Regular expressions:**
    Given a regular expression $R$ and a string $w$, is $w$ generated by $R$?
  - **Emptiness problem for DFA's:**
    Given a DFA $M$, is $L(M) = \emptyset$?
  - **Equivalence problem for DFA's:**
    Do two given deterministic finite automata $M_1$ and a $M_2$ accept the same language, i.e., is $L(M_1) = L(M_2)$?

# Decidable Problems - Grammars

- The following problems about context-free grammars and pushdown automata are decidable:
  - **Acceptance problem for CFG's:**
    Given a context-free grammar $G$ and a string $w$, can $w$ be generated by $G$?
  - **Acceptance problem for PDA's:**
    Given a pushdown automaton $M$ and a string $w$, is $w$ accepted by $M$?
  - **Emptiness problem:**
    Given a context-free grammar $G$, is $L(G) = \emptyset$?
  - Any context-free language $L$.