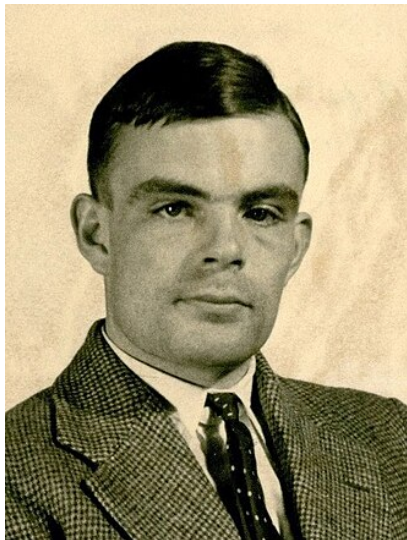


The Church-Turing thesis and Turing machines



Textbook: Chapter 3

History

Via Wikipedia

- ▶ Alonzo Church was Alan Turing's PhD advisor
 - ▶ Church formalized mathematics via lambda calculus
 - ▶ Turing then proved lambda calculus was equivalent to Turing machines, effectively founding computer science
- ▶ Turing then worked as a codebreaker during WW2 at Bletchley Park in England, being a large reason why Germany didn't win
- ▶ He was later persecuted by the English government (that he kept from being overthrown by Nazis) for being gay
 - ▶ They chemically castrated and fired him
- ▶ He died of cyanide poisoning at 41
 - ▶ Speculated to be a suicide
- ▶ The English government finally pardoned him in **2009**, 55 years after he died

Turing machines (informal)

- ▶ DFA have finite memory, PDA have restricted infinite memory: What if we have unrestricted infinite memory?
- ▶ Equivalently, what if a DFA could move around freely and write on an infinite input “tape”?
- ▶ A **Turing machine (TM)** satisfies these conditions
- ▶ They have an infinite read/write tape controlled by a finite set of states and transitions
- ▶ They can move left or right
- ▶ Some states are special and immediately halt operation with either a positive (accept) or negative (reject) result

Criteria	DFA/NFA	TM
Tape usage	Read-only	Read/write
Movement	Leftwards	Bidirectional
Input size	Finite	Infinite to the right
Halting	At input end	Only when entering special states

Church-Turing thesis

- ▶ *Hypothesizes* (does not prove or disprove) that no models are more powerful than Turing machines
 - ▶ “Anything that can be computed can be computed on a Turing machine”
- ▶ This holds for all models of computation that we’ve come up with so far

Turing machines (formal)

Def: Turing machines. A TM is a 7-tuple

$$(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

where Q, Σ, Γ are all finite sets and:

1. Q is the state set
2. Σ is the input alphabet **not containing the blank symbol** \sqcup
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$
4. $\delta : (Q \times \Gamma) \rightarrow (Q \times \Gamma \times \{L, R\})$ is the transition function
5. $q_0 \in Q$ is the start state
6. $q_{\text{accept}} \in Q$ is the accept state
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{reject}} \neq q_{\text{accept}}$

Where L, R mean to move the R/W head left and right one cell respectively.

TM configurations

- ▶ An automaton's **configuration** is a representation of all the information which can change
- ▶ A DFA configuration is an element of Q , while a PDA needs that and the contents of its stack

Def: A TM configuration contains the current state, the current tape contents, and the current head location. By convention, we write uqv for state q with tape contents uv where the R/W head is on the first cell of v .

Ex: The TM configuration $1011q_70111$ means that the tape contents are 1011 to the left of the R/W head, 0 under it, and 111 to its right. The machine is in state q_7 .

TM computation

- ▶ A TM receives its input on the leftmost cells of the tape
- ▶ All cells after the input contain the empty symbol \sqcup
- ▶ A TM **cannot go past the left end of the tape**. If it tries to, it stays at the same cell
- ▶ We say configuration C_1 **yields** C_2 iff the TM can legally go from C_1 to C_2 in a single move
- ▶ $\delta(q_i, b) = (q_j, c, L)$:
 - ▶ If in state q_i with b under the R/W head, move to state q_j , write c , and move one cell to the left
 - ▶ If already on left edge of tape, do all that without moving
- ▶ $\delta(q_i, b) = (q_j, c, R)$:
 - ▶ If in state q_i with b under the R/W head, move to state q_j , write c , and move one cell to the right

Configurations

Def: The **start configuration** of M on w is q_0w . A **rejecting configuration** is any of the form $uq_{\text{reject}}v\$$, and an **accepting configuration** is of the form $uq_{\text{accept}}v\$$. Rejecting and accepting configurations are **halting configurations** and never yield anything.

Def: Computation. A TM M **accepts** a string w iff a series of configurations C_1, C_2, \dots, C_k exists such that:

1. $C_1 = q_0w$ (the start configuration of M on w)
2. C_i yields C_{i+1} for all $1 \leq i < k$
3. C_k is an accepting configuration

Similar definitions hold for rejection and halting. The **language** of M , denoted $\mathcal{L}(M)$ is the set of all strings accepted by M .

Universal TMs

Def: A universal Turing machine (UTM) U is a TM which takes in a 2-tuple (M, w) (where M is some encoding of a TM and w is a string on M 's input alphabet) and simulates the operation of M on w . U is such that:

1. $U(M, w)$ halts iff $M(w)$ halts
2. $U(M, w)$ accepts iff $M(w)$ accepts
3. $U(M, w)$ rejects when $M(w)$ rejects or (by convention) when M is not an encoding of a valid TM

Turing completeness and equivalence

Via Wikipedia

- ▶ By definition, a UTM is a TM which can simulate any TM
- ▶ Thus, if a system Q can simulate a UTM, it can simulate every TM

Def: A system which can simulate a UTM is said to be **Turing complete** (or computationally universal).

Def: A system which can simulate *and be simulated by* a UTM is said to be **Turing equivalent**.

Corollary: Simulating any Turing complete system is sufficient to be Turing complete.

Turing completeness and equivalence cont.

Corollary: All Turing complete programming languages are computationally equivalent: They only differ in speed and the “coolness factor”.

Corollary: The Church-Turing thesis can be restated “all Turing complete systems are Turing equivalent.”

Note: In most cases the restrictions of finite memory are ignored. If this is done, all known physically-constructable systems are Turing equivalent. If not, none are even Turing complete.

Turing recognizability

Def: A language L is **Turing-recognizable** iff some TM M recognizes it ($\mathcal{L}(M) = L$).

- ▶ This handles *accept* states, but what about reject states?
- ▶ **What if the machine enters an infinite loop?**

Turing decidability

Def: A **decider** on language L is a TM which never enters an infinite loop: It only ever accepts or rejects. This TM is said to **decide** L .

Def: A language is **Turing-decidable** or simply **decidable** iff some Turing machine **decides** it.

Corollary: All Turing-decidable languages are Turing-recognizable.

Next time: The diagonalization proof

Unintentional Turing completeness [\[edit \]](#)

Some [software](#) and [video games](#) are Turing-complete by accident, i.e. not by design.

Software:

- [Microsoft Excel](#)^[17]

Games:

- [Dwarf Fortress](#)^[18]
- [Cities: Skylines](#)^[19]
- [Opus Magnum](#)^[20]
- [Minecraft](#)^[21]
- [Magic: The Gathering](#)^{[22][23]}
- Infinite-grid [Minesweeper](#)^[24]

Social media:

- [Habbo Hotel](#)^[25]

Computational languages:

- [C++ templates](#)^[26]
- [printf](#) format string^[27]
- [TypeScript](#)'s type system^[28]
- [x86 assembly](#)'s MOV instruction^{[29][30][31]}

Biology:

- [Chemical reaction networks](#)^{[32][33][34][35]} and [enzyme-based DNA computers](#)^[36] have been shown to be Turing-equivalent