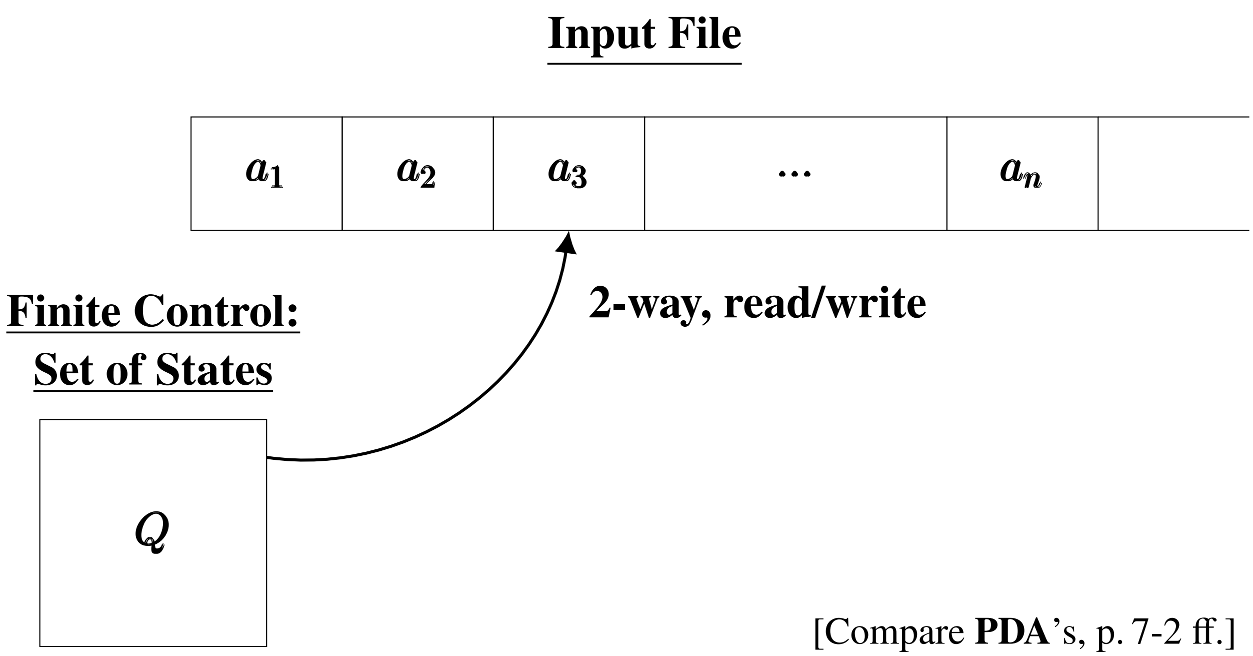
9 Turing Machines

[Linz Sec. 9, Kozen L. 28]

Most powerful automaton so far!

Can compute any function considered to be computable by any (deterministic) algorithm.

Informal description of a **Turing Machine (TM)** M:



[Compare **PDA**'s, p. 7-2 ff.]

Infinite 1-way tape. At start:

Input: finite word on left end. ("Blank" cells elsewhere)

Starts in state s, reading leftmost cell.

At each move: M **reads** cell

then, according to the **state** and **transition function**:

- **writes** new symbol in cell,
- **moves** left or right,
- enters **new state**.

Eventually (maybe!) **halts**

Output: word on tape.

Then we say M **computes** the **partial function**:

$$f : \Sigma^* \rightharpoonup \Sigma^*$$

Also, if $\Sigma = \{0, 1\}$, this gives

$$f : \mathbb{N} \rightharpoonup \mathbb{N}$$

(since binary strings code natural numbers).

We also say: f is $T-$**computable** (by M).

There are many **models of computation**:

- Turing Machines (Turing, 1936)
- λ-calculus (Church, 1933)
- High-Level Programming Languages (Java, C, Pascal,...)

They all embody the idea of **algorithm** and **effective computation**.

All have been shown to be **equivalent**!

Church's Thesis: A function computable by **any** algorithm is computable in λ-calculus

Turing's Thesis: A function computable by **any** algorithm is computable by a Turing Machine

Church-Turing (CT) Thesis

A function computable by **any** algorithm is computable by the **λ-calculus OR** a **Turing Machine OR** a **C program OR** ...

Note:

Although we can **prove** the **equivalence** of
TM-computability, **λ-computability**, **C-computability**, etc.,
we **cannot prove** the **CT Thesis**,
since the concept of algorithm is **not mathematically definable**.

However, there are strong arguments in its favour:

(1) Turing gives **good philosophical arguments** in its favour in his 1936 article.

(2) It is **robust**: many proposed models of computability have been shown to be equivalent.

(3) In the $\sim$ 80 years since its formulation,
no convincing counterexample has been discovered.

Q. What does it mean for a TM to **accept** a language, $L \subseteq \Sigma^*$?

The simplest approach:

Consider the **characteristic function** of L on Σ^*:

$$\chi_L : \Sigma^* \longrightarrow \mathbb{B}$$

where

$$\chi_L(w) = \begin{cases} 1 & \textbf{if } w \in L \\ 0 & \textbf{otherwise} \end{cases}$$

(We assume that $\mathbb{B} \subseteq \Sigma$ contains 2 special symbols:
0 and **1**, for **"false"** and **"true"**.)

Then we say: the TM M **accepts** L iff it computes χ_L.

We also say: L is **effectively decidable** if there is an algorithm to **decide membership** of L.

Then, by the **CT Thesis**:

$$L \textbf{ is effectively decidable} \iff \chi_L \textbf{ is TM-computable.}$$

Two References for Computability Theory:

(1) M. Davis: *The Universal Computer.* Norton, 2000

(2) M. Davis (ed.): *The Undecidable.* Raven Press, 1965.

Ref. (1) is a very readable history of the subject.

Ref. (2) contains classic papers by the pioneers in the field: Turing, Church, Gödel, and others.