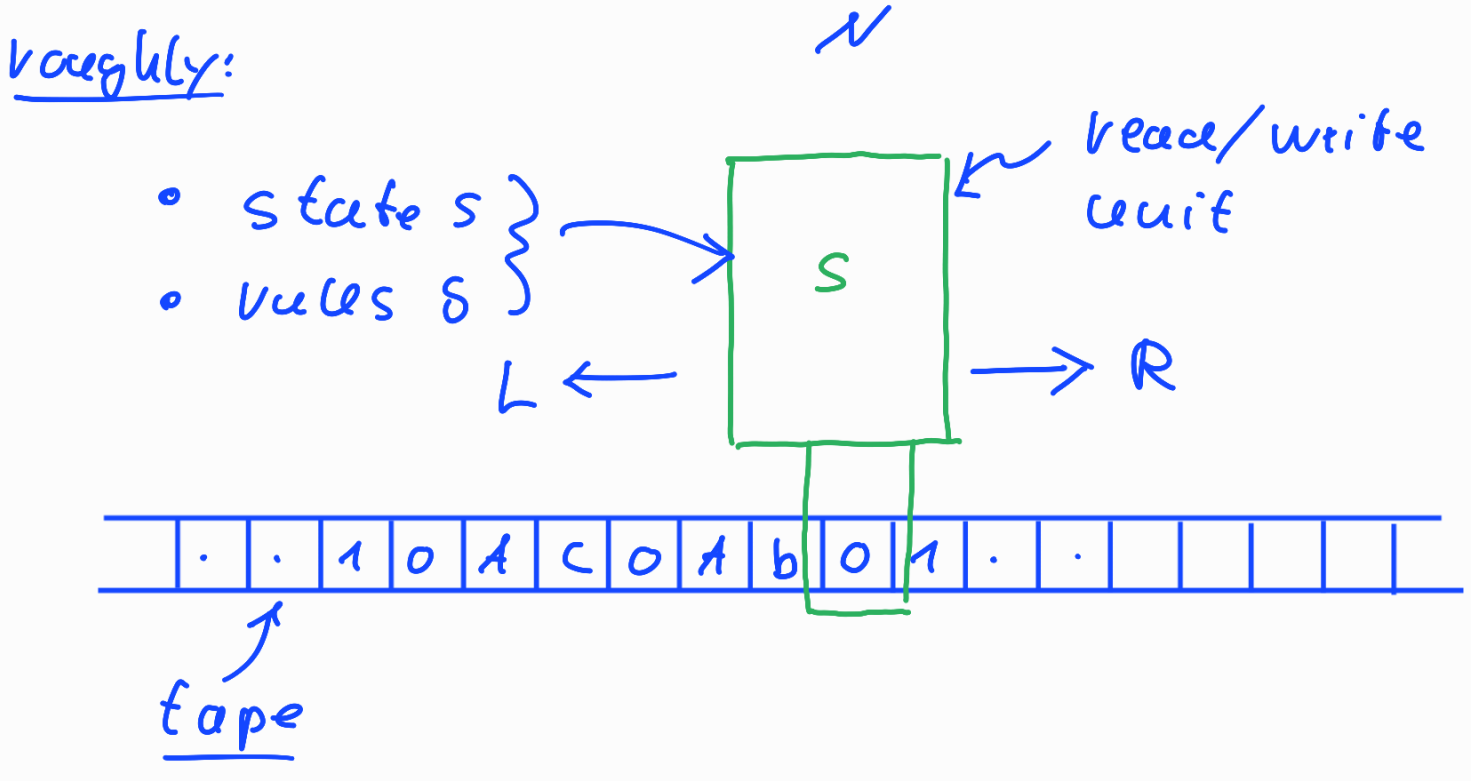
Computability and Complexity

Turing machine as model for (classical) computation

2 →
- computability
- complexity

Turing machine = mathematical abstraction of "paper, pen & brain";

roughly:



formal definition:

Turing machine = 7-tuple

$$M = (S, \Sigma, \Gamma, \delta, s_0, \square, E)$$

S : finite set of internal states

Σ : input alphabet (often $\Sigma = \{0,1\}$)

Γ : working alphabet (often $\Gamma = \{0,1,\square\}$)

s_0 : initial state

$\square$: "blank"

E : set of final states

δ : transition rules = map:

$$S \setminus E \times \Gamma \longrightarrow S \times \Gamma \times \{L, R, N\}$$

$$(s, \alpha) \longmapsto (s', \alpha', m)$$

m : movement

Notations

- Σ^* = set of all strings over alphabet Σ
- <u>configuration</u> h of a T.M. :

$$h \in \Gamma^* \underline{S} \Gamma^*$$

e.g. : $h = \underbrace{a_1 a_2 \cdots a_m}\ \underline{S}\ \underbrace{b_1 b_2 \cdots b_n}$

- <u>configuration up-date</u> :

$a_1 \cdots a_m\ S\ b_1 b_2 \cdots b_n$

$S(S, b_n) =$

$$\vdash \begin{cases} a_1 \cdots a_m\ \underline{S'}\ c\ b_2 \cdots b_n & : (S', c, \underline{N}) \\ a_1 \cdots a_m\ c\ \underline{S'}\ b_2 \cdots b_n & : (S', c, \underline{R}) \\ a_1 \cdots \underline{S'}\ a_m\ c\ b_2 \cdots b_n & : (S', c, \underline{L}) \end{cases}$$

Example:

$M_0 = (\{s_0, s_1, s_e\}, \{0,1\}, \{0,1,\square\}, \delta, s_0, \square, \{s_e\})$

δ :

$s_0, 0 \to s_0, 0, R$

$s_0, 1 \to s_0, 1, R$

$s_0, \square \to s_1, 0, L$

$s_1, 0 \to s_1, 0, L$

$s_1, 1 \to s_1, 1, L$

$s_1, \square \to s_e, \square, R$

works on input $x = 101$ as follows (blanks ($\square$) written only when needed):

$h_0 = s_0 101 \vdash 1 s_0 01 \vdash 10 s_0 1$

$\vdash 101 s_0 \square \vdash 10 s_1 10$

$\vdash 1 s_1 010 \vdash s_1 1010$

$\vdash s_1 \square 1010 \vdash s_e 1010$

Notation:

- $h, h' \in \Gamma^* \, s \, \Gamma^*$:

$h \vdash h' \quad :\Leftrightarrow \quad \exists \; h_1, \ldots, h_u \in \Gamma^* \, s \, \Gamma^*$

s.t. $h_1 = h \;, \; h_u = h'$

and $h_i \vdash h_{i+1}$

for M_0 as in example:

$$s_0 101 \vdash^* s_e 1010$$

for general binary x:

$$s_0 x \vdash^* s_e 2x$$

$\rightarrow$ M_0 „computes" $f(x) = 2x$!

Def: Turing-computability

function $f: \mathbb{N} \to \mathbb{N}$ Turing-computable

$:\Leftrightarrow$ $\exists$ T.M. M s.t.

$\forall x, y \in \mathbb{N}$:

$f(x) = y \Leftrightarrow s_0 x \vdash^* s_e y$

- $f(x) = 2x$ Turing-computable ✓.
- fact:

  f Turing-computable

  (1)

  $\Leftrightarrow$ f computable on any computer using any programming language

  (2)

  $\Leftrightarrow$ f computable on a quantum computer

(1): „$\Leftarrow$": ✓

„$\Rightarrow$": cf. comp. science

(2): „$\Rightarrow$": universal classical gates (e.g. NAND, BRANCH) can be simulated by e.g. quantum Toffoli-gate ✓

„$\Leftarrow$" simulate time-evolution of quantum-circuit on classical computer; this may require exponentially large memory and „ " running time, but this is irrelevant for the question of computability!

→

Church-Turing-Hypothesis:

Any physical model of computation is equivalent to a Turing-machine! ✓

Strong Church-Turing-Hypothesis:

Any physical model of computation is polynomially equivalent to a T.M.! ?

↑ probably wrong, as quantum algorithms by Simon and Shor demonstrate exponential speed-up!

Are there any uncomputable functions ?

Lots of them ! as seen by the following counting argument :

- set of functions $\mathbb{N} \to \mathbb{N}$ $\quad \mathcal{F} = \mathbb{N}^{\mathbb{N}}$ as least as big as power set $\wp(\mathbb{N})$ of $\mathbb{N}$ (= set of all sub-sets of $\mathbb{N}$)

⌜ because of injection

$$\wp(\mathbb{N}) \longrightarrow \mathcal{F}$$

$$A \longmapsto \chi_A : x \mapsto \begin{cases} 1 : x \in A \\ 0 : x \notin A \end{cases}$$

⌟

since $\wp(\mathbb{N})$ is uncountable infinite, there are uncountable infinite functions

$\mathbb{N} \to \mathbb{N}$.

- set of all Turing machines (TM) is countable infinite, because there is a surjection

$$\mathbb{N} \longrightarrow TM$$
$$w \longmapsto M_w$$

(„all Turing machines can be enumerated!")

→ there are uncountably many functions $\mathbb{N} \to \mathbb{N}$ that can't be computed!

explicite ones?

→

$$f(n) := \begin{cases} 1 : & n \text{ appears as consecutive} \\ & \text{digits in the digital ex-} \\ & \text{pansion of } \pi, \\ 0 : & \text{if it doesn't} \end{cases}$$

might be an incomputable function!

- a provable incomputable function is Turing's Halting-function:

Def.:

- T.M. M holds on input x

$$:\Leftrightarrow \exists\, q_e \in T^* \, E \, T^* : \quad s_0 x \vdash^* q_e$$

- enumeration of all Turing machines:

$$\mathbb{N} \longrightarrow \mathcal{TM}$$

$$w \longmapsto M_w$$

$\longrightarrow$

Halting-function

$h : \mathbb{N} \to \mathbb{N}$

$w \mapsto h(w) = \begin{cases} 1 : M_w \text{ on input } w \text{ holds} \\ 0 : M_w \text{ on input } w \text{ does not hold} \end{cases}$

Thm:

h is not Turing-computable !

proof by contradiction:

assume h is computable

$\to \exists$ T.M. M that computes h

modify M to another T.M. M' with the following properties:

(i) M' holds on input w if M on input w holds with output 0 (i.e. $h(w) = 0$)

(ii) M' on input w does not hold if

M on input w outputs 1

(i.e. $h(w) = 1$)

(explicit construction of M' based on existence of M in problem 15)

- let $w' \in \mathcal{W}$ s.t. $M' \doteq M_{w'}$,

then: M' holds on input w'

$\Leftrightarrow$ (construction of M') M on input w' holds in 0

$\Leftrightarrow$ (M computes h) $h(w') = 0$

$\Leftrightarrow$ (Def. of h) $M_{w'}$ does not hold on input w'

$\Leftrightarrow$ ($M_{w'} = M'$) M' does not hold on input w'