What is a computable function ?

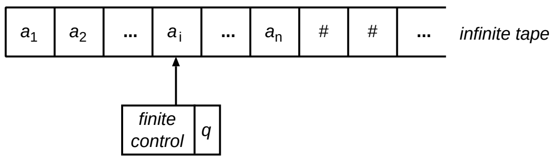
- Non-trivial question $\rightsquigarrow$ various formalizations, e.g.
  - General recursive functions *Gödel/Herbrand/Kleene 1936*
  - λ-calculus *Church 1936*
  - μ-recursive functions *Gödel/Kleene 1936*
  - Turing machines *Turing 1936*
  - Post systems *Post 1943*
  - Markov algorithms *Markov 1951*
  - Unlimited register machines *Shepherdson-Sturgis 1963*

  ...
- All these approaches have turned out to be equivalent.

Church-Turing thesis

The class of intuitively computable functions is equal to the class of Turing computable functions.

Turing machine



Depending on the symbol scanned and the state of the control, in each step the machine

- changes state,
- prints a symbol on the cell scanned, replacing what is written there,
- moves the head left or right one cell.

Formal definition

- $M = (Q, \Sigma, \Gamma, \delta, q_0, \#, F)$
- Q is the finite set of *states.*
- Γ is the finite alphabet of allowable *tape symbols.*
- $\# \in \Gamma$ is the *blank.*
- $\Sigma \subset \Gamma \setminus \{\#\}$ is the set of *input symbols.*
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the *next move function* (possibly undefined for some arguments)
- $q_0 \in Q$ is the *start state.*
- $F \subseteq Q$ is the set of *final (accepting) states.*

Recognizing languages

- *Instantaneous description:* $\alpha_l\, q\, \alpha_r$, where
  - q is the current state,
  - $\alpha_l \alpha_r \in \Gamma^*$ is the string on the tape up to the rightmost nonblank symbol,
  - the head is scanning the leftmost symbol of α_r.
- *Move:* $\alpha_l\, q\, \alpha_r \vdash \alpha_l'\, q'\, \alpha_r'$, by one step of the machine.
- *Language accepted*

$$L(M) = \{w \in \Sigma^* \mid q_0 w \vdash^* \alpha_l q \alpha_r, \text{ for some } q \in F \text{ and } \alpha_l, \alpha_r \in \Gamma^*\}$$

- M may not halt, if w is not accepted.

Example

- *Turing machine*

$$M = (\{q_0, \ldots, q_4\}, \{0,1\}, \{0,1,X,Y,\#\}, \delta, q_0, \#, \{q_4\})$$

  accepting the language $L = \{0^n 1^n \mid n \geq 1\}$

δ	0	1	X	Y	#
q_0	(q_1, X, R)	–	–	(q_3, Y, R)	–
q_1	$(q_1, 0, R)$	(q_2, Y, L)	–	(q_1, Y, R)	–
q_2	$(q_2, 0, L)$	–	(q_0, X, R)	(q_2, Y, L)	–
q_3	–	–	–	(q_3, Y, R)	$(q_4, \#, R)$
q_4	–	–	–	–	–

- *Example computation*

$$\begin{array}{llllllll}
q_0 0011 & \vdash & X q_1 011 & \vdash & X0 q_1 11 & \vdash & X q_2 0Y1 & \vdash \\
q_2 X0Y1 & \vdash & X q_0 0Y1 & \vdash & XX q_1 Y1 & \vdash & XXY q_1 1 & \vdash \\
XX q_2 YY & \vdash & X q_2 XYY & \vdash & XX q_0 YY & \vdash & XXY q_3 Y & \vdash \\
XXYY q_3 & \vdash & XXYY\# q_4 & & & & &
\end{array}$$

Recursive languages

- A language $L \subseteq \Sigma^*$ is *recursively enumerable* if $L = L(M)$, for some Turing machine M.

$$w \longrightarrow \boxed{\text{M}} \longrightarrow \begin{cases} \text{yes,} & \text{if } w \in L \\ \text{no,} & \text{if } w \notin L \\ M \text{ does not halt,} & \text{if } w \notin L \end{cases}$$

- A language $L \subseteq \Sigma^*$ is *recursive* if $L = L(M)$ for some Turing machine M that halts on all inputs $w \in \Sigma^*$.

$$w \longrightarrow \boxed{\text{M}} \longrightarrow \begin{cases} \text{yes,} & \text{if } w \in L \\ \text{no,} & \text{if } w \notin L \end{cases}$$

- **Lemma.** L is recursive iff both L and $\overline{L} = \Sigma^* \setminus L$ are recursively enumerable.

Enumerating languages

- An *enumerator* is a Turing machine M with extra output tape T, where symbols, once written, are never changed.
- M writes to T words from Σ^*, separated by \$.
- Let $G(M) = \{w \in \Sigma^* \mid w \text{ is written to } T\}$.

Some results

- **Lemma.** For any finite alphabet Σ, there exists a Turing machine that generates the words $w \in \Sigma^*$ in *canonical ordering* (i.e., $w \prec w' \Leftrightarrow |w| < |w|$ or $|w| = |w|$ and $w \prec_{lex} w'$).
- **Lemma.** There exists a Turing machine that generates all pairs of natural numbers (in binary encoding).

  Proof: Use the ordering $(0,0)$, $(1,0)$, $(0,1)$, $(2,0)$, $(1,1)$, $(0,2)$, ...
- **Proposition.** L is recursively enumerable iff $L = G(M)$, for some Turing machine M.

Computing functions

- Unary encoding of natural numbers: $i \in \mathbb{N} \mapsto \underbrace{||...|}_{i \text{ times}} = |^i$

  (binary encoding would also be possible)
- *M computes $f : \mathbb{N}^k \to \mathbb{N}$* with $f(i_1, ..., i_k) = m$:
  - Start: $|^{i_1} 0 |^{i_2} 0 ... |^{i_k}$
  - End: $|^m$
- *f partially recursive:*

$$i_1, ..., i_k \longrightarrow \boxed{\text{M}} \longrightarrow \begin{cases} \text{halts with } f(i_1, ..., i_k) = m, \\ \text{does not halt, i.e., } f \text{ undefined.} \end{cases}$$

- *f recursive:*

$$i_1, ..., i_k \longrightarrow \boxed{\text{M}} \longrightarrow \text{halts with } f(i_1, ..., i_k) = m.$$

Turing machines codes

- May assume

$$M = (Q, \{0,1\}, \{0,1,\#\}, \delta, q_1, \#, \{q_2\})$$

- Unary encoding

$$0 \mapsto 0, 1 \mapsto 00, \# \mapsto 000, L \mapsto 0, R \mapsto 00$$

- $\delta(q_i, X) = (q_j, Y, R)$ encoded by

$$0^i 1 \underbrace{0 \ldots 0}_{X} 1 0^j 1 \underbrace{0 \ldots 0}_{Y} 1 \underbrace{0 \ldots 0}_{R}$$

- δ encoded by

$$111\, \mathtt{code}_1\, 11\, \mathtt{code}_2\, 11 \ldots 11\, \mathtt{code}_r\, 111$$

- Encoding of Turing machine M denoted by $\langle M \rangle$.

Numbering of Turing machines

- **Lemma.** There exists a Turing machine that generates the natural numbers in binary encoding.
- **Lemma.** There exists a Turing machine *Gen* that generates the binary encodings of all Turing machines.
- **Proposition.** The language of Turing machine codes is recursive.
- **Corollary.** There exist a bijection between the set of natural numbers, Turing machine codes and Turing machines.

$$\begin{array}{ccccc} & \boxed{\text{Gen}} & \longrightarrow & \multirow{2}{*}{$\boxed{\begin{array}{c}\text{Equality test}\\ \text{+ counter}\end{array}}$} & \multirow{2}{*}{$\longrightarrow \text{number } n$} \\ M \longrightarrow & \langle M \rangle & \longrightarrow & & \end{array}$$

Diagonalization

- Let w_i be the i-th word in $\{0,1\}^*$ and M_j the j-th Turing machine.
- Table T with $t_{ij} = \begin{cases} 1, & \text{if } w_i \in L(M_j) \\ 0, & \text{if } w_i \notin L(M_j) \end{cases}$

$$\begin{array}{cc|ccccc} & & & j \longrightarrow & & & \\ & & 1 & 2 & 3 & 4 & \ldots \\ \hline & 1 & 0 & 1 & 1 & 0 & \ldots \\ i & 2 & 1 & 1 & 0 & 1 & \ldots \\ \downarrow & 3 & 0 & 0 & 1 & 0 & \ldots \\ & \vdots & \vdots & \vdots & \vdots & \vdots & \end{array}$$

- *Diagonal language* $L_d = \{w_i \in \{0,1\}^* \mid w_i \notin L(M_i)\}$.
- **Theorem.** L_d is not recursively enumerable.
- *Proof:* Suppose $L_d = L(M_k)$, for some $k \in \mathbb{N}$. Then

$$w_k \in L_d \Leftrightarrow w_k \notin L(M_k),$$

contradicting $L_d = L(M_k)$.