Computability and Complexity Theory

Computability and complexity

- *Computability theory*
  - What is an algorithm ?
  - What problems can be solved on a computer ?
  - What is a computable function ?
  - Solvable vs. unsolvable problems (decidability)
- *Complexity theory*
  - How much time and memory is needed to solve a problem ?
  - Tractable vs. intractable problems

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

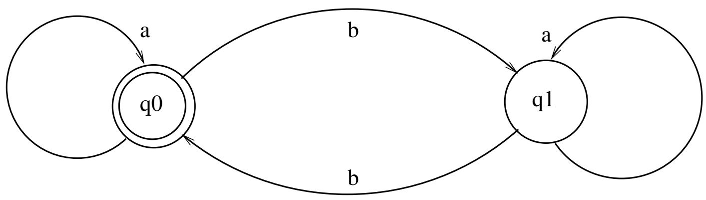
The class of intuitively computable functions is equal to the class of Turing computable functions.

Finite automata

Finite automaton: $M = (Q, \Sigma, \delta, q_0, F)$ with

- Q finite set of *states*
- Σ finite *input alphabet*
- $\delta : Q \times \Sigma \to Q$ *transition function*
- $q_0 \in Q$ *initial state*
- $F \subseteq Q$ set of *final states*

Example



$M^0 = (Q, \Sigma, \delta, q_0, F)$ with

- $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$
- $\delta(q_0, a) = q_0$, $\delta(q_0, b) = q_1$, $\delta(q_1, a) = q_1$, $\delta(q_1, b) = q_0$

Recognizing languages

- Denote by Σ^* the set of finite words (strings) over Σ, by $\varepsilon \in \Sigma^*$ the empty word.
- Define $\overline{\delta} : Q \times \Sigma^* \to Q$ by

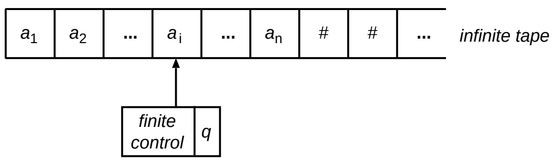
$$\begin{aligned} \overline{\delta}(q, \varepsilon) &= q \quad \text{and} \\ \overline{\delta}(q, wa) &= \delta(\overline{\delta}(q, w), a), \text{ for all } w \in \Sigma^*, a \in \Sigma. \end{aligned}$$

- *Language accepted by M:*

$$L(M) = \{w \in \Sigma^* \mid \overline{\delta}(q_0, w) = p, \text{ for some } p \in F\}$$

- *Example:* $L(M^0)$ is the set of all strings over $\Sigma = \{a, b\}$ with an even number of b's.
- Gene regulatory networks can be modeled as networks of finite automata.

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 by M*

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

- 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.