Intro to computability

Peter Mayr

Computability Theory, January 15, 2021

What can a computer do in principle?

Hilbert's Tenth Problem (1900)

Given a polynomial $p(x_1, \ldots, x_n)$ with integer coefficients, decide whether it has an integer zero.

Matiyasevich (1970)

No such algorithm exists. Hilbert's Tenth Problem is undecidable.

Other undecidable problems

1. Hilbert's Entscheidungsproblem for first order logic (Church, Turing 1936)
2. Halting Problem for Turing machines
3. Word problem for (semi)groups

What is efficiently computable?

- The computational complexity of an algorithm is usually measured in the time or space (memory) it requires depending on the size of the input.
- This depends on the specific computational model.

Topics of this course

- Models of computation
  - automata, regular languages
  - Turing machines
  - recursive functions
- Undecidability
  - Halting problem
  - Word problem for semigroups
- Degrees of unsolvability
  - Turing reductions
  - arithmetical hierarchy
  - Post's problem for Turing degrees
- Computational complexity
  - time and space complexity
  - P vs NP, NP-completeness
  - L, NL, PSPACE

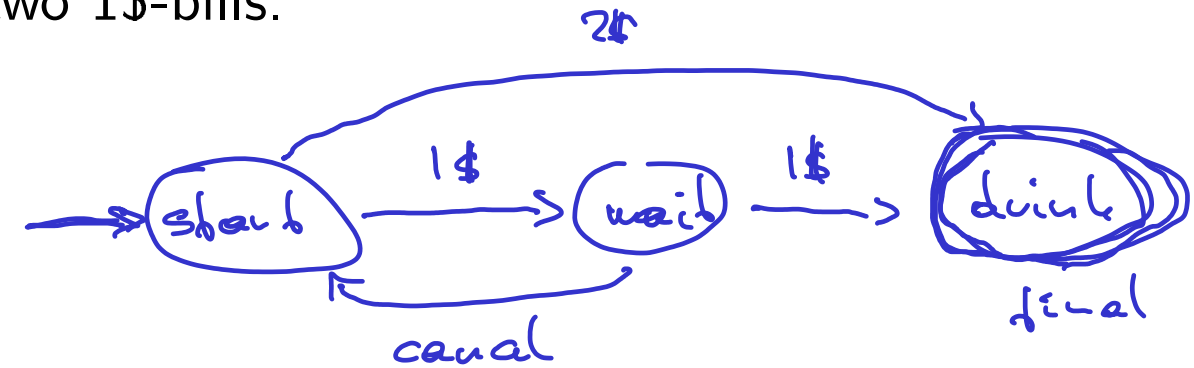
Some classic textbooks we will reference

- Hopcroft, Motwani, Ullman. Introduction to automata theory, languages, and computation. Pearson; 3rd edition, 2006.
- Odifreddi. Classical recursion theory. North-Holland Publishing Co., Amsterdam, 1989.
- Papadimitriou. Computational complexity. Addison-Wesley, 1994.
- Sipser. Introduction to the theory of computation. Thomson Course Technology, Boston, 2nd edition, 2006.
- Soare. Recursively enumerable sets and degrees. Springer-Verlag, Berlin, 1987.

1. Automata and regular languages

Example

Model a vending machine M_1 that delivers a drink for one 2$-coin or two 1$-bills.



Machine accepts inputs

11
2
1C11
1C2
⋮
(1C)*(11 + 2)

arbitrary sequences of 1C followed 11 or 2

δ	1	2	c
s = start	w	d	
w = wait	d		s
d = drink			
ϕ			

Final states $F = \{d\}$

For undefined transition we can add a new state ϕ

Automata

Definition

A **deterministic finite automaton (DFA)** is a 5-tuple $(Q, \Sigma, \delta, s, F)$ with

- Q a finite set (**states**),
- Σ a finite set (**input alphabet**),
- $\delta\colon Q \times \Sigma \to Q$ the **transition function**,
- $s \in Q$ the **start state**,
- $F \subseteq Q$ the set of **final/accepting states**.

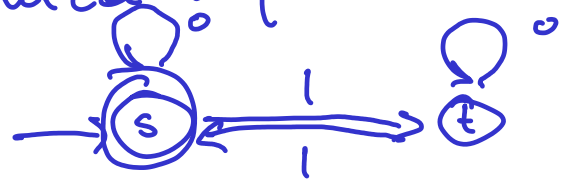
A DFA starts in state s and reads some input string $(a_1, \ldots, a_n)$ for $a_i \in \Sigma$. If it reads a_i in state q_i, it changes to state $\delta(q_i, a_i)$.

Example

$M_2 = (Q, \Sigma, \delta, s, \{s\})$ with $Q = \{s, t\}, \Sigma = \{0, 1\}$.

δ	0	1
s	s	t
t	t	s

graphical representation



Every word with an even number of 1s drives M_2 to a final state.

Languages

Definition

- $\Sigma^* := \bigcup_{n \in \mathbb{N}} \Sigma^n$ is the set of all **words** over Σ.
  $|w|$ is the **length** of a word.
  $\epsilon \in \Sigma^0$ is the **empty word** (length 0).
- uv is the **concatenation** of words u, v.
- $L \subseteq \Sigma^*$ is a **language**.

$\mathbb{N} = \{0, 1, 2, \ldots\}$
$|001| = 3$

Definition

For a DFA $(Q, \Sigma, \delta, s, F)$ the **extended transition function**

$$\delta : Q \times \Sigma^* \to Q$$

is defined inductively for $q \in Q, w \in \Sigma^*, a \in \Sigma$ by

$$\begin{aligned} \delta(q, \epsilon) &:= q \\ \delta(q, wa) &:= \delta(\delta(q, w), a) \end{aligned}$$

Note: δ restricted to Σ is just the original function.

Definition

Let $M = (Q, \Sigma, \delta, s, F)$ be a DFA, let $w \in \Sigma^*$.

- M **accepts** w if $\delta(s, w) \in F$.
- M **rejects** w otherwise.

The **language of** M is

$$L(M) := \{w \in \Sigma^* \ : \ M \text{ accepts } w\}.$$

Example (continued)

$L(M_2) = \{w \in \{0,1\}^* \ : \ w \text{ contains an even number of 1s } \}$