

Complexity Theory

Jan Křetínský

Chair for Foundations of Software Reliability and Theoretical Computer Science
Technical University of Munich
Summer 2016

Based on slides by Jörg Kreiker

Lecture 2

Turing Machines

Agenda

Formalize a **model of computation!**

- k -tape Turing machines
- robustness
- universal Turing machine
- computability, halting problem
- **P**

Which models of computation do you know?

Which models of computation do you know?

- programming languages
- hardware
- biological/chemical systems
- primitive/ μ -recursive functions/ λ -calculus
- logic
- automata
- quantum computers
- paper and pencil

Which models of computation do you know?

- programming languages
- hardware
- biological/chemical systems
- primitive/ μ -recursive functions/ λ -calculus
- logic
- automata
- quantum computers
- paper and pencil

Turing machines!

Which models of computation do you know?

- programming languages
- hardware
- biological/chemical systems
- primitive/ μ -recursive functions/ λ -calculus
- logic
- automata
- quantum computers
- paper and pencil

Turing machines!

Church-Turing Thesis: all models equally expressive

k -tape Turing machines

- k scratchpad tapes, infinitely long, contain cells
 - one input tape, read-only
 - one output tape
 - working tapes
 - k heads positioned on individual cells for reading and writing
 - finite control (finite set of rules)
 - vocabulary, alphabet to write in cells
 - actions: depending on
 - symbols under heads
 - control state
- one can
- move heads (right, left, stay)
 - write symbols into current cells

TMs – reading palindromes

TM for function $pal : \{0, 1\}^* \rightarrow \{0, 1\}$ which outputs 1 for **palindromes**.

TMs – reading palindromes

TM for function $pal : \{0, 1\}^* \rightarrow \{0, 1\}$ which outputs 1 for **palindromes**.

- **copy input** to work tape
- move input head to front, work tape head to end
- in each step
 - **compare** input and work tape
 - move input head **right**
 - move work head **left**
- if whole input processed, output 1

TMs – formally

Definition (*k*-tape Turing machine (syntax))

Turing machine is a triple (Γ, Q, δ) where

- Γ is a **finite alphabet** (tape symbols) comprising 0, 1, \square (empty cell), and \triangleright (start symbol)
- Q is **finite** set of **states** (control) containing q_{start} and q_{halt}
- $\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^{k-1} \times \{l, s, r\}^k$, **transition function** such that $\delta(q_{halt}, \vec{\sigma}) = (q_{halt}, \vec{\sigma}_2, \vec{s})$.

TMs – formally

Definition (Computing a function and running time)

Let M be a k -tape TM and $x \in (\Gamma \setminus \{\square, \triangleright\})^*$ an **input**. Let $T : \mathbb{N} \rightarrow \mathbb{N}$ and $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ be functions.

1. the **start configuration** of M on input x is $\triangleright x \square^\omega$ on the input tape and $\triangleright \square^\omega$ on the $k - 1$ other tapes; all heads are on \triangleright ; and M is in state q_{start}
2. if M is in state q and $(\sigma_1, \dots, \sigma_k)$ are symbols being read, and $\delta(q, (\sigma_1, \dots, \sigma_k)) = (q', (\sigma'_2, \dots, \sigma'_k), \vec{z})$, then **at the next step** M is in state q' , σ_i has been replaced by σ'_i for $i = 2..k$ and the heads have moved **left**, **stayed**, or **right** according to \vec{z}
3. M has **halted** if it gets to state q_{halt}
4. M **computes f in time T** if it halts on input x with $f(x)$ on its output tape and every $x \in \{0, 1\}^*$ requires **at most** $T(|x|)$ steps.

Remarks on TM definition

- TMs are **deterministic**
- going left from \triangleright means **staying**
- item 4: consider **time-constructible** functions T only
 - $T(n) \geq n$ and
 - exists TM M computing T in time T
- TM define **total functions**

Agenda

- k -tape Turing machines ✓
- robustness
- universal Turing machine
- computability, halting problem
- **P**

Robustness

Definition of TM is **robust**, most choices do **not change** complexity classes.

- **alphabet** size (two is enough)
- number of tapes (one is enough)
- tape dimensions (one-directional tapes, bi-directional tapes, two-dimensional tapes)
- **random access** TMs
- **oblivious** TMs
 - see exercises
 - head positions at i -th step of execution on input x depend only on $|x|$ and i

All variations can simulate each other with at most **polynomial overhead** in running time.

Agenda

- k -tape Turing machines ✓
- robustness ✓
- universal Turing machine
- computability, halting problem
- **P**

Universal TM

- TMs can be represented as **strings** (over $\{0, 1\}$) by encoding their transition function (can you?)
 - write M_α for TM **represented** by string α
 - every string α represents **some** TM
 - every TM has **infinitely many** representations
- if TM M computes f , **universal TM U** takes representation α of TM M and input x and computes $f(x)$
- like **general purpose computer** loaded with software
- like **interpreter** for a language written in same language
- U has **bounded** alphabet, rules, tapes; simulates much larger machines **efficiently**

Efficient simulation

Theorem (Universal TM)

There exists a TM U such that for every $x, \alpha \in \{0, 1\}^$, $U(x, \alpha) = M_\alpha(x)$. If M_α holds on x within T steps, then $U(x, \alpha)$ holds within $O(T \log T)$ steps.*

Agenda

- k -tape Turing machines ✓
- robustness ✓
- universal Turing machine ✓
- computability, halting problem
- **P**

Deciding languages

- often one is interested in functions $f : \{0, 1\}^* \rightarrow \{0, 1\}$
- f can be identified with the language $L_f = \{x \in \{0, 1\}^* \mid f(x) = 1\}$
- TM that computes f is said to **decide** L_f (and vice versa)

Halting Problem

There are languages that **cannot be decided** by any TM regardless time and space.

Example

The **halting problem** is the set of pairs of TM representations and inputs, such that the TMs eventually halt on the given input.

$$\text{Halt} = \{\langle \alpha, x \rangle \mid M_\alpha \text{ halts on } x\}$$

Theorem

Halt is not decidable by any TM.

Proof: diagonalization and reduction

DTIME

Definition (DTIME)

Let $T : \mathbb{N} \rightarrow \mathbb{N}$ be a function. $L \subseteq \{0, 1\}^*$ is in $\text{DTIME}(T)$ if there exists a TM deciding L in time T' for $T' \in O(T)$.

- **D** refers to **deterministic**
- **constants** are ignored since TM can be **sped up** by arbitrary constants

P

Definition (P)

$$P = \bigcup_{c \geq 1} \text{DTIME}(n^c)$$

- P captures tractable computations
- low-level choices of TM definitions are immaterial to P
- Connectivity, Primes \in P

What have we learnt?

- many equivalent ways to capture essence of computations (Church-Turing)
- k -tape TMs
- TM can be represented as strings; **universal TM** can simulate any TM given its representations with **polynomial overhead** only
- **uncomputable** functions do exist (halting problem): **diagonalization** and **reductions**
- **P** **robust** wrt. tweaks in TM definition (universal simulation)
- **P** captures **tractable** computations, solvable by TMs in **polynomial time**
- diagonalization, reduction
- up next: **NP**