Theory of Computing

Lecture 14

MAS 714

Hartmut Klauck

Part II Overview

- Problems that cannot be solved efficiently
  - P vs. NP
  - Time Hierarchy
- Problems that cannot be solved at all
  - Computability
- Weaker models of computation
  - Finite Automata

Languages

- **Definition**:
  - An *alphabet* is a finite set of symbols
  - Γ* is the set of all finite sequences/strings over the alphabet Γ
  - A *language* over alphabet Γ is a subset of Γ*
  - A machine *decides* a language L if on input x it outputs 1 if x$\in$L and 0 otherwise
- A *complexity class* is a set of languages that can be computed given some restricted resources

The Class P

- The class P consists of all languages that can be decided in polynomial time

- Which machine model?
  - RAM's with the logarithmic cost measure
  - Simpler: Turing machines
  - Polynomial size circuits (with simple descriptions)

The Class P

- For technical reasons P contains only decision problems
- Example: Sorting can be done in polynomial time, but is not a language
- Decision version:
  - ElementDistinctness=$\{x_1,\ldots, x_n$: the x_i are pairwise distinct strings of length $n\}$
- ElementDistinctness$\in$P

The Class P

- Problems solvable in polynomial time?
  - Sorting
  - Minimum Spanning Trees
  - Matching
  - Max Flow
  - Shortest Path
  - Linear Programming
  - Many more
- Decision version example: {G,W,K: there is a spanning tree of weight at most K in G}

Turing Machine

- Defined by Turing in 1936 to formalize the notion of computation
- A Turing machine has a finite control and a 1-dimensional storage tape it can access with its read/write head

- Operation: the machine reads a symbol from the tape, does an internal computation and writes another symbol to the tape, moves the head

Turing Machine

- A Turing machine is a 8-tuple
  (Q, Γ, b, Σ, q_0, A,R, δ)
- Q: set of states of the machine
- Γ: tape alphabet
- b$\in\Gamma$: blank symbol
- $\Sigma\subseteq\Gamma$-{b}: input alphabet
- $q_0\in$ Q: initial state
- A,R $\subseteq$ Q: accepting/rejecting states
- δ: Q- (A$\cup$R) $\times\ \Gamma\rightarrow$ Q$\times\Gamma\times${left,stay,right}: transition function

Operation

- The tape consists of an infinite number of cells labeled by all integers
- In the beginning the tape contains the input $x \in \Sigma^*$ starting at tape cell 0
- The rest of the tape contains blank symbols
- The machine starts in state q_0
- The head is on cell 0 in the beginning

Operation

- In every step the machine reads the symbol z at the position of the head
- Given z and the current state q it uses δ to determine the new state, the symbol that is written to the tape and the movement of the head
  - left, stay, right
- If the machine reaches a state in A it stops and accepts, on states in R it rejects

Example Turing Machine

- To compute the parity of $x \in \{0,1\}^*$
- $Q=\{q_0, q_1, q_a, q_r\}$
- $\Gamma=\{0,1,b\}$
- δ:

$q_0, 1 \rightarrow q_1, b$, right
$q_0, 0 \rightarrow q_0, b$, right
$q_1, 1 \rightarrow q_0, b$, right
$q_1, 0 \rightarrow q_1, b$, right
$q_1, b \rightarrow q_a$
$q_0, b \rightarrow q_r$

Example

- The Turing machine here only moves right and does not write anything useful
  - It is a finite automaton

Correctness/Time

- A TM *decides* L if it accepts all $x \in L$ and rejects all x not in L (and halts on all inputs)
- The *time* used by a TM on input x is the number of steps [evaluations of δ] before the machine reaches a state in A or R
- The *time complexity* of a TM M is the function t_M that maps n to the largest time used on any input in Σ^n
- The *time complexity* of L is upper bounded by g(n) if there is a TM M that decides L and has $t_M(n) \leq O(g(n))$

Notes

- DTIME(f(n)) is the class of all languages that have time complexity at most O(f(n))
- P is the class of languages L such that the time complexity of L can be upper bounded by a fixed polynomial in n [with a fixed highest power of n appearing in p]
- There are languages for which there is no asymptotically fastest TM [Speedup theorem]

Space

- The space used by a TM M on an input x is the number of cells visited by the head
- The space complexity of M is the function s_M mapping n to the largest space used on $x \in \Sigma^n$
- The space complexity of L is upper bounded by g(n) if there is a TM that decides L and $s_M(n)=O(g(n))$

Facts

- A Turing machine can simulate a RAM with log cost measure such that
  - polynomial time RAM gives a polynomial time TM
- A log-cost RAM can simulate a TM
  - Store the tape in the registers
  - Store the current state in a register
  - Each register stores a symbol or state [O(1) bits]
  - Store also the head position in a register [log s_M bits]
  - Compute the transition function by table lookup
- Hence the definition of P is robust

Criticism

- P is supposed to represent efficiently solvable problems
- P contains only languages
  - Can identify a problem with many outputs with a set of languages (one for each output bit)
- Problems with time complexity n^{1000} are deemed easy while problems with time complexity $2^{n/100000}$ hard
- Answer: P is mainly a theoretical tool
- In practice such problems don't seem to arise
- Once a problem is known to be in P we can start searching for more efficient algorithms

Criticism

- Turing machines might not be the most powerful model of computation
- All computers currently built can be simulated efficiently
  - Small issue: randomization
- Some models have been proposed that are faster, e.g. analogue computation
  - Usually not realistic models
- Exception: Quantum computers
  - Quantum Turing machines are probably faster than Turing machines for some problems

Why P?

- P has nice closure properties

- **Example:** closed under calling subroutines:
  - Suppose $R \in P$
  - If there is a polynomial time algorithm that solves L given a *free* subroutine that computes R, then L is also in P

Variants of TM

- Several tapes
  - Often easier to describe algorithms
    - Example: Palindrome={xy: x is y in reverse}
    - Compute length, copy x on another tape, compare x and y in reverse
    - Any 1-tape TM needs quadratic time
  - Any TM with O(1) tapes and time T(n) can be simulated by a 1-tape TM using time $O(T(n)^2)$