Complexity Theory

Jan Křetínský

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

Summer 2019

Partially based on slides by Jörg Kreiker

Lecture 1

Introduction

Agenda

- computational complexity and two problems
- your background and expectations
- organization
- basic concepts
- teaser
- summary

Computational Complexity

- quantifying the efficiency of computations
- not: computability, descriptive complexity, ...
- computation: computing a function $f : \{0,1\}^* \to \{0,1\}^*$
  - everything else matter of encoding
  - model of computation?
- efficiency: how many resources used by computation
  - time: number of basic operations with respect to input size
  - space: memory usage

Dinner Party

Example (Dinner Party)

You want to throw a dinner party. You have a list of pairs of friends who do not get along. What is the largest party you can throw such that you do not invite any two who don't get along?

Dinner Party

Example (Dinner Party)

You want to throw a dinner party. You have a list of pairs of friends who do not get along. What is the largest party you can throw such that you do not invite any two who don't get along?

person	does not get along with
Jack	James, John, Kate
James	Jack, Hugo, Sayid
John	Jack, Juliet, Sun
Kate	Jack, Claire, Jin
Hugo	James, Claire, Sun
Claire	Hugo, Kate, Juliet
Juliet	John, Sayid, Claire
Sun	John, Hugo, Jin
Sayid	James, Juliet, Jin
Jin	Sayid, Sun, Kate

Dinner Party

Example (Dinner Party)

You want to throw a dinner party. You have a list of pairs of friends who do not get along. What is the largest party you can throw such that you do not invite any two who don't get along?

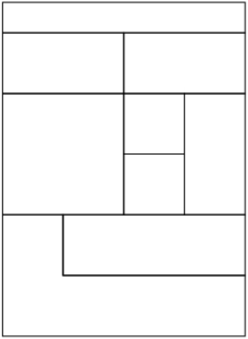
person	does not get along with
Jack	James, John, Kate
James	Jack, Hugo, Sayid
John	Jack, Juliet, Sun
Kate	Jack, Claire, Jin
Hugo	James, Claire, Sun
Claire	Hugo, Kate, Juliet
Juliet	John, Sayid, Claire
Sun	John, Hugo, Jin
Sayid	James, Juliet, Jin
Jin	Sayid, Sun, Kate

- largest party?
- naive computation
  - check all sets of people for compatibility
  - number of subsets of n element set is 2^n
  - intractable
- can we do better?
- observation: for a given set compatibility checking is easy

Map Coloring

Example (Map Coloring)

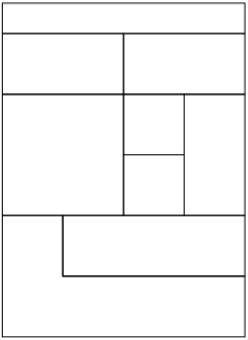
Can you color a map with three different colors, such that no pair of adjacent countries has the same color. Countries are adjacent if they have a non-zero length, shared border.



Map Coloring

Example (Map Coloring)

Can you color a map with three different colors, such that no pair of adjacent countries has the same color. Countries are adjacent if they have a non-zero length, shared border.



- naive algorithm: try all colorings and check
- number of 3-colorings for n countries: 3^n
- can we do better?
- observation: for a given coloring compatibilty checking is easy

What about you?

- What do you expect?
- What do you already know about complexity?
- Immediate feedback

Organization

- lecture in English
- course website:
  http://www7.in.tum.de/um/courses/complexity/SS19/
- concentrated into the first part of the semester, in 03.09.014
  - (reserved slot Monday 14-16)
  - Tuesday 10:05-11:35 and 12:25-13:55
  - Wednesday 8:25-9:55
  - Friday 12:05-13:35 and 14:00-15:30
- tutor: Mikhail Raskin
- weekly exercise sheets, not mandatory
- written or oral exam, depending on number of students
- bonus
  - several mini-tests during lectures (un-announced, cover 2-4 lectures)
  - self-assessment and feedback to us
  - if C is ratio of correct answers, exam bonus computed by

$$\frac{\lceil 5C - 1 \rceil}{2}$$

Literature

- lecture based on Computational Complexity: A Modern Approach by **Sanjeev Arora** and **Boaz Barak**
- book website: `http://www.cs.princeton.edu/theory/complexity/`
- useful links plus freely available draft
- lecture is self-contained
- more recommended reading on course website, e.g. Introduction to the Theory of Computation by **Michael Sipser**

Agenda

- computational complexity and two problems ✓
- your background and expectations ✓
- organization ✓
- basic concepts
- teaser
- summary

Prerequisites

- sets, relations, functions
- formal languages
- Turing machines
- graphs and algorithms on graphs
- little probability theory
- Landau symbols

Landau symbols

- characterize asymptotic behavior of functions (on integers, reals)
- ignore constant factors
- useful to talk about resource usage

Landau symbols

- characterize asymptotic behavior of functions (on integers, reals)
- ignore constant factors
- useful to talk about resource usage
- upper bound: $f \in \mathcal{O}(g)$ defined by
  $\exists c > 0.\ \exists n_0 > 0.\ \forall n > n_0.\ f(n) \leq c \cdot g(n)$
- dominated by: $f \in o(g)$ defined by $\forall \varepsilon > 0.\ \exists n_0 > 0.\ \forall n > n_0.\ \frac{f(n)}{g(n)} < \varepsilon$
- lower bound: $f \in \Omega(g)$ iff $g \in \mathcal{O}(f)$
- tight bound: $f \in \Theta(g)$ iff $f \in \mathcal{O}(g)$ and $f \in \Omega(g)$
- dominating: $f \in \omega(g)$ iff $g \in o(f)$

Intractability

POLYNOMIAL

versus

EXPONENTIAL

- computations using exponential time or space intractable for all but the smallest inputs
- for a map with 200 countries: app. $2.66 \cdot 10^{95}$ 3-colorings
- atoms in the universe (wikipedia): $8 \cdot 10^{80}$
- computational complexity: tractable vs. intractable
- tractable: problems with runtimes $\bigcup_{p>0} \mathcal{O}(n^p)$
- intractable: problems with worse runtimes, e.g. $2^{\Omega(n)}$
- independent of hardware

What about our examples?

- dinner party problem tractable?
- map coloring problem tractable?
- lower bounds on time/space consumption
- upper bounds on time/space consumption
- which is harder?

Dinner Party

person	does not get along with
Jack	James, John, Kate
James	Jack, Hugo, Sayid
John	Jack, Juliet, Sun
Kate	Jack, Claire, Jin
Hugo	James, Claire, Sun
Claire	Hugo, Kate, Juliet
Juliet	John, Sayid, Claire
Sun	John, Hugo, Jin
Sayid	James, Juliet, Jin
Jin	Sayid, Sun, Kate

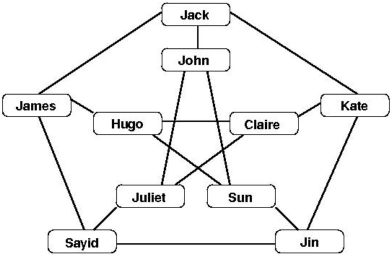
Dinner Party

person	does not get along with
Jack	James, John, Kate
James	Jack, Hugo, Sayid
John	Jack, Juliet, Sun
Kate	Jack, Claire, Jin
Hugo	James, Claire, Sun
Claire	Hugo, Kate, Juliet
Juliet	John, Sayid, Claire
Sun	John, Hugo, Jin
Sayid	James, Juliet, Jin
Jin	Sayid, Sun, Kate

- really a graph problem
- each person a node, each relation an edge
- find a maximal set of nodes, such that no two nodes are adjacent

Dinner Party

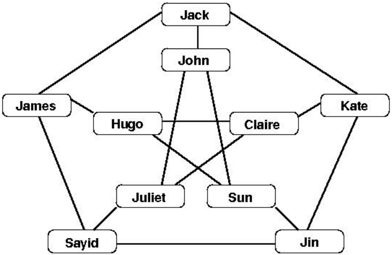
person	does not get along with
Jack	James, John, Kate
James	Jack, Hugo, Sayid
John	Jack, Juliet, Sun
Kate	Jack, Claire, Jin
Hugo	James, Claire, Sun
Claire	Hugo, Kate, Juliet
Juliet	John, Sayid, Claire
Sun	John, Hugo, Jin
Sayid	James, Juliet, Jin
Jin	Sayid, Sun, Kate



- really a graph problem
- each person a node, each relation an edge
- find a maximal set of nodes, such that no two nodes are adjacent

Dinner Party

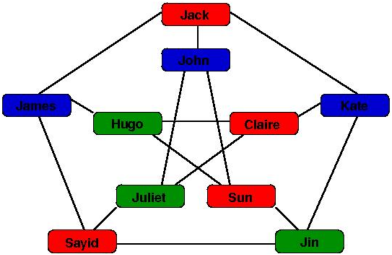
person	does not get along with
Jack	James, John, Kate
James	Jack, Hugo, Sayid
John	Jack, Juliet, Sun
Kate	Jack, Claire, Jin
Hugo	James, Claire, Sun
Claire	Hugo, Kate, Juliet
Juliet	John, Sayid, Claire
Sun	John, Hugo, Jin
Sayid	James, Juliet, Jin
Jin	Sayid, Sun, Kate



- really a graph problem
- each person a node, each relation an edge
- find a maximal set of nodes, such that no two nodes are adjacent
- the independent set problem: Indset
- probably not tractable, no algorithm better than naive one known

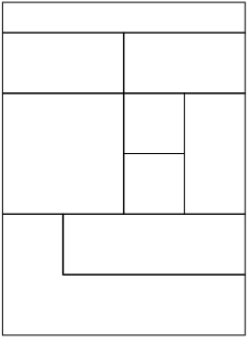
Dinner Party

person	does not get along with
Jack	James, John, Kate
James	Jack, Hugo, Sayid
John	Jack, Juliet, Sun
Kate	Jack, Claire, Jin
Hugo	James, Claire, Sun
Claire	Hugo, Kate, Juliet
Juliet	John, Sayid, Claire
Sun	John, Hugo, Jin
Sayid	James, Juliet, Jin
Jin	Sayid, Sun, Kate

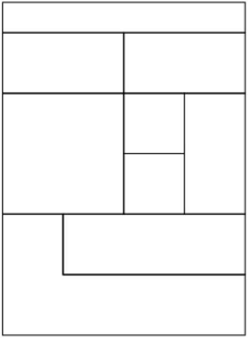


- really a graph problem
- each person a node, each relation an edge
- find a maximal set of nodes, such that no two nodes are adjacent
- the independent set problem: Indset
- probably not tractable, no algorithm better than naive one known
- here: maximal independent set of size 4

Map Coloring

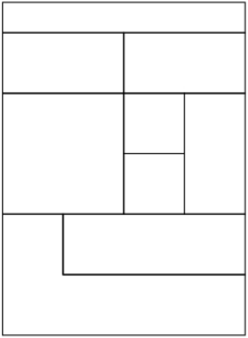


Map Coloring



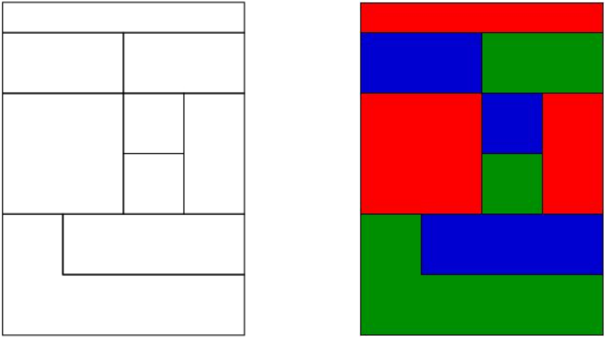
- really a graph problem
- each country a node, each border an edge
- color each node such that no two adjacent nodes have same color

Map Coloring



- really a graph problem
- each country a node, each border an edge
- color each node such that no two adjacent nodes have same color
- the three coloring problem: 3–Coloring
- probably not tractable, no algorithm better than naive one known

Map Coloring



- really a graph problem
- each country a node, each border an edge
- color each node such that no two adjacent nodes have same color
- the three coloring problem: 3−Coloring
- probably not tractable, no algorithm better than naive one known
- here: answer is yes

Bounds

- upper bounds
  - time (naive algorithm): $2^{\mathcal{O}(n)}$ for n persons/countries
  - space (naive algorith): $\mathcal{O}(n^p)$ for n persons/countries and a small p

Bounds

- upper bounds
  - time (naive algorithm): $2^{\mathcal{O}(n)}$ for n persons/countries
  - space (naive algorith): $\mathcal{O}(n^p)$ for n persons/countries and a small p
- lower bounds
  - very little known
  - difficult because of infinitely many algorithms
  - both problems could have a linear time and a logarithmic space algorithm
  - but not simultaneously

Which is harder?

- instead of tight bounds say which problem is harder
- ⇒ reductions

Which is harder?

- instead of tight bounds say which problem is harder
- $\Rightarrow$ reductions

IF there is an efficient procedure for B using a procedure for A

THEN B cannot be radically harder than A

notation: $B \leq A$

Which is harder?

- instead of tight bounds say which problem is harder
- $\Rightarrow$ reductions

IF there is an efficient procedure for B using a procedure for A

THEN B cannot be radically harder than A

notation: $B \leq A$

Applications:

- **IF**
  - there is an efficient procedure for problem A and
  - $B \leq A$

  THEN there is an efficient procedure for problem B
- **IF**
  - there is no efficient procedure for problem B and
  - $B \leq A$

  THEN there is no efficient procedure for problem A

3−Coloring $\leq$ Indset

How can we solve 3−Coloring using an algorithm to solve Indset?

$3-$Coloring $\leq$ Indset

How can we solve $3-$Coloring using an algorithm to solve Indset?

- triplicate the original graph (V, E) into $(V \times \{1,2,3\}, E')$ where

$$\begin{aligned} E' = &\{((v,i),(w,i)) \mid (v,w) \in E, i \in \{1,2,3\}\} \cup \\ &\{((v,i),(v,j)) \mid v \in V, i \neq j \in \{1,2,3\}\} \end{aligned}$$

efficient

$3-$Coloring $\leq$ Indset

How can we solve $3-$Coloring using an algorithm to solve Indset?

- triplicate the original graph (V, E) into $(V \times \{1,2,3\}, E')$ where

$$\begin{aligned} E' = &\{((v,i),(w,i)) \mid (v,w) \in E, i \in \{1,2,3\}\} \cup \\ &\{((v,i),(v,j)) \mid v \in V, i \neq j \in \{1,2,3\}\} \end{aligned}$$

efficient

- check whether there is an independent set of size $|V|$

assumed efficient

$3-$Coloring $\leq$ Indset

How can we solve $3-$Coloring using an algorithm to solve Indset?

- triplicate the original graph (V, E) into $(V \times \{1,2,3\}, E')$ where

$$
\begin{aligned}
E' = & \{((v,i),(w,i)) \mid (v,w) \in E, i \in \{1,2,3\}\} \cup \\
& \{((v,i),(v,j)) \mid v \in V, i \neq j \in \{1,2,3\}\}
\end{aligned}
$$

efficient

- check whether there is an independent set of size $|V|$

assumed efficient

Need to ensure: procedure returns **yes** if and only if the graph is 3-colorable.

Polynomial certificates: NP

- whole class of problems can be reduced to Indset
- all of them have polynomially checkable certificates
- characterizes (in)famous class **NP**
- all problems in **NP** reducible to Indset makes Indset **NP**-hard.
- 3−Coloring also **NP**-hard
- no polynomial-time algorithms known for **NP**-hard problems
- not all problems have polynomial certificates, e.g. winning strategy in chess

Agenda

- computational complexity and two problems ✓
- your background and expectations ✓
- organization ✓
- basic concepts ✓
- teaser
- summary

Lots of things to explore

- precise definition of computational model and resources
- problems with polynomial time checkable certificates
- space classes
- approximations
- hierarchies (polynomial, time/space tradeoffs)
- randomization
- parallelism
- average case complexities
- probabilistically checkable proofs
- (quantum computing)
- (logical characterizations of complexity classes)
- a bag of proof techniques

What have we learnt?

- polynomial $\sim$ tractable; exponential $\sim$ intractable
- lower bounds hard to come by
- reductions key to establish relations among (classes of problems)
- **NP**: polynomially checkable certificates
- Indset $\in$ **NP**, 3$-$Coloring $\in$ **NP**

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 9, 2019

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!

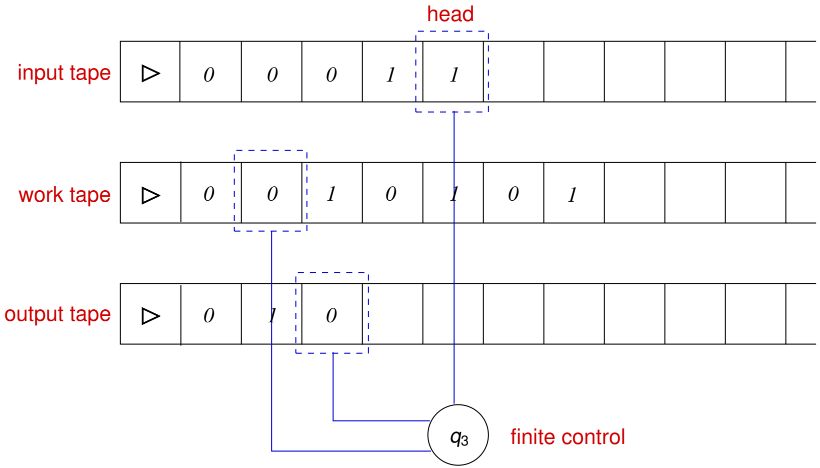
Which models of computation do you know?

- 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

TMs – illustrated



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

TMs – reading palindromes

TM for function $pal : \{0,1\}^* \to \{0,1\}$ 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}
- $\delta : Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{l, s, r\}^k$, transition function such that $\delta(q_{halt}, \vec{\sigma}) = (q_{halt}, \vec{\sigma}_{2..k}, \vec{s})$.

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 $T : \mathbb{N} \to \mathbb{N}$ and $f : \{0,1\}^* \to \{0,1\}^*$ 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 $(\sigma_1, \ldots, \sigma_k)$ are symbols being read, and $\delta(q, (\sigma_1, \ldots, \sigma_k)) = (q', (\sigma'_2, \ldots, \sigma'_k), \vec{z})$, 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 *l*eft, *s*tayed, or *r*ight 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

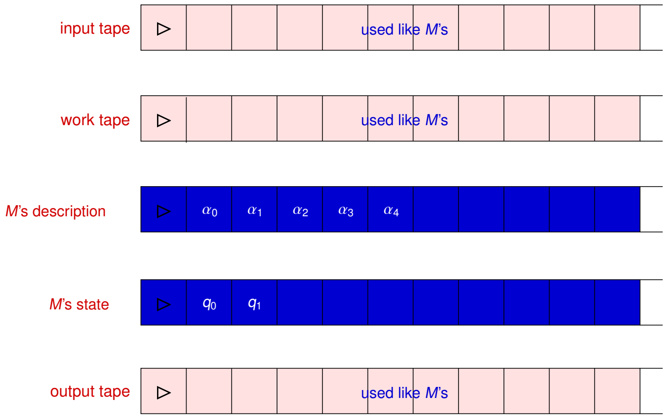
- TMs can be represented as strings (over $\{0, 1\}$) by encoding their transition function (can you?)
  - write M_α for TM represented by string α
  - every string α represents some TM
  - every TM has infinitely many representations
- if TM M computes f, universal TM $\mathcal{U}$ takes representation 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
- $\mathcal{U}$ has bounded alphabet, rules, tapes; simulates much larger machines efficiently

Efficient simulation

Theorem (Universal TM)

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

Construction of $\mathcal{U}$



Simulating another TM

How does $\mathcal{U}$ execute TM M?

Simulating another TM

How does $\mathcal{U}$ execute TM M?

1. transform M into M' with one input, one work, and one output tape computing the same function — quadratic overhead
2. write M''s description α onto third tape — $|M'|$
3. write encoding of M' start state on fourth tape — $|Q'|$
4. for each step of M'
   - 4.1 depending on state and tapes of M' scan δ' tape — $|\delta'|$
   - 4.2 update — constant

Simulation can be done with logarithmic slowdown using clever encoding of k tapes in one.

Agenda

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

Deciding languages

- often one is interested in functions $f : \{0,1\}^* \to \{0,1\}$
- 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 $T : \mathbb{N} \to \mathbb{N}$ be a function. $L \subseteq \{0, 1\}^*$ is in **DTIME**(T) if there exists a TM deciding L in time T' for $T' \in \mathcal{O}(T)$.

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

P

Definition (P)

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

- **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**

Complexity Theory

Jan Křetínský

Technical University of Munich
Summer 2019

April 29, 2019

Lecture 3

Basic Complexity Classes

Agenda

- decision vs. search
- basic complexity classes
  - time and space
  - deterministic and non-deterministic
- sample problems

Decision vs. Search

- often one is interested in functions $f : \{0,1\}^* \to \{0,1\}$
- 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)

Example (Indset)

Consider the independent set problem.

Search What is the largest independent set of a graph?

Decision $\text{Indset} = \{\langle G, k\rangle \mid G \text{ has independent set of size } k\}$

Often decision plus binary search can solve search problems.

Agenda

- decision vs. search ✓
- basic complexity classes
  - time and space
  - deterministic and non-deterministic
- sample problems

Time complexity

Definition (DTIME)

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

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

Space complexity

Definition (SPACE)

Let $S : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$. Define $L \in \mathbf{SPACE}(S)$ iff

- there exists a TM M deciding L
- no more than $S'(n)$ locations on M's work tapes ever visited during computations on every input of length n for $S' \in \mathcal{O}(S)$

Remarks

- more detailed definition (cf. exercises): count non-$\square$ symbols, where $\square$ must not be written
- constants do not matter
- as for time complexity, require space-constructible bounds
  - S is space-constructible: there is TM M computing $S(|x|)$ in $\mathcal{O}(S(|x|))$ space on input x
  - TM knows its bounds
- work tape restrictions: allows to store input
- space bounds $< n$ make sense (as opposed to time)
- require space $\log n$ to remember positions in input

Non-deterministic TMs

Definition (NDTM)

A non-deterministic TM (NDTM) is a triple (Γ, Q, δ) like a deterministic TM except

- Q contains a distinguished state q_{accept}
- δ is a pair (δ_0, δ_1) of transition functions

- in each step, NDTM non-deterministically chooses to apply either δ_0 or δ_1
- NDTM M accepts x, $M(x) = 1$ if there exists a sequence of choices s.t. M reaches q_{accept}
- $M(x) = 0$ if every sequence of choices makes M halt without reaching q_{accept}

On non-determinism

- not supposed to model realistic devices
- remember impact of non-determinism finite state machines, pushdown automata
- NDTM compute the same functions as DTM (why?)
- non-determinism $\sim$ guessing

Non-deterministic complexity
Define **NTIME**(T) and **NSPACE**(S) such that T and S are bounds regardless of non-deterministic choices.

Basic complexity classes

deterministic | non-deterministic

time

P	=	$\bigcup_{p\geq 1}$ **DTIME**(n^p)	**NP**	=	$\bigcup_{p\geq 1}$ **NTIME**(n^p)
EXP	=	$\bigcup_{p\geq 1}$ **DTIME**(2^{n^p})	**NEXP**	=	$\bigcup_{p\geq 1}$ **NTIME**(2^{n^p})

space

L	=	**SPACE**$(\log n)$	**NL**	=	**NSPACE**$(\log n)$
PSPACE	=	$\bigcup_{p>0}$ **SPACE**(n^p)	**NPSPACE**	=	$\bigcup_{p>0}$ **NSPACE**(n^p)

Agenda

- decision vs. search ✓
- basic complexity classes ✓
    - time and space
    - deterministic and non-deterministic
- sample problems

Interesting examples

Most examples are the hardest within a given complexity class. They are complete for the class (wrt suitable reductions).

Interesting examples

Most examples are the hardest within a given complexity class. They are complete for the class (wrt suitable reductions).

L: essentially constant number of pointers into input plus logarithmically many boolean flags

- UPath $= \{\langle G, s, t\rangle \mid \exists$a path from s to t in **undirected** graph $G\}$ [Reingold 2004]
- Even $= \{x \mid x$ has an even number of 1s$\}$

Interesting examples

Most examples are the hardest within a given complexity class. They are complete for the class (wrt suitable reductions).

L: essentially constant number of pointers into input plus logarithmically many boolean flags

- UPath $= \{\langle G, s, t\rangle \mid \exists$a path from s to t in **undirected** graph $G\}$ [Reingold 2004]
- Even $= \{x \mid x$ has an even number of 1s$\}$

NL: **L** plus guessing, read-once certificates

- Path $= \{\langle G, s, t\rangle \mid \exists$a path from s to t in **directed** graph $G\}$
- 2SAT $= \{\varphi \mid$ φ satisfiable Boolean formula in CNF with two literals per clause $\}$

Interesting examples

P: polynomial time, tractable, low-level choices of TM definitions are immaterial to **P**

- $\text{Circuit} - \text{Eval} = \{\langle C, x\rangle \mid C \text{ is a } n\text{−in}/1\text{−out circuit, } x \text{satisfying signals}\}$
- $\text{Primes} = \{x \mid x \text{ prime}\}$ [AKS 2004]
- many graph problems like DFS and BFS

Interesting examples

P: polynomial time, tractable, low-level choices of TM definitions are immaterial to **P**

- $\mathrm{Circuit-Eval} = \{\langle C, x\rangle \mid C$ is a $n-$in$/1-$out circuit, $x$satisfying signals$\}$
- $\mathrm{Primes} = \{x \mid x$ prime$\}$ [AKS 2004]
- many graph problems like DFS and BFS

NP: polynomially verifiable certificates, puzzles

- $\mathrm{Indset} = \{\langle G, k\rangle \mid G$ has an independent set of size $k\}$
- $3-\mathrm{Coloring} = \{G \mid G$ is 3-colorable$\}$
- $\mathrm{3SAT} = \{\varphi \mid \varphi$ satisfiable Boolean formula in CNF with three literals per clause $\}$

Interesting examples

P: polynomial time, tractable, low-level choices of TM definitions are immaterial to **P**

- $\mathrm{Circuit-Eval} = \{\langle C, x\rangle \mid C$ is a $n-$in$/1-$out circuit, $x$satisfying signals$\}$
- $\mathrm{Primes} = \{x \mid x$ prime$\}$ [AKS 2004]
- many graph problems like DFS and BFS

NP: polynomially verifiable certificates, puzzles

- $\mathrm{Indset} = \{\langle G, k\rangle \mid G$ has an independent set of size $k\}$
- $3-\mathrm{Coloring} = \{G \mid G$ is 3-colorable$\}$
- $\mathrm{3SAT} = \{\varphi \mid \varphi$ satisfiable Boolean formula in CNF with three literals per clause $\}$

PSPACE: polynomial space, games, for instance
$\mathrm{TQBF} = \{Q_1 x_1 \ldots Q_k x_k \varphi \mid k \geq 0, Q_i \in \{\forall, \exists\}, \varphi$ Boolean formula over x_i such that whole formula is true $\}$

Interesting examples

P: polynomial time, tractable, low-level choices of TM definitions are immaterial to **P**

- $\text{Circuit}-\text{Eval} = \{\langle C, x\rangle \mid C$ is a $n-$in$/1-$out circuit, $x$satisfying signals$\}$
- $\text{Primes} = \{x \mid x$ prime$\}$ [AKS 2004]
- many graph problems like DFS and BFS

NP: polynomially verifiable certificates, puzzles

- $\text{Indset} = \{\langle G, k\rangle \mid G$ has an independent set of size $k\}$
- $3-\text{Coloring} = \{G \mid G$ is 3-colorable$\}$
- $\text{3SAT} = \{\varphi \mid \varphi$ satisfiable Boolean formula in CNF with three literals per clause $\}$

PSPACE: polynomial space, games, for instance
$\text{TQBF} = \{Q_1 x_1 \ldots Q_k x_k \varphi \mid k \geq 0, Q_i \in \{\forall, \exists\}, \varphi$ Boolean formula over x_i such that whole formula is true $\}$

EXP: exponential-time, for instance the language
$\text{Halt}_\text{k} = \{\langle M, x, k\rangle \mid$ DTM M stops on input x within k steps $\}$

Complements

Definition (Complement classes)

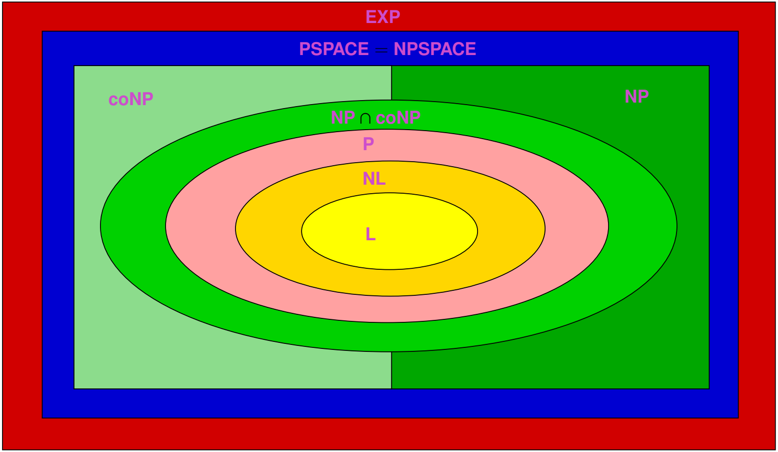
Let $\mathcal{C} \subseteq \mathcal{P}(\{0,1\}^*)$ be a complexity class. We define $co\mathcal{C} = \{\overline{L} \mid L \in \mathcal{C}\}$ to be the complement class of $\mathcal{C}$, where $\overline{L} = \{0,1\}^* \setminus L$ is the complement of L.

- important class **coNP**
- **coNP** is not the complement of **NP**
- example: Tautology $\in$ **coNP**, where a tautology is Boolean formula that is true for every assignment
- reminder: closure under complement wrt expressiveness and conciseness
  - finite state machines
  - pushdown automata
  - DTM, NDTM
- note: **P** $\subseteq$ **NP** $\cap$ **coNP**

Agenda

- universal Turing machine ✓
- decision vs. search ✓
- computability, halting problem ✓
- basic complexity classes ✓

Relation between classes



Teaser

A regular expression over $\{0, 1\}$ is defined by

$$r ::= 0 \mid 1 \mid rr \mid r|r \mid r^*$$

The language defined by r is written $\mathcal{L}(r)$.

What is the computational complexity of

- deciding whether two regular expressions are equivalent, that is $\mathcal{L}(r_1) = \mathcal{L}(r_2)$?
- deciding whether a regular expression is universal, that is $\mathcal{L}(r) = \{0, 1\}^*$?
- deciding the same for star-free regular expressions?

What have we learnt?

- 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
- non-deterministic TMs
- space, time, deterministic, non-deterministic, complement complexity classes
- **L**, **NL**, **P**, **NP**, **EXP**, **PSPACE**
- 2SAT, 3SAT, Path, UPath, TQBF, Primes, Indset, 3−Coloring
- big picture
- up next: justify and explore the big picture

Complexity Theory

Jan Křetínský

Technical University of Munich

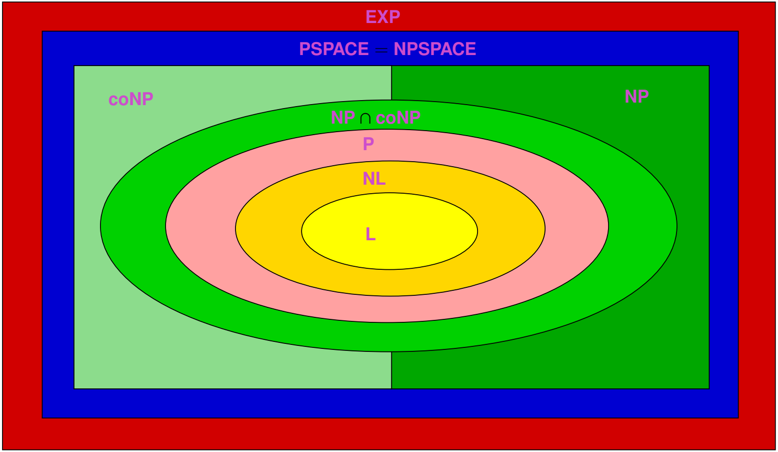
Summer 2019

April 29, 2019

Lecture 4

NP-completeness

Recap: relations between classes



Agenda

- efficiently checkable certificates
- reductions, hardness, completeness
- Cook-Levin: 3SAT is **NP**-complete

NP: efficiently checkable certificates

NP computable with NDTM in polynomial time.

NP: efficiently checkable certificates

NP computable with NDTM in polynomial time.

Theorem (Certificates)

*For every $L \subseteq \{0,1\}^*$ holds: $L \in$ **NP** if and only if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0,1\}^*$*

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)}.\ M(x,u) = 1$$

NP: efficiently checkable certificates

NP computable with NDTM in polynomial time.

Theorem (Certificates)

*For every $L \subseteq \{0,1\}^*$ holds: $L \in$ **NP** if and only if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0,1\}^*$*

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)}.\ M(x,u) = 1$$

- M is called verifier
- u is called certificate

NP: efficiently checkable certificates

NP computable with NDTM in polynomial time.

Theorem (Certificates)

For every $L \subseteq \{0,1\}^*$ *holds:* $L \in$ **NP** *if and only if there exists a polynomial* $p : \mathbb{N} \to \mathbb{N}$ *and a polynomial-time TM* M *such that for every* $x \in \{0,1\}^*$

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)}.\ M(x,u) = 1$$

- M is called verifier
- u is called certificate

Proof:

$\Rightarrow$ certificate is sequence of choices

$\Leftarrow$ NDTM guesses certificate

Examples

- Indset: certificate is set of nodes, size of certificate for k nodes in graph with n nodes $\mathcal{O}(k \log n)$
- 0/1−ILP: given a list of m linear inequalities with rational coefficients over variables $x_1, \ldots, x_k$; find out if there is an assignment of 0s and 1s to x_i satisfying all inequalities; certificate is assignment.
- Iso: given two $n \times n$ adjacency matrices; do they define isomorphic graphs; certificate is a permutation $\pi : [n] \to [n]$

Agenda

- efficiently checkable certificates ✓
- reductions, hardness, completeness
- Cook-Levin: 3SAT is **NP**-complete

Reductions – reminder

IF there is an efficient procedure for B
using a procedure for A (as an efficient black box)

THEN B cannot be radically harder than A

notation: $B \leq A$

(reduction does not make anything *smaller*)

We have seen (at least) two reductions.

- 3−Coloring was reduced to Indset
- the diagonalized, undecidable language reduced to Halt

Reductions – definition

Definition (Karp reduction)

Let $L, L' \subseteq \{0,1\}^*$ be languages. L is polynomial-time Karp reducible to L' iff there exists a polynomial-time computable funtion $f : \{0,1\}^* \to \{0,1\}^*$ such that for all $x \in \{0,1\}^*$

$$x \in L \Leftrightarrow f(x) \in L'$$

We write $L \leq_p L'$.

Reductions – definition

Definition (Karp reduction)

Let $L, L' \subseteq \{0,1\}^*$ be languages. L is polynomial-time Karp reducible to L' iff there exists a polynomial-time computable funtion $f : \{0,1\}^* \to \{0,1\}^*$ such that for all $x \in \{0,1\}^*$

$$x \in L \Leftrightarrow f(x) \in L'$$

We write $L \leq_p L'$.

Note: $\leq_p$ is a transitive relation on languages (because the composition of polynomials is a polynomial).

Hardness and Completeness

Definition (NP-hardness and -completness)

Let $L \subseteq \{0,1\}^*$ be a language.

- L is **NP**-hard if $L' \leq_p L$ for every $L' \in$ **NP**
- L is **NP**-complete if L is **NP**-hard and $L \in$ **NP**.

Hardness and Completeness

Definition (NP-hardness and -completness)

Let $L \subseteq \{0,1\}^*$ be a language.

- L is **NP**-hard if $L' \leq_p L$ for every $L' \in$ **NP**
- L is **NP**-complete if L is **NP**-hard and $L \in$ **NP**.

Examples of **NP**-hard languages: Indset, Halt_k, Halt

Hardness and Completeness

Definition (NP-hardness and -completness)

Let $L \subseteq \{0,1\}^*$ be a language.

- L is **NP**-hard if $L' \leq_p L$ for every $L' \in$ **NP**
- L is **NP**-complete if L is **NP**-hard and $L \in$ **NP**.

Examples of **NP**-hard languages: Indset, Halt_k, Halt

Observation

- L **NP**-hard and $L \in$ **P** implies **P** $=$ **NP**
- L **NP**-complete implies $L \in$ **P** iff **P** $=$ **NP**

Do NP-complete languages exist?

- upcoming result independently discovered by Cook (1971) and Levin (1973)
- uses notion of satisfiable Boolean formulas
- Boolean formula φ over variables $X = \{x_1, \ldots, x_k\}$ defined by

$$\varphi ::= x \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$$

- write $\overline{x}$ instead of $\neg x$, x and $\overline{x}$ literals u
- assume formulas are in CNF:

$$\varphi = \bigwedge_i \bigvee_j u_{i_j}$$

- disjunctions $\bigvee_j u_{i_j}$ called clauses
- formula is in k-CNF if the no clause has more than k literals

Cook-Levin Theorem

- φ is satisfiable iff there exists an assignments $a : X \to \{0, 1\}$ making φ true
- 3SAT $= \{\varphi \mid \varphi$ in 3-CNF and satisfiable$\}$

Theorem

3SAT *is* **NP***-complete.*

Proof agenda

1. SAT is **NP**-complete (without restriction to clauses of size three)
   1.1 SAT, 3SAT $\in$ **NP**
   1.2 for every $L \in$ **NP** $L \leq_p$ SAT
2. Show that SAT $\leq_p$ 3SAT

What have we learnt?

- **NP** is polynomial certificates
- Karp reductions, hardness, completeness
- Cook-Levin: reduce any language in **NP** to 3SAT
- up next: the proof, more **NP**-complete problems, **P** vs. **NP**, tool demos

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 9, 2019

Lecture 5

NP-completeness (2)

Agenda

- Cook-Levin
- SAT demo
- see old friends
  - $0/1-$ILP
  - Indset
  - $3-$Coloring

Cook-Levin: 3SAT is NP-complete

- 3SAT $\in$ **NP**

Cook-Levin: 3SAT is NP-complete

- 3SAT $\in$ **NP**
  - the assignement forms a polynomial certificate
- 3SAT is **NP**-hard

Cook-Levin: 3SAT is NP-complete

- 3SAT $\in$ **NP**
  - the assignement forms a polynomial certificate
- 3SAT is **NP**-hard
  - choose $L \in$ **NP** arbitrary, $L \subseteq \{0, 1\}^*$
  - find reduction f from L to 3SAT

Cook-Levin: 3SAT is NP-complete

- 3SAT $\in$ **NP**
  - the assignement forms a polynomial certificate
- 3SAT is **NP**-hard
  - choose $L \in$ **NP** arbitrary, $L \subseteq \{0, 1\}^*$
  - find reduction f from L to 3SAT
    - $\forall x \in \{0, 1\}^*$: $x \in L \Leftrightarrow f(x) \in$ 3SAT i.e. φ_x is satisfiable
    - f is polynomial time computable

TMs for L and f

$L \in$ **NP** iff there exists a TM M that runs in time T and there is a polynomial p such that

$$\forall x \in L \; \exists u \in \{0,1\}^{p(|x|)} \; M(x,u) = 1 \Leftrightarrow x \in L$$

TMs for L and f

$L \in$ **NP** iff there exists a TM M that runs in time T and there is a polynomial p such that

$$\forall x \in L \ \exists u \in \{0,1\}^{p(|x|)} \ M(x,u) = 1 \Leftrightarrow x \in L$$

Assumptions

- fix $n \in \mathbb{N}$ and $x \in \{0,1\}^n$ arbitrary
- $m = n + p(n)$
- $M = (\Gamma, Q, \delta)$
- M is oblivious
- M has two tapes
- define TM M_f that takes M, T, p, x and outputs φ_x

M_f exploits obliviousness

1. simulate M on $0^{n+p(n)}$ for $T(n+p(n))$ steps

M_f exploits obliviousness

1. simulate M on $0^{n+p(n)}$ for $T(n+p(n))$ steps
2. for each $1 \leq i \leq T(n+p(n))$ store
   - $inputpos(i)$: position of input head after i steps
   - $prev(i)$: previous step when work head was here (default 1)

M_f exploits obliviousness

1. simulate M on $0^{n+p(n)}$ for $T(n+p(n))$ steps
2. for each $1 \leq i \leq T(n+p(n))$ store
   - $inputpos(i)$: position of input head after i steps
   - $prev(i)$: previous step when work head was here (default 1)
3. compute and output φ_x

M_f exploits obliviousness

1. simulate M on $0^{n+p(n)}$ for $T(n+p(n))$ steps
2. for each $1 \leq i \leq T(n+p(n))$ store
   - $inputpos(i)$: position of input head after i steps
   - $prev(i)$: previous step when work head was here (default 1)
3. compute and output φ_x

It does all this in time polynomial in n!

Variables of φ_x

- "input variables" $y_1, \ldots, y_n, y_{n+1}, \ldots y_{n+p(n)}$

Variables of φ_x

- "input variables" $y_1, \ldots, y_n, y_{n+1}, \ldots y_{n+p(n)}$
    - to encode the read-only input tape

Variables of φ_x

- "input variables" $y_1, \ldots, y_n, y_{n+1}, \ldots y_{n+p(n)}$
  - to encode the read-only input tape
  - $y_1, \ldots, y_n$ determined by x

Variables of φ_x

- "input variables" $y_1, \ldots, y_n, y_{n+1}, \ldots y_{n+p(n)}$
  - to encode the read-only input tape
  - $y_1, \ldots, y_n$ determined by x
  - $y_{n+1}, \ldots y_{n+p(n)}$ will be certificate

Variables of φ_x

- "input variables" $y_1, \dots, y_n, y_{n+1}, \dots y_{n+p(n)}$
  - to encode the read-only input tape
  - $y_1, \dots, y_n$ determined by x
  - $y_{n+1}, \dots y_{n+p(n)}$ will be certificate
- "computation variables"

$$\begin{array}{lllll} z_1 & z_2 & \dots & z_{c-1} & z_c \\ z_{c+1} & z_{c+2} & \dots & z_{2c-1} & z_{2c} \\ \vdots & & & & \vdots \\ z_{c(T(m)-1)+1} & & & & z_{cT(m)} \end{array}$$

Variables of φ_x

- "input variables" $y_1, \ldots, y_n, y_{n+1}, \ldots y_{n+p(n)}$
  - to encode the read-only input tape
  - $y_1, \ldots, y_n$ determined by x
  - $y_{n+1}, \ldots y_{n+p(n)}$ will be certificate
- "computation variables"

$$
\begin{array}{lllll}
z_1 & z_2 & \cdots & z_{c-1} & z_c \\
z_{c+1} & z_{c+2} & \cdots & z_{2c-1} & z_{2c} \\
\vdots & & & & \vdots \\
z_{c(T(m)-1)+1} & & & & z_{cT(m)}
\end{array}
$$

  - each row a snapshot
  - needs $c-2$ bits to encode state q (independent of x) and 2 bits for the symbols read
- φ_x means "computation on the input is accepting"

Snapshot $s_i = \langle q, 0, 1 \rangle$

- state of M at step i, input and work symbol currently read

Snapshot $s_i = \langle q, 0, 1 \rangle$

- state of M at step i, input and work symbol currently read

Accepting computation of M on $\langle x, u \rangle$ is a sequence of $T(m)$ snapshots such that

Snapshot $s_i = \langle q, 0, 1 \rangle$

- state of M at step i, input and work symbol currently read

Accepting computation of M on $\langle x, u \rangle$ is a sequence of $T(m)$ snapshots such that

- first snapshot s_1 is $\langle q_{start}, \triangleright, \triangleright \rangle$

Snapshot $s_i = \langle q, 0, 1 \rangle$

- state of M at step i, input and work symbol currently read

Accepting computation of M on $\langle x, u \rangle$ is a sequence of $T(m)$ snapshots such that

- first snapshot s_1 is $\langle q_{start}, \triangleright, \triangleright \rangle$
- last snapshot $s_{T(m)}$ has state q_{halt} and ouputs 1

Snapshot $s_i = \langle q, 0, 1 \rangle$

- state of M at step i, input and work symbol currently read

Accepting computation of M on $\langle x, u \rangle$ is a sequence of $T(m)$ snapshots such that

- first snapshot s_1 is $\langle q_{start}, \triangleright, \triangleright \rangle$
- last snapshot $s_{T(m)}$ has state q_{halt} and ouputs 1
- s_{i+1} computed correctly from
  - δ
  - s_i
  - $y_{inputpos(i+1)}$
  - $s_{prev(i+1)}$

$$\varphi_x = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$$

1. relate x and $y_1, \ldots, y_m$: $\bigwedge_{1 \le i \le n} x_i = y_i$, where
   $x = y \Leftrightarrow (x \vee \overline{y}) \wedge (\overline{x} \vee y)$

$\varphi_x = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$

1. relate x and $y_1, \ldots, y_m$: $\bigwedge_{1 \le i \le n} x_i = y_i$, where
   $x = y \Leftrightarrow (x \vee \overline{y}) \wedge (\overline{x} \vee y)$
   $\rightarrow$ size $4n$

$\varphi_x = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$

1. relate x and $y_1, \ldots, y_m$: $\bigwedge_{1 \leq i \leq n} x_i = y_i$, where $x = y \Leftrightarrow (x \vee \overline{y}) \wedge (\overline{x} \vee y)$
   $\rightarrow$ size $4n$
2. relate $z_1, \ldots, z_c$ with $\langle q_{start}, \triangleright, \triangleright \rangle$
   $\rightarrow$ size $\mathcal{O}(c)$ (CNF, independent of $|x|$)

$\varphi_x = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$

1. relate x and $y_1, \dots, y_m$: $\bigwedge_{1 \le i \le n} x_i = y_i$, where $x = y \Leftrightarrow (x \vee \overline{y}) \wedge (\overline{x} \vee y)$
   $\rightarrow$ size $4n$
2. relate $z_1, \dots, z_c$ with $\langle q_{start}, \triangleright, \triangleright \rangle$
   $\rightarrow$ size $\mathcal{O}(c)$ (CNF, independent of $|x|$)
3. relate $z_{c(T(m)-1)+1}, \dots, z_{cT(m)}$ with accepting snapshot
   $\rightarrow$ analogous

$\varphi_x = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$

1. relate x and $y_1, \ldots, y_m$: $\bigwedge_{1 \le i \le n} x_i = y_i$, where $x = y \Leftrightarrow (x \vee \overline{y}) \wedge (\overline{x} \vee y)$
   $\rightarrow$ size $4n$
2. relate $z_1, \ldots, z_c$ with $\langle q_{start}, \triangleright, \triangleright\rangle$
   $\rightarrow$ size $O(c)$ (CNF, independent of $|x|$)
3. relate $z_{c(T(m)-1)+1}, \ldots, z_{cT(m)}$ with accepting snapshot
   $\rightarrow$ analogous
4. relate $z_{ci+1}, \ldots, z_{c(i+1)}$ (snapshot s_{i+1}) with
   - $z_{c(i-1)+1}, \ldots, z_{ci-2}$ (state of snapshot s_i)
   - $y_{inputpos(i+1)}$
   - $z_{c \cdot prev(i)}$ (next work tape symbol, from snapshot $s_{prev(i)}$)

$\varphi_x = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$

1. relate x and $y_1, \ldots, y_m$: $\bigwedge_{1 \leq i \leq n} x_i = y_i$, where
   $x = y \Leftrightarrow (x \vee \overline{y}) \wedge (\overline{x} \vee y)$
   $\rightarrow$ size $4n$
2. relate $z_1, \ldots, z_c$ with $\langle q_{start}, \triangleright, \triangleright \rangle$
   $\rightarrow$ size $\mathcal{O}(c)$ (CNF, independent of $|x|$)
3. relate $z_{c(T(m)-1)+1}, \ldots, z_{cT(m)}$ with accepting snapshot
   $\rightarrow$ analogous
4. relate $z_{ci+1}, \ldots, z_{c(i+1)}$ (snapshot s_{i+1}) with
   - $z_{c(i-1)+1}, \ldots, z_{ci-2}$ (state of snapshot s_i)
   - $y_{inputpos(i+1)}$
   - $z_{c \cdot prev(i)}$ (next work tape symbol, from snapshot $s_{prev(i)}$)
   - CNF formula over $2c$ variables, size $\mathcal{O}(c2^{2c})$

$\varphi_x = \varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4$

1. relate x and $y_1, \ldots, y_m$: $\bigwedge_{1 \le i \le n} x_i = y_i$, where
   $x = y \Leftrightarrow (x \vee \overline{y}) \wedge (\overline{x} \vee y)$
   $\rightarrow$ size $4n$
2. relate $z_1, \ldots, z_c$ with $\langle q_{start}, \triangleright, \triangleright \rangle$
   $\rightarrow$ size $\mathcal{O}(c)$ (CNF, independent of $|x|$)
3. relate $z_{c(T(m)-1)+1}, \ldots, z_{cT(m)}$ with accepting snapshot
   $\rightarrow$ analogous
4. relate $z_{ci+1}, \ldots, z_{c(i+1)}$ (snapshot s_{i+1}) with
   - $z_{c(i-1)+1}, \ldots, z_{ci-2}$ (state of snapshot s_i)
   - $y_{inputpos(i+1)}$
   - $z_{c \cdot prev(i)}$ (next work tape symbol, from snapshot $s_{prev(i)}$)
   - CNF formula over $2c$ variables, size $\mathcal{O}(c2^{2c})$

Polynomial in n!

Stop!

- $|\varphi_x|$ polynomial in n
- if φ_x is satisfiable, the satisfying assignment yields certificate $y_{n+1}, \ldots y_{n+p(n)}$
- if a certificate exists in $\{0,1\}^{p(n)}$, we get a satisfying assignment
- M_f can output φ_x in polynomial time

$\Rightarrow$ reduction

Stop!

- $|\varphi_x|$ polynomial in n
- if φ_x is satisfiable, the satisfying assignment yields certificate $y_{n+1}, \ldots y_{n+p(n)}$
- if a certificate exists in $\{0,1\}^{p(n)}$, we get a satisfying assignment
- M_f can output φ_x in polynomial time

$\Rightarrow$ reduction

- but: not to 3SAT

From CNF to 3CNF

As a last polynomial step, M_f applies the following transformation for each clause

$$u_1 \vee u_2 \vee \ldots \vee u_k$$
$$\leadsto$$

From CNF to 3CNF

As a last polynomial step, M_f applies the following transformation for each clause

$$u_1 \vee u_2 \vee \ldots \vee u_k$$
$$\leadsto$$
$$\begin{array}{llllll} & (u_1 & \vee & u_2 & \vee & x_1) \\ \wedge & (\overline{x_1} & \vee & u_3 & \vee & x_2) \end{array}$$

From CNF to 3CNF

As a last polynomial step, M_f applies the following transformation for each clause

$$u_1 \vee u_2 \vee \ldots \vee u_k$$

$$\rightsquigarrow$$

$$\begin{array}{lccccc} & (u_1 & \vee & u_2 & \vee & x_1) \\ \wedge & (\overline{x_1} & \vee & u_3 & \vee & x_2) \\ \wedge & (\overline{x_2} & \vee & u_4 & \vee & x_3) \end{array}$$

From CNF to 3CNF

As a last polynomial step, M_f applies the following transformation for each clause

$$u_1 \vee u_2 \vee \ldots \vee u_k$$

$$\rightsquigarrow$$

$$\begin{array}{lccccc} & (u_1 & \vee & u_2 & \vee & x_1) \\ \wedge & (\overline{x_1} & \vee & u_3 & \vee & x_2) \\ \wedge & (\overline{x_2} & \vee & u_4 & \vee & x_3) \\ \ldots & & & & & \\ \wedge & (\overline{x_{k-2}} & \vee & u_{k-1} & \vee & u_k) \end{array}$$

From CNF to 3CNF

As a last polynomial step, M_f applies the following transformation for each clause

$$u_1 \vee u_2 \vee \ldots \vee u_k$$
$$\rightsquigarrow$$
$$\begin{array}{lccccc} & (u_1 & \vee & u_2 & \vee & x_1) \\ \wedge & (\overline{x_1} & \vee & u_3 & \vee & x_2) \\ \wedge & (\overline{x_2} & \vee & u_4 & \vee & x_3) \\ \ldots & & & & & \\ \wedge & (\overline{x_{k-2}} & \vee & u_{k-1} & \vee & u_k) \end{array}$$

Each clause with k variables transformed into equivalent $k-2$ 3-clauses with $2k-2$ variables. All x_i fresh.

From CNF to 3CNF

As a last polynomial step, M_f applies the following transformation for each clause

$$u_1 \vee u_2 \vee \ldots \vee u_k$$

$$\leadsto$$

$$\begin{array}{lccccc} & (u_1 & \vee & u_2 & \vee & x_1) \\ \wedge & (\overline{x_1} & \vee & u_3 & \vee & x_2) \\ \wedge & (\overline{x_2} & \vee & u_4 & \vee & x_3) \\ \ldots & & & & & \\ \wedge & (\overline{x_{k-2}} & \vee & u_{k-1} & \vee & u_k) \end{array}$$

Each clause with k variables transformed into equivalent $k-2$ 3-clauses with $2k-2$ variables. All x_i fresh.
Example. $x \vee \overline{y} \vee \overline{z} \vee w$ becomes $x \vee \overline{y} \vee q$ and $\overline{q} \vee \overline{z} \vee w$.

What you need to remember

- for each $L \in$ **NP** take TM M deciding L in polynomial time
- define TM M_f computing a reduction to formula φ_x for each input
- due to obliviousness M_f pre-computes head positions and every computation takes time $T(n + p(n))$ steps
- and is a sequence of snapshots $\langle q, 0, 1 \rangle$
- φ has four parts
  - correct input x, u with u being the certificate
  - correct starting snapshot
  - correct halting snapshot
  - how to go from s_i to s_{i+1}
- finally: CNF transformed to 3CNF

Agenda

- Cook-Levin ✓
- SAT demo
- see old friends
  - $0/1-$ILP
  - Indset
  - $3-$Coloring

So 3SAT is intractable?

- if **P** $\neq$ **NP**, no polynomial time algorithm for SAT
- contrapositive: if you find one, you prove **P** $=$ **NP**
- every problem in **NP** solvable by exhaustive search for certificates
- which implies **NP** $\subseteq$ **PSPACE** (try each possible re-using space)

SAT **is easy!**

- well-researched problem
- has its own conference
- 1000s of tools, academic and commercial
- extremely useful for modelling
  - verification
  - planning and scheduling
  - AI
  - games (Sudoku!)
- useful for reductions due to low combinatorial complexity
- `satlive.org`: solvers, jobs, competitions

Demo

- `www.sat4j.org`
- two termination problems from string/term-rewriting
- 10000s of variables, millions of clauses
- solvable in a few seconds!

Agenda

- Cook-Levin ✓
- SAT demo ✓
- see old friends
  - 0/1 − ILP
  - Indset
  - 3 − Coloring

More reductions from 3SAT

We will now describe reductions from 3SAT to

- $0/1-\mathrm{ILP}$: the set of satisfiable sets of integer linear programs with boolean solutions
- Indset $= \{\langle G, k\rangle \mid G$ has independent set of size at least $k\}$
- $3-\mathrm{Coloring} = \{G \mid G$ is 3-colorable$\}$

This establishes **NP**-hardness for all of the problems. Of course, they are easily in **NP** as well, hence complete.

3SAT $\leq_p$ 0/1−ILP

$$(x \vee \overline{y} \vee z) \wedge (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee w) \wedge (\overline{x} \vee y \vee \overline{w})$$

3SAT $\leq_p$ 0/1−ILP

$$(x \vee \overline{y} \vee z) \wedge (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee w) \wedge (\overline{x} \vee y \vee \overline{w})$$

$$\begin{aligned} x + (1-y) + z &\geq 1 \\ x + (1-y) + (1-z) &\geq 1 \\ (1-x) + (1-y) + w &\geq 1 \\ (1-x) + y + (1-w) &\geq 1 \end{aligned}$$

3SAT $\leq_p$ 0/1–ILP

$$(x \vee \overline{y} \vee z) \wedge (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee w) \wedge (\overline{x} \vee y \vee \overline{w})$$

$$\begin{array}{rcl} x + (1-y) + z & \geq & 1 \\ x + (1-y) + (1-z) & \geq & 1 \\ (1-x) + (1-y) + w & \geq & 1 \\ (1-x) + y + (1-w) & \geq & 1 \end{array}$$

- $f(x) = x$
- $f(\overline{x}) = (1-x)$
- $f(u_1 \vee \ldots \vee u_k) = f(u_1) + \ldots + f(u_k) \geq 1$

$3SAT \le_p 0/1{-}ILP$

$$(x \vee \overline{y} \vee z) \wedge (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee w) \wedge (\overline{x} \vee y \vee \overline{w})$$

$$\begin{array}{rcl} x + (1-y) + z & \geq & 1 \\ x + (1-y) + (1-z) & \geq & 1 \\ (1-x) + (1-y) + w & \geq & 1 \\ (1-x) + y + (1-w) & \geq & 1 \end{array}$$

- $f(x) = x$
- $f(\overline{x}) = (1-x)$
- $f(u_1 \vee \ldots \vee u_k) = f(u_1) + \ldots + f(u_k) \geq 1$
- linear reduction
- φ satisfiable iff $f(\varphi)$ has boolean solution

3SAT $\leq_p$ Indset

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$

3SAT $\leq_p$ Indset

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$
- reduce to graph $G = (V, E)$, such that each clause gets a node per satisfying assignment
  - $V = \{C_i^{a_i} \mid a : vars(C_i) \to \{0, 1\}, C_i \text{ holds under assignment } a_i\}$

3SAT $\leq_p$ Indset

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$
- reduce to graph $G = (V, E)$, such that each clause gets a node per satisfying assignment
  - $V = \{C_i^{a_i} \mid a : vars(C_i) \to \{0, 1\}, C_i \text{ holds under assignment } a_i\}$
- edges denote conflicting assignments
  - $E = \{\{C_i^a, C_{i'}^{a'}\} \mid i, i' \in [m], \exists x. a(x) \neq a'(x)\}$

3SAT $\leq_p$ Indset

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$
- reduce to graph $G = (V, E)$, such that each clause gets a node per satisfying assignment
  - $V = \{C_i^{a_i} \mid a : vars(C_i) \rightarrow \{0, 1\}, C_i \text{ holds under assignment } a_i\}$
- edges denote conflicting assignments
  - $E = \{\{C_i^{a}, C_{i'}^{a'}\} \mid i, i' \in [m], \exists x. a(x) \neq a'(x)\}$
- G has $7m$ nodes and $\mathcal{O}(m^2)$ edges and can be computed in polynomial time

3SAT $\leq_p$ Indset

- φ is satisfiable

$\Rightarrow$ exists assignment $a : X \rightarrow \{0, 1\}$ that makes φ true

$\Rightarrow$ a makes every clause true

$\Rightarrow$ $\{C_i^{a|vars(i)} \mid 1 \leq i \leq m\}$ is an independent set of size m

3SAT $\leq_p$ Indset

- φ is satisfiable

$\Rightarrow$ exists assignment $a : X \to \{0, 1\}$ that makes φ true

$\Rightarrow$ a makes every clause true

$\Rightarrow$ $\{C_i^{a|vars(i)} \mid 1 \leq i \leq m\}$ is an independent set of size m

- G has an independent set of size m

$\Rightarrow$ ind. set covers all clauses

$\Rightarrow$ ind. set yields composable, partial assignments per clause

$\Rightarrow$ φ is satisfiable

3SAT $\leq_p$ 3−Coloring

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$

3SAT $\leq_p$ 3−Coloring

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$
- reduce to graph $G = (V, E)$
- V is the union of
  - $X \cup \overline{X}$ to capture assignments
  - special nodes $\{u, v\}$
  - one little house per clause with 5 nodes: $\{v_{ij}, a_i, b_i \mid i \in [m], j \in [3]\}$

3SAT $\leq_p$ 3−Coloring

- given: formula φ with m clauses of form $C_i = u_{i1} \vee u_{i2} \vee u_{i3}$
- reduce to graph $G = (V, E)$
- V is the union of
  - $X \cup \overline{X}$ to capture assignments
  - special nodes $\{u, v\}$
  - one little house per clause with 5 nodes: $\{v_{ij}, a_i, b_i \mid i \in [m], j \in [3]\}$
- E comprised of
  - edge $\{u, v\}$
  - for each literal in each clause, a connection to the assignment graph: $\{\{u_{ij}, v_{ij}\} \mid i \in [m], j \in [3]\}$
  - house edges: $\{\{v, a_i\}, \{v, b_i\}, \{v_{i1}, a_i\}, \{v_{i1}, b_i\}, \{v_{i2}, a_i\}, \{v_{i3}, b_i\}, \{v_{i2}, v_{i3}\} \mid i \in [m]\}$
- G has $2n + 5m + 2$ nodes and $\mathcal{O}(m^2)$ edges and can be computed in polynomial time
- three colors: $\{red, true, false\}$

3SAT $\leq_p$ 3−Coloring

- φ is satisfiable,

$\Rightarrow$ there is an assignment $a : X \to \{0, 1\}$ that makes every clause true

$\Rightarrow$ coloring u red, v false, and x true iff $a(x) = 1$ leads to a correct 3-coloring

3SAT $\leq_p$ 3−Coloring

- φ is satisfiable,

$\Rightarrow$ there is an assignment $a : X \rightarrow \{0, 1\}$ that makes every clause true

$\Rightarrow$ coloring u red, v false, and x true iff $a(x) = 1$ leads to a correct 3-coloring

- G is 3-colorable
- wlog. assume u is red and v is false
- assume there is a clause j such that all literals are colored false

$\Rightarrow$ v_{j2} and v_{j3} are colored true and red

$\Rightarrow$ a_j and b_j are colored true and red

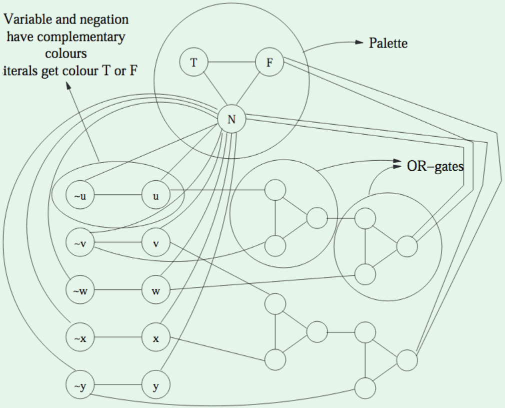
$\Rightarrow$ v_{j1} colored false, which is a contradiction, because it is connected to a false literal

3SAT $\leq_p$ 3−Coloring

Alternatively:

Example

$$\varphi = (u \vee \neg v \vee w) \wedge (v \vee x \vee \neg y)$$



What have you learnt?

- SAT is **NP**-complete
- SAT is practically feasible
- SAT has lots of academic and industrial applications
- SAT can be reduced to independent set, 3-coloring and boolean ILP, which makes those **NP**-hard
- up next: **coNP**, Ladner

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 9, 2019

Lecture 6

coNP

Agenda

- **coNP**
- the importance of **P** vs. **NP** vs. **coNP**
- neither in **P** nor **NP**-complete: Ladner's theorem
- wrap-up Lecture 1-6

coNP

- reminder: $L \subseteq \{0,1\}^* \in$ **coNP** iff $\{0,1\}^* \setminus L \in$ **NP**
- example: $\overline{\text{SAT}}$ contains

coNP

- reminder: $L \subseteq \{0,1\}^* \in$ **coNP** iff $\{0,1\}^* \setminus L \in$ **NP**
- example: $\overline{\text{SAT}}$ contains
  - not well-formed formulas
  - unsatisfiable formulas

coNP

- reminder: $L \subseteq \{0,1\}^* \in$ **coNP** iff $\{0,1\}^* \setminus L \in$ **NP**
- example: $\overline{\text{SAT}}$ contains
  - not well-formed formulas
  - unsatisfiable formulas
- does $\overline{\text{SAT}}$ have polynomial certificates?

coNP

- reminder: $L \subseteq \{0,1\}^* \in$ **coNP** iff $\{0,1\}^* \setminus L \in$ **NP**
- example: $\overline{\text{SAT}}$ contains
  - not well-formed formulas
  - unsatisfiable formulas
- does $\overline{\text{SAT}}$ have polynomial certificates?
- not known: open problem whether **NP** is closed under complement
- note that **P** is closed under complement, compare with NFA vs DFA closure

For all certificates

- like for **NP** there is a characterization in terms of certificates
- for **coNP** it is dual: for all certificates
- $\overline{\text{3SAT}}$: to prove unsatifiability one must check all assignments, for satisfiability only one

For all certificates

- like for **NP** there is a characterization in terms of certificates
- for **coNP** it is dual: for all certificates
- $\overline{\text{3SAT}}$: to prove unsatisfiability one must check all assignments, for satisfiability only one

Theorem (coNP certificates)

A language $L \subseteq \{0,1\}^*$ is in **coNP** *iff there exists a polynomial p and a TM M such that*

$$\forall x \in \{0,1\}^* \; x \in L \Leftrightarrow \forall u \in \{0,1\}^{p(|x|)} \; M(x,u) = 1$$

Completeness

- like for **NP** one can define **coNP**-hardness and completeness
- L is **coNP**-complete iff $L \in$ **coNP** and all problems in **coNP** are polynomial-time Karp-reducible to L
- classical example: Tautology $= \{\varphi \mid$
φ is Boolean formula that is true for every assignment$\}$
- example: $x \vee \overline{x} \in$ Tautology
- proof?

Completeness

- like for **NP** one can define **coNP**-hardness and completeness
- L is **coNP**-complete iff $L \in$ **coNP** and all problems in **coNP** are polynomial-time Karp-reducible to L
- classical example: Tautology $= \{\varphi \mid$ φ is Boolean formula that is true for every assignment$\}$
- example: $x \vee \overline{x} \in$ Tautology
- proof?
  - note that L is **coNP**-complete, if $\overline{L}$ is **NP**-complete

  $\Rightarrow$ $\overline{\text{SAT}}$ is **coNP**-complete

  $\Rightarrow$ Tautology is **coNP**-complete (reduction from $\overline{\text{SAT}}$ by negating formula)

Regular Expression Equivalence

xA regular expression over $\{0, 1\}$ is defined by

$$r ::= 0 \mid 1 \mid rr \mid r|r \mid r \cap r \mid r^*$$

The language defined by r is written $\mathcal{L}(r)$.

Regular Expression Equivalence

xA regular expression over $\{0, 1\}$ is defined by

$$r ::= 0 \mid 1 \mid rr \mid r|r \mid r \cap r \mid r^*$$

The language defined by r is written $\mathcal{L}(r)$.

- let $\varphi = C_1 \wedge \ldots \wedge C_m$ be a Boolean formula in 3CNF over variables $x_1, \ldots, x_n$

Regular Expression Equivalence

xA regular expression over $\{0, 1\}$ is defined by

$$r ::= 0 \mid 1 \mid rr \mid r|r \mid r \cap r \mid r^*$$

The language defined by r is written $\mathcal{L}(r)$.

- let $\varphi = C_1 \wedge \ldots \wedge C_m$ be a Boolean formula in 3CNF over variables $x_1, \ldots, x_n$
- compute from φ a regular expression: $f(\varphi)$=$(\alpha_1|\alpha_2|\ldots|\alpha_m)$

Regular Expression Equivalence

xA regular expression over $\{0,1\}$ is defined by

$$r ::= 0 \mid 1 \mid rr \mid r|r \mid r \cap r \mid r^*$$

The language defined by r is written $\mathcal{L}(r)$.

- let $\varphi = C_1 \wedge \ldots \wedge C_m$ be a Boolean formula in 3CNF over variables $x_1, \ldots, x_n$
- compute from φ a regular expression: $f(\varphi)$=$(\alpha_1|\alpha_2|\ldots|\alpha_m)$
- $\alpha_i = \gamma_{i1} \ldots \gamma_{in}$

Regular Expression Equivalence

xA regular expression over $\{0, 1\}$ is defined by

$$r ::= 0 \mid 1 \mid rr \mid r|r \mid r \cap r \mid r^*$$

The language defined by r is written $\mathcal{L}(r)$.

- let $\varphi = C_1 \wedge \ldots \wedge C_m$ be a Boolean formula in 3CNF over variables $x_1, \ldots, x_n$
- compute from φ a regular expression: $f(\varphi)$=$(\alpha_1|\alpha_2|\ldots|\alpha_m)$
- $\alpha_i = \gamma_{i1} \ldots \gamma_{in}$
- $\gamma_{ij} = \begin{cases} 0 & x_j \in C_i \\ 1 & \overline{x_j} \in C_i \\ (0|1) & \text{otherwise} \end{cases}$

Regular Expression Equivalence

xA regular expression over $\{0, 1\}$ is defined by

$$r ::= 0 \mid 1 \mid rr \mid r|r \mid r \cap r \mid r^*$$

The language defined by r is written $\mathcal{L}(r)$.

- let $\varphi = C_1 \wedge \ldots \wedge C_m$ be a Boolean formula in 3CNF over variables $x_1, \ldots, x_n$
- compute from φ a regular expression: $f(\varphi)$=$(\alpha_1|\alpha_2|\ldots|\alpha_m)$
- $\alpha_i = \gamma_{i1} \ldots \gamma_{in}$
- $\gamma_{ij} = \begin{cases} 0 & x_j \in C_i \\ 1 & \overline{x_j} \in C_i \\ (0|1) & \text{otherwise} \end{cases}$
- example: $(x \vee y \vee \overline{z}) \wedge (\overline{y} \vee z \vee w)$ transformed to $(001(0|1)) \mid (0|1)100)$

Regular Expression Equivalence

xA regular expression over $\{0, 1\}$ is defined by

$$r ::= 0 \mid 1 \mid rr \mid r|r \mid r \cap r \mid r^*$$

The language defined by r is written $\mathcal{L}(r)$.

- let $\varphi = C_1 \wedge \ldots \wedge C_m$ be a Boolean formula in 3CNF over variables $x_1, \ldots, x_n$
- compute from φ a regular expression: $f(\varphi)$=$(\alpha_1|\alpha_2|\ldots|\alpha_m)$
- $\alpha_i = \gamma_{i1} \ldots \gamma_{in}$
- $\gamma_{ij} = \begin{cases} 0 & x_j \in C_i \\ 1 & \overline{x_j} \in C_i \\ (0|1) & \text{otherwise} \end{cases}$
- example: $(x \vee y \vee \overline{z}) \wedge (\overline{y} \vee z \vee w)$ transformed to $(001(0|1)) \mid (0|1)100)$
- observe: φ is unsatisfiable iff $f(\varphi) = \{0, 1\}^n$

Regular expressions and computational complexity

- previous slide establishes: $\overline{\mathrm{3SAT}} \leq_p \mathrm{RegExpEq}_0$
- that is: regular expression equivalence is **coNP**-hard

Regular expressions and computational complexity

- previous slide establishes: $\overline{\text{3SAT}} \leq_p \text{RegExpEq}_0$
- that is: regular expression equivalence is **coNP**-hard
- it is **coNP**-complete for expressions without $*, \cap$
- because one needs to check for all expressions of length n whether they are included (test polynomial by NFA transformation)

Regular expressions and computational complexity

- previous slide establishes: $\overline{\text{3SAT}} \leq_p \text{RegExpEq}_0$
- that is: regular expression equivalence is **coNP**-hard
- it is **coNP**-complete for expressions without $*, \cap$
- because one needs to check for all expressions of length n whether they are included (test polynomial by NFA transformation)
- the problem becomes **PSPACE**-complete when $*$ is added
- the problem becomes **EXP**-complete when $*, \cap$ is added

Agenda

- **coNP** ✓
- the importance of **P** vs. **NP** vs. **coNP**
- neither in **P** nor **NP**-complete: Ladner's theorem
- wrap-up Lecture 1-6

Open and known problems

OPEN

- **P** $=$ **NP**?
- **NP** $=$ **coNP**?

Open and known problems

OPEN

- **P** $=$ **NP**?
- **NP** $=$ **coNP**?

KNOWN

- if an **NP**-complete problem is in **P**, then **P** $=$ **NP**
- **P** $\subseteq$ **coNP** $\cap$ **NP**
- if $L \in$ **coNP** and L **NP**-complete then **NP** $=$ **coNP**
- if **P** $=$ **NP** then **P** $=$ **NP** $=$ **coNP**
- if **NP** $\neq$ **coNP** then **P** $\neq$ **NP**
- if **EXP** $\neq$ **NEXP** then **P** $\neq$ **NP** (equalities scale up, inequalities scale down – by padding)

What if $\mathsf{P} = \mathsf{NP}$?

- one of the most important open problems
- computational utopia
- SAT has polynomial algorithm
- 1000s of other problems, too (due to reductions, completeness)
- finding solutions is as easy as verifying them
- guessing can be done deterministically
- decryption as easy as encryption
- randomization can be de-randomized

What if NP $=$ coNP

Problems have short certificates that don't seem to have any!

- like tautology, unsatisfiability
- like unsatisfiable ILPs
- like regular expression equivalence

How to cope with NP-complete problems?

- ignore (see SAT), it may still work
- modify your problem (2SAT, 2Coloring)
- NP-completeness talks about worst cases and exact solutions
  - $\longrightarrow$ try average cases
  - $\longrightarrow$ try approximations
- randomize
- explore special cases (TSP)

In praise of reductions

- reductions help, when lower bounds are hard to come by
- reductions helped to prove **NP**-completeness for 1000s of natural problems
- in fact, most natural problems (exceptions are Factoring and Iso) are either in **P** or **NP**-complete
- but, unless **P** $=$ **NP**, there exist such problems

Agenda

- **coNP** ✓
- the importance of **P** vs. **NP** vs. **coNP** ✓
- neither in **P** nor **NP**-complete: Ladner's theorem
- wrap-up Lecture 1-6

Ladner's Theorem

P/**NP** intermediate languages exist!

Theorem (Ladner)

If **P** $\neq$ **NP** *then there exists a language* $L \subseteq$ **NP** $\setminus$ **P** *that is not* **NP***-complete.*

Proof

- let $H : \mathbb{N} \to \mathbb{N}$ be a function
- define SAT_H to be

$$\{\varphi 0 1^{n^{H(n)}} \mid \varphi \in \mathsf{SAT}, n = |\varphi|\}$$

Proof

- let $H : \mathbb{N} \to \mathbb{N}$ be a function
- define $\mathrm{SAT_H}$ to be

$$\{\varphi 0 1^{n^{H(n)}} \mid \varphi \in \mathrm{SAT}, n = |\varphi|\}$$

Using the definition of SAT_H one can show

1. $H(n) \in \mathcal{O}(1) \Rightarrow SAT_H \notin$ **P**
2. $\lim_{n\to\infty} H(n) = \infty \Rightarrow SAT_H$ is not **NP**-complete

Proof

- let $H : \mathbb{N} \to \mathbb{N}$ be a function
- define $\mathrm{SAT_H}$ to be

$$\{\varphi 0 1^{n^{H(n)}} \mid \varphi \in \mathrm{SAT}, n = |\varphi|\}$$

Using the definition of SAT_H one can show

1. $H(n) \in \mathcal{O}(1) \Rightarrow SAT_H \notin \mathbf{P}$
2. $\lim_{n\to\infty} H(n) = \infty \Rightarrow SAT_H$ is not **NP**-complete

For $H(n)$ at most a constant, padding is polynomial and the SAT_H is **NP**-complete, hence not in **P**.

Proof

- let $H : \mathbb{N} \to \mathbb{N}$ be a function
- define $\mathrm{SAT_H}$ to be

$$\{\varphi 01^{n^{H(n)}} \mid \varphi \in \mathrm{SAT}, n = |\varphi|\}$$

Using the definition of SAT_H one can show

1. $H(n) \in \mathcal{O}(1) \Rightarrow SAT_H \notin \mathbf{P}$
2. $\lim_{n\to\infty} H(n) = \infty \Rightarrow SAT_H$ is not **NP**-complete

For $H(n)$ at most a constant, padding is polynomial and the SAT_H is **NP**-complete, hence not in **P**.

If SAT_H is **NP**-complete, then there is a reduction from SAT to SAT_H in time $\mathcal{O}(n^i)$ for some constant. For large n it maps SAT instances φ to SAT_H instances $\psi 01^{|\psi|^{H(|\psi|)}}$ of size $|\psi| + |\psi|^{H(|\psi|)} = \mathcal{O}(|\varphi|^i)$. This implies $|\psi| \in o(|\varphi|)$ and by repeated application SAT $\in$ **P**. Contradiction!

Proof

Combine the approaches:

- define the function H and fix SAT_H

Proof

Combine the approaches:

- define the function H and fix SAT_H
- $H(n)$ is
  - the smallest $i < \log \log n$ such that

Proof

Combine the approaches:

- define the function H and fix SAT_H
- $H(n)$ is
  - the smallest $i < \log \log n$ such that
    $\forall x \in \{0,1\}^*$ with $|x| \leq \log n$
    M_i (the i-th TM) outputs $SAT_H(x)$
    within $i|x|^i$ steps

Proof

Combine the approaches:

- define the function H and fix SAT_H
- $H(n)$ is
  - the smallest $i < \log\log n$ such that
    $\forall x \in \{0,1\}^*$ with $|x| \leq \log n$
    M_i (the i-th TM) outputs $SAT_H(x)$
    within $i|x|^i$ steps
  - if no such i exists then $H(n) = \log\log n$
- if $SAT_H(x) \in$ **P**, say computed in kn^k
  then there is $j > k$ such that M_i computes $SAT_H(x)$
  hence for $n > 2^{2^j}$ we have $H(n) \leq j$

Proof

Combine the approaches:

- define the function H and fix SAT_H
- $H(n)$ is
  - the smallest $i < \log\log n$ such that
    $\forall x \in \{0,1\}^*$ with $|x| \leq \log n$
    M_i (the i-th TM) outputs $SAT_H(x)$
    within $i|x|^i$ steps
  - if no such i exists then $H(n) = \log\log n$
- if $SAT_H(x) \in$ **P**, say computed in kn^k
  then there is $j > k$ such that M_i computes $SAT_H(x)$
  hence for $n > 2^{2^j}$ we have $H(n) \leq j$
- H tends to ∞ since $SAT_H(x)$ cannot be computed in **P** and each M_i
  must be wrong on a long enough input

Agenda

- **coNP** ✓
- the importance of **P** vs. **NP** vs. **coNP** ✓
- neither in **P** nor **NP**-complete: Ladner's theorem ✓
- wrap-up Lecture 1-6

What you should know by now

- deterministic TMs capture the inuitive notion of algorithms and computability
- universal TM $\sim$ general-purpose computer or an interpreter
- some problems are not computable aka. undecidable, like the halting problem
- this is proved by diagonalization
- complexity class **P** captures tractable problems
- **P** is robust under TM definition tweaks (tapes, alphabet size, obliviousness, universal simulation)
- non-deterministic TMs can be simulated by TM in exponential time
- **NP** $\sim$ non-det. poly. time $\sim$ polynomially checkable certificates

What you should know by now

- **NP** $\sim$ non-det. poly. time $\sim$ polynomially checkable certificates
- reductions allow to define hardness and completeness of problems
- complete problems are the hardest within a class, if they can be solved efficiently the whole class can
- **NP** complete problems: 3SAT (by Cook-Levin); Indset, $3-$Coloring, ILP (by reduction from 3SAT)
- SAT is practically useful and feasible
- **coNP** complete problems: Tautology, star-free regular expression equivalence
- probably there are problems neither in **P** nor **NP**-complete (Ladner)

What's next?

- space classes
- space and time hierarchy theorems
- generalization of **NP** and **coNP**: polynomial hierarchy
- probabilistic TMs, randomization
- complexity and proofs

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 9, 2019

Lecture 7

Hierarchies

Agenda

- deterministic time hierarchy theorem
- non-deterministic time hierarchy theorem
- space hierarchy theorem
- relation between space and time

Time Hierarchy Theorem

Theorem (Time Hierarchy)

Let $f, g : \mathbb{N} \to \mathbb{N}$ be time-constructible such that $f \cdot \log f \in o(g)$. Then $\mathbf{DTIME}(f(n)) \subset \mathbf{DTIME}(g(n))$.

Time Hierarchy Theorem

Theorem (Time Hierarchy)

Let $f, g : \mathbb{N} \to \mathbb{N}$ be time-constructible such that $f \cdot \log f \in o(g)$. Then $\mathbf{DTIME}(f(n)) \subset \mathbf{DTIME}(g(n))$.

- inclusion is strict
- proof: diagonalization
  - TM D simulates M_x on x for $g(|x|)/\log(|x|)$ steps and flips any answer
  - D runs in $\mathcal{O}(g)$
  - if computable by $E = M_i$ in $\mathcal{O}(f)$ then $D(i) \neq M_i(i) = E(i)$, contradiction

Time Hierarchy Theorem

Theorem (Time Hierarchy)

Let $f, g : \mathbb{N} \to \mathbb{N}$ be time-constructible such that $f \cdot \log f \in o(g)$. Then $\mathbf{DTIME}(f(n)) \subset \mathbf{DTIME}(g(n))$.

- inclusion is strict
- proof: diagonalization
  - TM D simulates M_x on x for $g(|x|)/\log(|x|)$ steps and flips any answer
  - D runs in $\mathcal{O}(g)$
  - if computable by $E = M_i$ in $\mathcal{O}(f)$ then $D(i) \neq M_i(i) = E(i)$, contradiction
- logarithmic factor due to slowdown in universal simulation
- shows that **P** does not collapse to level k
- corollary: $\mathbf{P} \subset \mathbf{EXP}$

Non-deterministic versions

Theorem (Time Hierarchy (non-det))

Let $f, g : \mathbb{N} \to \mathbb{N}$ *be time-constructible such that* $f(n+1) \in o(g(n))$. *Then* $\mathbf{NTIME}(f(n)) \subset \mathbf{NTIME}(g(n))$.

Non-deterministic versions

Theorem (Time Hierarchy (non-det))

Let $f, g : \mathbb{N} \to \mathbb{N}$ *be time-constructible such that* $f(n+1) \in o(g(n))$*. Then* $\mathbf{NTIME}(f(n)) \subset \mathbf{NTIME}(g(n))$.

- inclusion is strict
- proof by lazy diagonalization (see: AB Th. 3.2)
- note: proof of deterministic theorem does not carry over

Space Hierarchy Theorem

Theorem (Space Hierarchy)

Let $f, g : \mathbb{N} \to \mathbb{N}$ be space-constructible such that $f \in o(g)$. Then $\mathbf{SPACE}(f(n)) \subset \mathbf{SPACE}(g(n))$.

Space Hierarchy Theorem

Theorem (Space Hierarchy)

Let $f, g : \mathbb{N} \to \mathbb{N}$ be space-constructible such that $f \in o(g)$. Then **SPACE**$(f(n)) \subset$ **SPACE**$(g(n))$.

- inclusion is strict
- stronger theorem than corresponding time theorem
  - universal TM for space-bounded computation incurs only constant space overhead
  - f, g can be logarithmic too
- proof analogous to deterministic time hierarchy
- corollary: **L** $\subset$ **PSPACE**

Agenda

- deterministic time hierarchy theorem ✓
- non-deterministic time hierarchy theorem ✓
- space hierarchy theorem ✓
- relation between space and time

Relation between time and space

Theorem (Time vs. Space)

Let $s : \mathbb{N} \to \mathbb{N}$ *be space-constructible. Then*

$$\mathbf{DTIME}(s(n)) \subseteq \mathbf{SPACE}(s(n)) \subseteq \mathbf{NSPACE}(s(n)) \subseteq \mathbf{DTIME}(2^{O(s(n))})$$

Relation between time and space

Theorem (Time vs. Space)

Let $s : \mathbb{N} \to \mathbb{N}$ be space-constructible. Then

$$\mathbf{DTIME}(s(n)) \subseteq \mathbf{SPACE}(s(n)) \subseteq \mathbf{NSPACE}(s(n)) \subseteq \mathbf{DTIME}(2^{O(s(n))})$$

- inclusions are non-strict
- first two are obvious
- third inclusion requires notion of configuration graphs
- first inclusion can be strengthened to $\mathbf{DTIME}(s(n)) \subseteq \mathbf{SPACE}(\frac{s(n)}{\log n})$

Configuration Graphs

Let M be a deterministic or non-deterministic TM using $s(n)$ space. Let x be some input.

Configuration Graphs

Let M be a deterministic or non-deterministic TM using $s(n)$ space. Let x be some input.

- this induces a configuration graph $G(M, x)$
- nodes are configuration
  - states
  - content of work tapes
- edges are transitions (steps) that M can take

Properties of configuration graph

- outdegree of $G(M, x)$ is 1 if M is deterministic; 2 if M is non-deterministic
- $G(M, x)$ has at most $|Q| \cdot \Gamma^{c \cdot s(n)}$ nodes (c some constant)
- which is in $2^{\mathcal{O}(s(n))}$
- $G(M, x)$ can be made to have unique source and sink
- acceptance $\sim$ existence of path from source to sink
- which can be checked in time $\mathcal{O}(G(M, x))$ using BFS

Properties of configuration graph

- outdegree of $G(M,x)$ is 1 if M is deterministic; 2 if M is non-deterministic
- $G(M,x)$ has at most $|Q| \cdot \Gamma^{c \cdot s(n)}$ nodes (c some constant)
- which is in $2^{\mathcal{O}(s(n))}$
- $G(M,x)$ can be made to have unique source and sink
- acceptance $\sim$ existence of path from source to sink
- which can be checked in time $\mathcal{O}(G(M,x))$ using BFS

$\Rightarrow$ **NSPACE**$(s(n)) \subseteq$ **DTIME**$(2^{\mathcal{O}(s(n))})$

Properties of configuration graph

- outdegree of $G(M, x)$ is 1 if M is deterministic; 2 if M is non-deterministic
- $G(M, x)$ has at most $|Q| \cdot \Gamma^{c \cdot s(n)}$ nodes (c some constant)
- which is in $2^{\mathcal{O}(s(n))}$
- $G(M, x)$ can be made to have unique source and sink
- acceptance $\sim$ existence of path from source to sink
- which can be checked in time $\mathcal{O}(G(M, x))$ using BFS

$\Rightarrow$ **NSPACE**$(s(n)) \subseteq$ **DTIME**$(2^{\mathcal{O}(s(n))})$

$\Rightarrow$ **DTIME**$(s(n)) \subseteq$ **NTIME**$(s(n)) \subseteq$ **SPACE**$(s(n))$

- configurations include a counter over all possible choices

References

- the proof of Ladner's theorem given here follows AB, Th. 3.3
- nice survey, see `blog.computationalcomplexity.org/media/ladner.pdf`
- original proof of time hierarchy by *Hartmanis and Stearns* On the computational complexity of algorithms in Transactions of the American Mathematical Society 117.
- non-det time hierarchy by *Stephen Cook*: A hierarchy for nondeterministic time complexity in 4th annual ACM Symposium on Theory of Computing.
- stronger result on time vs space using pebble games by *Hopcroft, Paul, and Valiant* On time versus space in Journal of the ACM 24(2):332-337, April 1977.

Summary

- a lot of diagonalization
- Ladner: **NP**-intermediate languages exist
- $f \cdot \log f \in o(g)$ implies $\mathbf{DTIME}(f(n)) \subset \mathbf{DTIME}(g(n))$
- $f \in o(g)$ implies $\mathbf{SPACE}(f(n)) \subset \mathbf{SPACE}(g(n))$
- $\mathbf{DTIME}(f(n)) \subseteq \mathbf{SPACE}(s(n)) \subseteq \mathbf{NSPACE}(s(n)) \subseteq \mathbf{DTIME}(2^{O(s(n))})$
- $\mathbf{P} \subset \mathbf{EXP}$ and $\mathbf{L} \subset \mathbf{PSPACE}$

Next time: **PSPACE**

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 9, 2019

Lecture 8

PSPACE

Agenda

- succinctness
- QBF and GG
- **PSPACE** completeness
- QBF is **PSPACE**-complete
- Savitch's theorem

Succinctness vs Expressiveness

Some intuition:

- $5 \cdot 5$ is more succinct than $5 + 5 + 5 + 5 + 5$

$\Rightarrow$ multiplication allows for more succinct representation of arithmetic expressions

- but it is not more expressive

Succinctness vs Expressiveness

Some intuition:

- $5 \cdot 5$ is more succinct than $5 + 5 + 5 + 5 + 5$

$\Rightarrow$ multiplication allows for more succinct representation of arithmetic expressions

- but it is not more expressive

regular expressions

- regular expressions with squaring are more succinct than without
- example: strings over $\{1\}$ with length divisible by 16
  - $((((00)^2)^2)^2)^*$ versus
  - $(0000000000000000)^*$
- but obviously squaring does not add expressiveness

More succinct means more difficult to handle

Non-deterministic finite automata

- NFAs can be exponentially more succinct than DFAs
- but equally expressive
- example: k-last symbol is 1
- complementation, equivalence are polynomial for DFAs and exponential for NFAs

Succinct Boolean formulas

Consider the following formula where $\psi = x \vee y \vee \overline{z}$

$$
\begin{array}{rl}
 & (x \wedge y \wedge \psi) \\
\wedge & (x \wedge \overline{y} \wedge \psi) \\
\wedge & (\overline{x} \wedge y \wedge \psi) \\
\wedge & (\overline{x} \wedge \overline{y} \wedge \psi)
\end{array}
$$

Succinct Boolean formulas

Consider the following formula where $\psi = x \vee y \vee \overline{z}$

$$\begin{array}{ll} & (x \wedge y \wedge \psi) \\ \wedge & (x \wedge \overline{y} \wedge \psi) \\ \wedge & (\overline{x} \wedge y \wedge \psi) \\ \wedge & (\overline{x} \wedge \overline{y} \wedge \psi) \end{array}$$

Formula is satisfiable iff $\exists z\ \forall x\ \forall y . \psi$ is true, where variables range over $\{0, 1\}$.

Succinct Boolean formulas

Consider the following formula where $\psi = x \vee y \vee \overline{z}$

$$\begin{array}{ll} & (x \wedge y \wedge \psi) \\ \wedge & (x \wedge \overline{y} \wedge \psi) \\ \wedge & (\overline{x} \wedge y \wedge \psi) \\ \wedge & (\overline{x} \wedge \overline{y} \wedge \psi) \end{array}$$

Formula is satisfiable iff $\exists z\ \forall x\ \forall y . \psi$ is true, where variables range over $\{0, 1\}$.

$\Rightarrow$ Quantified Boolean Formulas

Quantified Boolean Formulas

Definition (QBF)

A quantified Boolean formula is a formula of the form

$$Q_1 x_1 Q_2 x_2 \dots Q_n x_n \varphi(x_1, \dots, x_n)$$

- where each $Q_i \in \{\forall, \exists\}$
- each x_i ranges over $\{0, 1\}$
- φ is quantifier-free

Quantified Boolean Formulas

Definition (QBF)

A quantified Boolean formula is a formula of the form

$$Q_1 x_1 Q_2 x_2 \dots Q_n x_n \varphi(x_1, \dots, x_n)$$

- where each $Q_i \in \{\forall, \exists\}$
- each x_i ranges over $\{0, 1\}$
- φ is quantifier-free

- wlog we can assume prenex form
- formulas are closed, i.e. each QBF is true or false
- QBF $= \{\varphi \mid \varphi$ is a true QBF$\}$
- if all $Q_i = \exists$, we obtain SAT as a special case
- if all $Q_i = \forall$, we obtain Tautology as a special case

QBF is in PSPACE

Polynomial space algorithm to decide QBF

qbfsolve(ψ)
  if ψ is quantifier-free
    return evaluation of ψ
  if $\psi = Qx.\psi'$
    if $Q = \exists$
      if qbfsolve($\psi'[x \mapsto 0]$) return true
      if qbfsolve($\psi'[x \mapsto 1]$) return true
    if $Q = \forall$
      b_1 = qbfsolve($\psi'[x \mapsto 0]$)
      b_2 = qbfsolve($\psi'[x \mapsto 1]$)
      return $b_1 \wedge b_2$
return false

QBF is in PSPACE

Polynomial space algorithm to decide QBF

qbfsolve(ψ)
  if ψ is quantifier-free
    return evaluation of ψ
  if $\psi = Qx.\psi'$
    if $Q = \exists$
      if qbfsolve($\psi'[x \mapsto 0]$) return true
      if qbfsolve($\psi'[x \mapsto 1]$) return true
    if $Q = \forall$
      b_1 = qbfsolve($\psi'[x \mapsto 0]$)
      b_2 = qbfsolve($\psi'[x \mapsto 1]$)
      return $b_1 \wedge b_2$
return false

- each recursive call can re-use same space!
- qbsolve uses at most $\mathcal{O}(|\psi|^2)$ space

Generalized Geography

- children's game, where people take turn naming cities
- next city must start with previous city's final letter
- as in München → Nürnberg
- no repetitions
- lost if no more choices left

Generalized Geography

- children's game, where people take turn naming cities
- next city must start with previous city's final letter
- as in München → Nürnberg
- no repetitions
- lost if no more choices left

Formalization

Given a graph and a node, players take turns choosing an unvisited adjacent node until no longer possible.

$\text{GG} = \{\langle G, u \rangle \mid \text{player 1 has winning strategy from node } u \text{ in } G\}$

GG $\in$ PSPACE

and here is the algorithm to prove it:

ggsolve(G, u)
if u has no outgoing edge return false
remove u and its adjacent edges from G to obtain G'
for each u_i adjacent to u
$b_i = \text{ggsolve}(G', u_i)$
return $\bigvee_i \overline{b_i}$

GG $\in$ PSPACE

and here is the algorithm to prove it:

ggsolve(G, u)
  if u has no outgoing edge return false
  remove u and its adjacent edges from G to obtain G'
  for each u_i adjacent to u
    $b_i = \text{ggsolve}(G', u_i)$
  return $\bigvee_i \overline{b_i}$

- stack depth 1 for recursion implies polynomial space
- QBF $\leq_p$ GG

Agenda

- succinctness ✓
- QBF and GG ✓
- **PSPACE** completeness
- QBF is **PSPACE**-complete
- Savitch's theorem

PSPACE-completness

Definition (PSPACE-completeness)

Language L is **PSPACE**-hard if for every $L' \in$ **PSPACE** $L' \leq_p L$. L is **PSPACE**-complete if $L \in$ **PSPACE** and L is **PSPACE**-hard.

QBF is PSPACE-complete

Theorem

QBF *is* **PSPACE***-complete.*

QBF is PSPACE-complete

Theorem

QBF *is* PSPACE*-complete.*

- have already shown that QBF $\in$ **PSPACE**
- need to show that every problem $L \in$ **PSPACE** is polynomial-time reducible to QBF

Proof

- let $L \in$ **PSPACE** arbitrary

Proof

- let $L \in$ **PSPACE** arbitrary
- $L \in$ **SPACE**$(s(n))$ for polynomial s

Proof

- let $L \in$ **PSPACE** arbitrary
- $L \in$ **SPACE**$(s(n))$ for polynomial s
- $m \in \mathcal{O}(s(n))$: bits needed to encode configuration C

Proof

- let $L \in$ **PSPACE** arbitrary
- $L \in$ **SPACE**$(s(n))$ for polynomial s
- $m \in \mathcal{O}(s(n))$: bits needed to encode configuration C
- exists Boolean formula $\varphi_{M,x}$ with size $\mathcal{O}(m)$ such that $\varphi_{M,x}(C, C') = 1$ iff $C, C' \in \{0,1\}^m$ encode adjacent configurations; see Cook-Levin

Proof

- let $L \in$ **PSPACE** arbitrary
- $L \in$ **SPACE**$(s(n))$ for polynomial s
- $m \in \mathcal{O}(s(n))$: bits needed to encode configuration C
- exists Boolean formula $\varphi_{M,x}$ with size $\mathcal{O}(m)$ such that $\varphi_{M,x}(C, C') = 1$ iff $C, C' \in \{0,1\}^m$ encode adjacent configurations; see Cook-Levin
- define QBF ψ such that $\psi(C, C')$ is true iff there is a path in $G(M, x)$ from C to C'

Proof

- let $L \in$ **PSPACE** arbitrary
- $L \in$ **SPACE**$(s(n))$ for polynomial s
- $m \in \mathcal{O}(s(n))$: bits needed to encode configuration C
- exists Boolean formula $\varphi_{M,x}$ with size $\mathcal{O}(m)$ such that $\varphi_{M,x}(C, C') = 1$ iff $C, C' \in \{0, 1\}^m$ encode adjacent configurations; see Cook-Levin
- define QBF ψ such that $\psi(C, C')$ is true iff there is a path in $G(M, x)$ from C to C'
- $\psi(C_{start}, C_{accept})$ is true iff M accepts x

Proof – cont'd

Define ψ inductively!

- $\psi_i(C, C')$: there is a path of length at most 2^i from C to C'

Proof – cont'd

Define ψ inductively!

- $\psi_i(C, C')$: there is a path of length at most 2^i from C to C'
- $\psi = \psi_m$ and $\psi_0 = \varphi_{M,x}$

Proof – cont'd

Define ψ inductively!

- $\psi_i(C, C')$: there is a path of length at most 2^i from C to C'
- $\psi = \psi_m$ and $\psi_0 = \varphi_{M,x}$

$$\psi_i(C, C') = \exists C''.\psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C')$$

Proof – cont'd

Define ψ inductively!

- $\psi_i(C, C')$: there is a path of length at most 2^i from C to C'
- $\psi = \psi_m$ and $\psi_0 = \varphi_{M,x}$

$$\psi_i(C, C') = \exists C''.\psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C')$$

might be exponential size,

Proof – cont'd

Define ψ inductively!

- $\psi_i(C, C')$: there is a path of length at most 2^i from C to C'
- $\psi = \psi_m$ and $\psi_0 = \varphi_{M,x}$

$$\psi_i(C, C') = \exists C''.\psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C')$$

might be exponential size, therefore use equivalent

$$\begin{aligned} \psi_i(C, C') \quad = \quad & \exists C''.\forall D_1.\forall D_2. \\ & ((D_1 = C \wedge D_2 = C'') \vee (D_1 = C'' \wedge D_2 = C')) \\ & \Rightarrow \psi_{i-1}(D_1, D_2) \end{aligned}$$

Size of ψ

$$
\begin{aligned}
\psi_i(C, C') \quad = \quad & \exists C''.\forall D_1.\forall D_2. \\
& ((D_1 = C \wedge D_2 = C'') \vee (D_1 = C'' \wedge D_2 = C')) \\
& \Rightarrow \psi_{i-1}(D_1, D_2)
\end{aligned}
$$

- C'' stands for m variables

$\Rightarrow \quad |\psi_i| = |\psi_{i-1}| + \mathcal{O}(m)$

$\Rightarrow \quad |\psi| \in \mathcal{O}(m^2)$

Observations and consequences

- GG is **PSPACE**-complete
- if **PSPACE** $\neq$ **NP** then QBF and GG have no short certificates
- note: proof does not make use of outdegree of $G(M, x)$

$\Rightarrow$ QBF is **NPSPACE**-complete

$\Rightarrow$ **NPSPACE** $=$ **PSPACE**!

- in fact, the same reasoning can be used to prove a stronger result

Savitch's Theorem

Theorem (Savitch)

For every space-constructible $s : \mathbb{N} \to \mathbb{N}$ *with* $s(n) \geq \log n$
$\mathbf{NSPACE}(s(n)) \subseteq \mathbf{SPACE}(s(n)^2)$.

Proof

Let M be a NDTM accepting L. Let $G(M, x)$ be its configuration graph of size $m \in \mathcal{O}(2^{s(n)})$;
each node is represented using $\log m$ space.

Proof

Let M be a NDTM accepting L. Let $G(M, x)$ be its configuration graph of size $m \in \mathcal{O}(2^{s(n)})$;
each node is represented using $\log m$ space.

M accepts x iff there is a path of length at most m from C_{start} to C_{accept}.

Proof

Let M be a NDTM accepting L. Let $G(M, x)$ be its configuration graph of size $m \in \mathcal{O}(2^{s(n)})$;
each node is represented using $\log m$ space.

M accepts x iff there is a path of length at most m from C_{start} to C_{accept}.

Consider the following algorithm reach(u,v,i) to determine whether there is a path from u to v of length at most 2^i.

- for each node z of M
  - $b_1 = \text{reach}(u, z, i-1)$
  - $b_2 = \text{reach}(z, v, i-1)$
  - return $b_1 \wedge b_2$

Proof

Let M be a NDTM accepting L. Let $G(M, x)$ be its configuration graph of size $m \in \mathcal{O}(2^{s(n)})$;
each node is represented using $\log m$ space.

M accepts x iff there is a path of length at most m from C_{start} to C_{accept}.

Consider the following algorithm reach(u,v,i) to determine whether there is a path from u to v of length at most 2^i.

- for each node z of M
  - $b_1 = \text{reach}(u, z, i-1)$
  - $b_2 = \text{reach}(z, v, i-1)$
  - return $b_1 \wedge b_2$

$\Rightarrow$ $\text{reach}(C_{start}, C_{accept}, m)$ takes space $\mathcal{O}((\log m)^2) = \mathcal{O}(s(n)^2)$

Further Reading

- *L. J. Stockmeyer and A. R. Meyer.* Word problems requiring exponential time. STOC, pages 1-9, 1973
  - contains the original proof of **PSPACE** completeness of QBF
  - **PSPACE**-completeness of NFA equivalence
- regular expression equivalence with squaring is **EXPSPACE**-complete:
  `http://people.csail.mit.edu/meyer/rsq.pdf`
- *Gilbert, Lengauer, Tarjan* The Pebbling Problem is Complete in Polynomial Space. SIAM Journal on Computing, Volume 9, Issue 3, 1980, pages 513-524.
- `http://www.qbflib.org/`
  - tools (solvers)
  - many QBF models from verification, games, planning
  - competitions
- **PSPACE**-completeness of Hex, Atomix, Gobang, Chess
- *W.J.Savitch* Relationship between nondeterministic and deterministic tape classes JCSS, 4, pp 177-192, 1970.

What have we learnt

- succinctness leads to more difficult problems
- **PSPACE**: computable in polynomial space (deterministically)
- **PSPACE**-completeness defined in terms of polynomial Karp reductions
- canonical **PSPACE**-complete problem: QBF generalizes SAT
- other complete problems: generalized geography, chess, Hex, Sokoban, Reversi, NFA equivalence, regular expressions equivalence
- **PSPACE** $\sim$ winning strategies in games rather than short certificates
- **PSPACE** $=$ **NPSPACE**
- Savitch: non-deterministic space can be simulated by deterministic space with quadratic overhead (by path enumeration in configuration graph)

Up next: **NL**

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 22, 2019

Lecture 9

NL

Agenda

- about logarithmic space
- paths . . .
- . . . and the absence thereof
- Immerman-Szelepcsényi and others

What can one do with logarithmic space?

In essence an algorithm can maintain a constant number of

- pointers into the input
  - for instance node identities (graph problems)
  - head positions
- counters up to input length

What can one do with logarithmic space?

In essence an algorithm can maintain a constant number of

- pointers into the input
  - for instance node identities (graph problems)
  - head positions
- counters up to input length

Examples:

- **L**: basic arithmetic
- **NL**: paths in graphs

Technical issues

- space usage refers to work tapes only
- read-only input and write-once output is allowed to use more than $\log n$ cells
- write-once: output head must not move to the left
- logspace reductions (because polynomial time-reductions too powerful)

Logspace reductions

Definition (logspace reduction)

Let $L, L' \subseteq \{0,1\}^*$ be languages. We say that L is logspace-reducible to L', written $L \leq_{log} L'$ if there is a function $f : \{0,1\}^* \to \{0,1\}^*$ computed by a deterministic TM using logarithmic space such that $x \in L \Leftrightarrow f(x) \in L'$ for every $x \in \{0,1\}^*$.

- $\leq_{log}$ is transitive
- $C \in \mathbf{L}$ and $B \leq_{log} C$ implies $B \in \mathbf{L}$

- **NL**-hardness and **NL**-completeness defined in terms of logspace reductions

Logspace reductions

Definition (logspace reduction)

Let $L, L' \subseteq \{0,1\}^*$ be languages. We say that L is logspace-reducible to L', written $L \leq_{log} L'$ if there is a function $f : \{0,1\}^* \to \{0,1\}^*$ computed by a deterministic TM using logarithmic space such that $x \in L \Leftrightarrow f(x) \in L'$ for every $x \in \{0,1\}^*$.

- $\leq_{log}$ is transitive
- $C \in \mathbf{L}$ and $B \leq_{log} C$ implies $B \in \mathbf{L}$
  - Space does not bound time and output size: possibly $|f(w)| \neq \mathcal{O}(\log(|w|))$

- **NL**-hardness and **NL**-completeness defined in terms of logspace reductions

Logspace reductions

Definition (logspace reduction)

Let $L, L' \subseteq \{0,1\}^*$ be languages. We say that L is logspace-reducible to L', written $L \leq_{log} L'$ if there is a function $f : \{0,1\}^* \to \{0,1\}^*$ computed by a deterministic TM using logarithmic space such that $x \in L \Leftrightarrow f(x) \in L'$ for every $x \in \{0,1\}^*$.

- $\leq_{log}$ is transitive
- $C \in \mathbf{L}$ and $B \leq_{log} C$ implies $B \in \mathbf{L}$
  - Space does not bound time and output size: possibly $|f(w)| \neq \mathcal{O}(\log(|w|))$
  - Compute $f(x)$ on demand: store only current symbol and its cell number
- **NL**-hardness and **NL**-completeness defined in terms of logspace reductions

Read-once Certificates

Similar to **NP**, also **NL** has a characterization using certificates

Theorem (read-once certificates)

*$L \subseteq \{0,1\}^*$ is in **NL** iff there exists a det. logspace TM M (verifier) and a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for every $x \in \{0,1\}^*$*

$$x \in L \text{ iff } \exists u \in \{0,1\}^{p(|x|)}.M(x,u) = 1$$

Certificate u is written on an additional read-once input tape of M.

Read-once Certificates

Similar to **NP**, also **NL** has a characterization using certificates

Theorem (read-once certificates)

$L \subseteq \{0,1\}^*$ *is in* **NL** *iff there exists a det. logspace TM* M *(verifier) and a polynomial* $p : \mathbb{N} \to \mathbb{N}$ *such that for every* $x \in \{0,1\}^*$

$$x \in L \text{ iff } \exists u \in \{0,1\}^{p(|x|)}.M(x,u) = 1$$

Certificate u *is written on an additional read-once input tape of* M.

- example: path in a graph is a read-once certificate

$\Rightarrow$ certificate is sequence of choices

$\Leftarrow$ certificate is guessed bit-wise (it cannot be stored)

Agenda

- about logarithmic space ✓
- paths . . .
- . . . and the absence thereof
- Immerman-Szelepcsényi and others

NL is all about paths

Recall the language Path in directed graphs defined as

$$\{\langle G, s, t\rangle \mid \exists \text{a path from } s \text{ to } t \text{ in directed graph } G\}$$

NL is all about paths

Recall the language Path in directed graphs defined as

$$\{\langle G, s, t\rangle \mid \exists \text{a path from } s \text{ to } t \text{ in directed graph } G\}$$

We have seen in Lecture 3 that Path $\in$ **NL** by guessing a path:

- non-deterministic walks on graphs of n nodes
- if there is a path, it has length $\leq n$
- maintain one pointer to current node
- one counter counting up to n

NL is all about paths

Recall the language Path in directed graphs defined as

$$\{\langle G, s, t\rangle \mid \exists \text{a path from } s \text{ to } t \text{ in directed graph } G\}$$

We have seen in Lecture 3 that Path $\in$ **NL** by guessing a path:

- non-deterministic walks on graphs of n nodes
- if there is a path, it has length $\leq n$
- maintain one pointer to current node
- one counter counting up to n

In fact we even have:

Theorem (Path)

Path *is* **NL***-complete.*

Proof

- let $L \in$ **NL** be arbitrary, decided by NDTM M

Proof

- let $L \in$ **NL** be arbitrary, decided by NDTM M
- on input $x \in \{0,1\}^n$ reduction f outputs configuration graph $G(M,x)$ of size $2^{O(\log n)}$ by counting to n

Proof

- let $L \in$ **NL** be arbitrary, decided by NDTM M
- on input $x \in \{0,1\}^n$ reduction f outputs configuration graph $G(M,x)$ of size $2^{O(\log n)}$ by counting to n
- there exists a path from C_{start} to C_{accept} in $G(M,x)$ iff M accepts x
- path itself can be used as read-once certificate

More path problems

- many natural problems correspond to path (reachability) problems
- the word problem for NFAs: $\{\langle A, w\rangle \mid w \text{ is accepted by NFA } A\}$
- cycle detection/connected components in directed graphs
- $\overline{\text{2SAT}} \in$ **NL**

More path problems

- many natural problems correspond to path (reachability) problems
- the word problem for NFAs: $\{\langle A, w\rangle \mid w$ is accepted by NFA $A\}$
- cycle detection/connected components in directed graphs
- $\overline{\text{2SAT}} \in$ **NL**
  - $x \vee y$ equivalent to $\neg x \implies y$ equivalent to $\neg y \implies x$
  - yields an implication graph (computable in logspace)
  - unsatisfiable iff there exists a path $x \to \overline{x} \to x$ in implication graph for variable x

Certificates for absence of paths?

- recall the open problem $\mathbf{NP} = \mathbf{coNP}$?
- equivalent to asking whether unsatisfiability has short certificates
- possibly not

Certificates for absence of paths?

- recall the open problem **NP** $=$ **coNP**?
- equivalent to asking whether unsatisfiability has short certificates
- possibly not

What about absence of paths from s to t in graph G with n nodes named 1,. . . ,n?

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)
- membership in C_i has read-once certificates (paths)

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)
- membership in C_i has read-once certificates (paths)
- non-membership of v in C_i also has read-once certificates if $|C_i|$ is known

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)
- membership in C_i has read-once certificates (paths)
- non-membership of v in C_i also has read-once certificates if $|C_i|$ is known
  1. list all membership certificates for all $u \in C_i$ sorted in ascending order
  2. check validity and sortedness
  3. check that v is not in the list
  4. check that the list has length $|C_i|$

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)
- membership in C_i has read-once certificates (paths)
- non-membership of v in C_i also has read-once certificates if $|C_i|$ is known
    1. list all membership certificates for all $u \in C_i$ sorted in ascending order
    2. check validity and sortedness
    3. check that v is not in the list
    4. check that the list has length $|C_i|$
- non-membership in C_i is known given $|C_{i-1}|$ (checking neighbors in (3) as well)

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)
- membership in C_i has read-once certificates (paths)
- non-membership of v in C_i also has read-once certificates if $|C_i|$ is known
  1. list all membership certificates for all $u \in C_i$ sorted in ascending order
  2. check validity and sortedness
  3. check that v is not in the list
  4. check that the list has length $|C_i|$
- non-membership in C_i is known given $|C_{i-1}|$ (checking neighbors in (3) as well)
- $|C_i| = c$ can be certified given $|C_{i-1}|$ using $C_0 = \{s\}$ as base case

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)
- membership in C_i has read-once certificates (paths)
- non-membership of v in C_i also has read-once certificates if $|C_i|$ is known
  1. list all membership certificates for all $u \in C_i$ sorted in ascending order
  2. check validity and sortedness
  3. check that v is not in the list
  4. check that the list has length $|C_i|$
- non-membership in C_i is known given $|C_{i-1}|$ (checking neighbors in (3) as well)
- $|C_i| = c$ can be certified given $|C_{i-1}|$ using $C_0 = \{s\}$ as base case

Certificate is certificate for non-membership in C_n!

Absence of path has read-once cert.!

- let C_i be the set of nodes reachable from s in at most i steps (bounded reachability)
- membership in C_i has read-once certificates (paths)
- non-membership of v in C_i also has read-once certificates if $|C_i|$ is known
  1. list all membership certificates for all $u \in C_i$ sorted in ascending order
  2. check validity and sortedness
  3. check that v is not in the list
  4. check that the list has length $|C_i|$
- non-membership in C_i is known given $|C_{i-1}|$ (checking neighbors in (3) as well)
- $|C_i| = c$ can be certified given $|C_{i-1}|$ using $C_0 = \{s\}$ as base case

Certificate is certificate for non-membership in C_n!

Its size is polynomial in number of nodes and read-once!

NL algorithm for $\overline{PATH}$

$M =$ "On input $\langle G, s, t\rangle$:

1. Let $c_0 = 1$. ⟦ $A_0 = \{s\}$ has 1 node ⟧
2. For $i = 0$ to $m - 1$: ⟦ compute c_{i+1} from c_i ⟧
3. Let $c_{i+1} = 1$. ⟦ c_{i+1} counts nodes in A_{i+1} ⟧
4. For each node $v \neq s$ in G: ⟦ check if $v \in A_{i+1}$ ⟧
5. Let $d = 0$. ⟦ d re-counts A_i ⟧
6. For each node u in G: ⟦ check if $u \in A_i$ ⟧
7. Nondeterministically either perform or skip these steps:
8. Nondeterministically follow a path of length at most i from s and *reject* if it doesn't end at u.
9. Increment d. ⟦ verified that $u \in A_i$ ⟧
10. If (u, v) is an edge of G, increment c_{i+1} and go to stage 5 with the next v. ⟦ verified that $v \in A_{i+1}$ ⟧
11. If $d \neq c_i$, then *reject*. ⟦ check whether found all A_i ⟧
12. Let $d = 0$. ⟦ c_m now known; d re-counts A_m ⟧
13. For each node u in G: ⟦ check if $u \in A_m$ ⟧
14. Nondeterministically either perform or skip these steps:
15. Nondeterministically follow a path of length at most m from s and *reject* if it doesn't end at u.
16. If $u = t$, then *reject*. ⟦ found path from s to t ⟧
17. Increment d. ⟦ verified that $u \in A_m$ ⟧
18. If $d \neq c_m$, then *reject*. ⟦ check whether found all of A_m ⟧
Otherwise, *accept*."

NL $=$ coNL

We have just argued the existence of polynomial read-once certificates for absence of paths.

Theorem (Immerman-Szelepcsényi)

NL $=$ coNL.

Further Reading

- paths in undirected graphs is in **L**
  - *Omer Reingold* Undirected ST-Connectivity in Log-Space, STOC 2005
  - available from

`http://www.wisdom.weizmann.ac.il/˜reingold/publications/sl.ps`

- an alternative characterization of **NL** by reachability is at the heart of descriptive complexity
  - **NL** is first-order logic plus transitive closure
  - *Neil Immerman*, Descriptive Complexity, Springer 1999.

What have we learnt?

- space classes closed under complement
  - so are context-sensitive language (see exercises)
- analogous results for time complexity unlikely
- space classes beyond logarithmic closed under non-determinism
- **NL** is all about reachability
- $\overline{\text{2SAT}}$ is in **NL** and thus also 2SAT (in fact, hard for **NL**)
- **NL** has polynomial read-once certificates
- logarithmic space $\sim$ constant number of pointers and counters

Up next: the polynomial hierarchy **PH**

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 22, 2019

Lecture 10

The polynomial hierarchy PH

Agenda

- ExactIndset, MinEqDNF, and bounded QBF
- Σ_i^p, Π_i^p, and **PH**
- properties of the polynomial hierarchy
- more examples

Exact independent set

Recall the independent set problem

$$\text{Indset} = \{\langle G, k\rangle \mid G \text{ has an independent set of size } k\}$$

which was shown to be **NP**-complete.

Exact independent set

Recall the independent set problem

$$\text{Indset} = \{\langle G, k\rangle \mid G \text{ has an independent set of size } k\}$$

which was shown to be **NP**-complete.

What about the variation

$$\text{ExactIndset} = \{\langle G, k\rangle \mid \text{the largest independent set of } G \text{ has size } k\}$$

Exact independent set

Recall the independent set problem

$$\text{Indset} = \{\langle G, k\rangle \mid G \text{ has an independent set of size } k\}$$

which was shown to be **NP**-complete.

What about the variation

$$\text{ExactIndset} = \{\langle G, k\rangle \mid \text{the largest independent set of } G \text{ has size } k\}$$

One needs to show

1. there exists an independent set of size k and
2. all other independent set have size at most k

Exact independent set

Recall the independent set problem

$$\text{Indset} = \{\langle G, k\rangle \mid G \text{ has an independent set of size } k\}$$

which was shown to be **NP**-complete.

What about the variation

$$\text{ExactIndset} = \{\langle G, k\rangle \mid \text{the largest independent set of } G \text{ has size } k\}$$

One needs to show

1. there exists an independent set of size k and
2. all other independent set have size at most k

(1) is a $\exists$ certificate (as in **NP**) while (2) is a $\forall$ certificate (as in **coNP**)!

Minimizing Boolean formulas

Let DNF be disjunctive normal form and $\equiv$ denote logic equivalence.

$$\text{MinEqDNF} = \{\langle \varphi, k\rangle \mid \text{there is a DNF formula } \psi \text{ of size at most } k \text{ s.t. } \varphi \equiv \psi\}$$

Minimizing Boolean formulas

Let DNF be disjunctive normal form and $\equiv$ denote logic equivalence.

$$\text{MinEqDNF} = \{\langle \varphi, k \rangle \mid \text{there is a DNF formula } \psi \text{ of size at most } k \text{ s.t. } \varphi \equiv \psi\}$$

What about certificates for membership?

- there exists a formula ψ such that
- for all assignments φ and ψ evaluate to the same

Minimizing Boolean formulas

Let DNF be disjunctive normal form and $\equiv$ denote logic equivalence.

$$\text{MinEqDNF} = \{\langle \varphi, k \rangle \mid \text{there is a DNF formula } \psi \text{ of size at most } k \text{ s.t. } \varphi \equiv \psi\}$$

What about certificates for membership?

- there exists a formula ψ such that
- for all assignments φ and ψ evaluate to the same

What about $\overline{\text{MinEqDNF}}$?

Σ_2^p

Recall the certificate-based definitions of **NP** and **coNP**, where $q : \mathbb{N} \to \mathbb{N}$ is a polynomial, $x \in \{0,1\}^*$ and M is a polynomial-time, det. verifier.

NP $x \in L$ iff $\exists u \in \{0,1\}^{q(|x|)}.\ M(x,u) = 1$

coNP $x \in L$ iff $\forall u \in \{0,1\}^{q(|x|)}.\ M(x,u) = 1$

Σ_2^p

Recall the certificate-based definitions of **NP** and **coNP**, where $q : \mathbb{N} \to \mathbb{N}$ is a polynomial, $x \in \{0,1\}^*$ and M is a polynomial-time, det. verifier.

NP $x \in L$ iff $\exists u \in \{0,1\}^{q(|x|)}.\ M(x,u) = 1$

coNP $x \in L$ iff $\forall u \in \{0,1\}^{q(|x|)}.\ M(x,u) = 1$

ExactIndset and MinEqDNF are in a class defined by

$$x \in L \text{ iff } \exists u \in \{0,1\}^{q(|x|)}. \forall v \in \{0,1\}^{q(|x|)}.\ M(x,u,v) = 1$$

Σ_2^p

Recall the certificate-based definitions of **NP** and **coNP**, where $q : \mathbb{N} \to \mathbb{N}$ is a polynomial, $x \in \{0,1\}^*$ and M is a polynomial-time, det. verifier.

NP $x \in L$ iff $\exists u \in \{0,1\}^{q(|x|)}.\ M(x,u) = 1$

coNP $x \in L$ iff $\forall u \in \{0,1\}^{q(|x|)}.\ M(x,u) = 1$

ExactIndset and MinEqDNF are in a class defined by

$$x \in L \text{ iff } \exists u \in \{0,1\}^{q(|x|)}. \forall v \in \{0,1\}^{q(|x|)}.\ M(x,u,v) = 1$$

This class is called Σ_2^p.

Bounded QBF

Another natural problem within $\mathbf{\Sigma_2^p}$ is QBF with one alternation!

$$\Sigma_2\text{SAT} = \{\exists \vec{u_1} \forall \vec{u_2}.\varphi(\vec{u_1}, \vec{u_2}) \mid \text{formula is true}\}$$

where $\vec{u_i}$ denotes a finite sequence of Boolean variables.

Bounded QBF

Another natural problem within $\mathbf{\Sigma_2^p}$ is QBF with one alternation!

$$\Sigma_2\mathrm{SAT} = \{\exists \vec{u_1} \forall \vec{u_2}.\varphi(\vec{u_1}, \vec{u_2}) \mid \text{formula is true } \}$$

where $\vec{u_i}$ denotes a finite sequence of Boolean variables.

Remarks

- in fact, Σ_2SAT is complete for $\mathbf{\Sigma_2^p}$
- more alternations lead to a whole hierarchy
- all of it is contained in **PSPACE**

Agenda

- ExactIndset, MinEqDNF, and bounded QBF ✓
- Σ_i^p, Π_i^p, and PH
- properties of the polynomial hierarchy
- more examples

Definition

Definition (Polynomial Hierarchy)

For $i \geq 1$, a language $L \subseteq \{0,1\}^*$ is in $\mathbf{\Sigma_i^p}$ if there exists a polynomial-time TM M and a polynomial q such that

$$
\begin{aligned}
& x \in L \\
& \quad \textbf{if and only if} \\
& \exists u_1 \in \{0,1\}^{q(|x|)}. \\
& \forall u_2 \in \{0,1\}^{q(|x|)}. \\
& \dots \\
& Q_i u_i \in \{0,1\}^{q(|x|)}. \\
& \quad M(x, u_1, u_2, \dots, u_i) = 1
\end{aligned}
$$

where Q_i is $\exists$ if i is odd and $\forall$ otherwise.

Definition

Definition (Polynomial Hierarchy)

For $i \geq 1$, a language $L \subseteq \{0,1\}^*$ is in $\mathbf{\Sigma_i^p}$ if there exists a polynomial-time TM M and a polynomial q such that

$$x \in L$$

if and only if

$$\exists u_1 \in \{0,1\}^{q(|x|)}.$$
$$\forall u_2 \in \{0,1\}^{q(|x|)}.$$
$$\ldots$$
$$Q_i u_i \in \{0,1\}^{q(|x|)}.$$
$$M(x, u_1, u_2, \ldots, u_i) = 1$$

where Q_i is $\exists$ if i is odd and $\forall$ otherwise.

- the polynomial hierarchy is the set $\mathbf{PH} = \bigcup_{i \geq 1} \mathbf{\Sigma_i^p}$
- $\mathbf{\Pi_i^p} = \mathbf{co\Sigma_i^p} = \{\overline{L} \mid L \in \mathbf{\Sigma_i^p}\}$

Generalization of NP and coNP

- NP $= \Sigma_1^p$ and coNP $= \Pi_1^p$

Generalization of NP and coNP

- $\mathbf{NP} = \Sigma_1^p$ and $\mathbf{coNP} = \Pi_1^p$
- $\Sigma_i^p \subseteq \Pi_{i+1}^p \subseteq \Sigma_{i+2}^p$

Generalization of NP and coNP

- $\mathbf{NP} = \mathbf{\Sigma_1^p}$ and $\mathbf{coNP} = \mathbf{\Pi_1^p}$
- $\mathbf{\Sigma_i^p} \subseteq \mathbf{\Pi_{i+1}^p} \subseteq \mathbf{\Sigma_{i+2}^p}$
- hence $\mathbf{PH} = \bigcup_{i \geq 1} \mathbf{\Pi_i^p}$
- $\mathbf{PH} \subseteq \mathbf{PSPACE}$

Collapse

It is an open problem whether there is an *i* such that $\Sigma_{i}^{p} = \Sigma_{i+1}^{p}$.

Collapse

It is an open problem whether there is an *i* such that $\mathbf{\Sigma^p_i} = \mathbf{\Sigma^p_{i+1}}$.

This would imply that $\mathbf{\Sigma^p_i} = \mathbf{PH}$: the hierarchy collapses to the *i*-th level.

Collapse

It is an open problem whether there is an *i* such that $\mathbf{\Sigma_i^p} = \mathbf{\Sigma_{i+1}^p}$.

This would imply that $\mathbf{\Sigma_i^p} = \mathbf{PH}$: the hierarchy collapses to the *i*-th level.

Most researchers believe that the hierarchy does not collapse.

Collapse

It is an open problem whether there is an i such that $\Sigma_i^p = \Sigma_{i+1}^p$.

This would imply that $\Sigma_i^p = \mathbf{PH}$: the hierarchy collapses to the i-th level.

Most researchers believe that the hierarchy does not collapse.

Theorem (Collapse)

- *For every $i \geq 1$, if $\Sigma_i^p = \Pi_i^p$ then $\mathbf{PH} = \Sigma_i^p$*
- *If $\mathbf{P} = \mathbf{NP}$ then $\mathbf{PH} = \mathbf{P}$, i.e. the hierarchy collapses to $\mathbf{P}$.*

Completeness

For each level of the hierarchy completeness is defined in terms of polynomial Karp reductions.

Completeness

For each level of the hierarchy completeness is defined in terms of polynomial Karp reductions.

- if there exists a **PH**-complete language, then the hierarchy collapses
- **PH** $\neq$ **PSPACE** unless the hierarchy collapses

Completeness

For each level of the hierarchy completeness is defined in terms of polynomial Karp reductions.

- if there exists a **PH**-complete language, then the hierarchy collapses
- **PH** $\neq$ **PSPACE** unless the hierarchy collapses

Theorem (bounded QBF)

For each $i \geq 1$, Σ_iSAT is $\boldsymbol{\Sigma_i^p}$-complete, where Σ_iSAT is the language of true quantified Boolean formulas of the form

$$\exists \vec{u_1} \forall \vec{u_2} \ldots Q_i \vec{u_i}.\varphi(\vec{u_1}, \vec{u_1}, \ldots, \vec{u_i})$$

Agenda

- ExactIndset, MinEqDNF, and bounded QBF ✓
- Σ_i^p, Π_i^p, and PH ✓
- properties of the polynomial hierarchy ✓
- more examples

Integer Expressions

An integer expression I is defined by the following BNF for binary numbers $\vec{b}$:

$$I ::= \vec{b} \mid I + I \mid I \cup I$$

The language $\mathcal{L}(I) \subseteq \mathbb{N}$ is defined by

- $\mathcal{L}(\vec{b}) = \{n\}$ where n is the natural number represented by $\vec{b}$
- $\mathcal{L}(I_1 + I_2) = \{n_1 + n_2 \mid n_i \in \mathcal{L}(I_i)\}$
- $\mathcal{L}(I_1 \cup I_2) = \mathcal{L}(I_1) \cup \mathcal{L}(I_2)$

Example: $\mathcal{L}(1 + (2 \cup (3 + 4))) = \{3, 8\}$

A set $M \subseteq \mathbb{N}$ is connected if for all $x, z \in M$ and every $x < y < z$ also $y \in M$.

A component of M is a maximal connected subset of M.

Integer Expressions

- membership of a number in the language of an integer expression: **NP**-complete
- integer expression inequivalence: $\mathbf{\Sigma_2^p}$-complete
- *Does* $\mathcal{L}(I)$ *have a component of size at least* k?: $\mathbf{\Sigma_3^p}$-complete

Regular Expressions

Consider regular expressions with union and concatentation only. In addition, we define an interleaving operator on words

$$x_1 x_2 \ldots x_k \mid y_1 y_2 \ldots y_k$$
$$=$$
$$x_1 y_1 x_2 y_2 \ldots x_k y_k$$

where y_i can be strings of arbitrary length.

Regular expression equivalence for star-free expressions with interleaving is Π_2^p-complete.

Context-free languages

Consider context-free grammars defining unary languages.

- $\{\langle G_1, G_2 \rangle \mid \mathcal{L}(G_1) \neq \mathcal{L}(G_2)\}$ is $\mathbf{\Sigma_2^p}$-complete
- note that for non-unary languages this problem is undecidable

Further Reading

Survey on complete problems for various levels of the hierarchy:

- *Schaefer and Umans* Completeness in the Polynomial-Time Hierarchy — A Compendium

What have we learnt?

- the polynomial hierarchy is a natural generalization of **NP** and **coNP**
- bounded alternation QBFs are complete problems for each level of the hierarchy
- in the limit – unbounded alternations – the hierarchy approaches **PSPACE**
- the hierarchy is widely believed not to collapse to any level

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 22, 2019

Lecture 10–Part II

PH & co.

Agenda

- oracles
- oracles and **PH**
- relativization and **P** vs. **NP**
- alternation and **PH**

Minimizing Boolean formulas

Let DNF be disjunctive normal form and $\equiv$ denote logic equivalence.

$$\text{MinEqDNF} = \{\langle \varphi, k \rangle \mid \text{there is a DNF formula } \psi \text{ of size at most } k \text{ s.t. } \varphi \equiv \psi\}$$

Minimizing Boolean formulas

Let DNF be disjunctive normal form and $\equiv$ denote logic equivalence.

$$\text{MinEqDNF} = \{\langle \varphi, k \rangle \mid \text{there is a DNF formula } \psi \text{ of size at most } k \text{ s.t. } \varphi \equiv \psi\}$$

Certificate for membership:

- there exists a formula ψ such that
- for all assignments φ and ψ evaluate to the same

Minimizing Boolean formulas

Let DNF be disjunctive normal form and $\equiv$ denote logic equivalence.

$$\text{MinEqDNF} = \{\langle \varphi, k \rangle \mid \text{there is a DNF formula } \psi \text{ of size at most } k \text{ s.t. } \varphi \equiv \psi\}$$

Certificate for membership:

- there exists a formula ψ such that
- for all assignments φ and ψ evaluate to the same

Thus MinEqDNF $\in \mathbf{\Sigma_2^p}$.

Minimizing Boolean formulas

Let DNF be disjunctive normal form and $\equiv$ denote logic equivalence.

$$\text{MinEqDNF} = \{\langle \varphi, k \rangle \mid \text{there is a DNF formula } \psi \text{ of size at most } k \text{ s.t. } \varphi \equiv \psi\}$$

Certificate for membership:

- there exists a formula ψ such that
- for all assignments φ and ψ evaluate to the same

Thus MinEqDNF $\in \Sigma_2^p$.

What if we can check equivalence of formulae for free?

Oracle

Definition

An oracle is a language A.
An oracle Turing machine M^A is a Turing machine that

1. has an extra *oracle* tape, and
2. can ask whther the string currently written on the oracle tape belongs to A and in a *single* computation step gets the answer.

$\mathbf{P}^A$ is a class of languages decidable by a polynomial-time oracle Turing machine with an oracle A; similarly $\mathbf{NP}^A$ etc.

Examples

- MinEqDNF $\in$ $\mathbf{NP}^{\mathrm{SAT}}$

Examples

- MinEqDNF $\in$ $\mathbf{NP}^{\mathrm{SAT}}$
- $\mathbf{NP} \subseteq \mathbf{P}^{\mathrm{SAT}}$
- $\mathbf{coNP} \subseteq \mathbf{P}^{\mathrm{SAT}}$ since $\mathbf{P}$ and $\mathbf{P}^{\mathrm{SAT}}$ are deterministic classes and thus closed under complement

Examples

- MinEqDNF $\in$ $\mathbf{NP}^{\mathrm{SAT}}$
- $\mathbf{NP} \subseteq \mathbf{P}^{\mathrm{SAT}}$
- $\mathbf{coNP} \subseteq \mathbf{P}^{\mathrm{SAT}}$ since $\mathbf{P}$ and $\mathbf{P}^{\mathrm{SAT}}$ are deterministic classes and thus closed under complement
- We often write classes instead of the complete languages, e.g., $\mathbf{P}^{\mathbf{NP}} = \mathbf{P}^{\mathrm{SAT}} = \mathbf{P}^{\mathbf{coNP}}$

Oracles and PH

Recall that

$$\Sigma_i\mathrm{SAT} = \{\exists \vec{u_1} \forall \vec{u_2} \cdots Q \vec{u_i}.\varphi(\vec{u_1}, \ldots, \vec{u_i}) \mid \text{formula is true}\ \}$$

is $\mathbf{\Sigma_i^p}$-complete.

Oracles and PH

Recall that

$$\Sigma_i\mathrm{SAT} = \{\exists \vec{u_1}\,\forall \vec{u_2} \cdots Q\vec{u_i}.\varphi(\vec{u_1}, \ldots, \vec{u_i}) \mid \text{formula is true}\,\}$$

is $\mathbf{\Sigma_i^p}$-complete.

Theorem

For every i, $\mathbf{\Sigma_i^p} = \mathbf{NP}^{\Sigma_{i-1}\mathrm{SAT}} = \mathbf{NP}^{\mathbf{\Sigma_{i-1}^p}}$.

e.g. $\mathbf{\Sigma_3^p} = \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}}$

Oracles and PH

Recall that

$$\Sigma_i\mathrm{SAT} = \{\exists \vec{u_1} \forall \vec{u_2} \cdots Q\vec{u_i}.\varphi(\vec{u_1}, \ldots, \vec{u_i}) \mid \text{formula is true}\}$$

is $\boldsymbol{\Sigma_i^p}$-complete.

Theorem

For every i, $\boldsymbol{\Sigma_i^p} = \mathbf{NP}^{\Sigma_{i-1}\mathrm{SAT}} = \mathbf{NP}^{\boldsymbol{\Sigma_{i-1}^p}}$.

e.g. $\boldsymbol{\Sigma_3^p} = \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}}$

Proof

$\subseteq$: easy

$\supseteq$ (here for i=2, i.e. $\boldsymbol{\Sigma_2^p} \supseteq \mathbf{NP}^{\mathrm{SAT}}$): Let φ_i denote the ith query

$x \in L \iff \exists c_1, \ldots, c_m, a_1, \ldots, a_k, u_1, \ldots, u_k \forall v_1, \ldots, v_k$ such that

TM accepts x using choices $c_1, \ldots, c_m$ and answers $a_1, \ldots, a_k$ AND

$\forall i \in [k]$ if $a_i = 1$ then $\varphi_i(u_i) = 1$ AND

$\forall i \in [k]$ if $a_i = 0$ then $\varphi_i(v_i) = 0$

Relativization and limits of diagonalization

- Diagonalization is based on simulation.
- Simulation-based proofs about TMs can be copied for oracle TMs.

Relativization and limits of diagonalization

- Diagonalization is based on simulation.
- Simulation-based proofs about TMs can be copied for oracle TMs.
- If we can prove $\mathbf{P} = \mathbf{NP}$ using only simulation, we can also prove $\mathbf{P}^A = \mathbf{NP}^A$ for all A.
- If we can prove $\mathbf{P} \neq \mathbf{NP}$ using only simulation, we can also prove $\mathbf{P}^A \neq \mathbf{NP}^A$ for all A.

Relativization and limits of diagonalization

- Diagonalization is based on simulation.
- Simulation-based proofs about TMs can be copied for oracle TMs.
- If we can prove $\mathbf{P} = \mathbf{NP}$ using only simulation, we can also prove $\mathbf{P}^A = \mathbf{NP}^A$ for all A.
- If we can prove $\mathbf{P} \neq \mathbf{NP}$ using only simulation, we can also prove $\mathbf{P}^A \neq \mathbf{NP}^A$ for all A.
- But there exist oracles X and Y:
  - $\mathbf{P}^X \neq \mathbf{NP}^X$ (See Sipser p.378)
  - $\mathbf{P}^Y = \mathbf{NP}^Y$ (Proof: $\mathbf{NP}^{\text{QBF}} \subseteq \mathbf{NPSPACE} \subseteq \mathbf{PSPACE} \subseteq \mathbf{P}^{\text{QBF}}$)

Relativization and limits of diagonalization

- Diagonalization is based on simulation.
- Simulation-based proofs about TMs can be copied for oracle TMs.
- If we can prove $\mathbf{P} = \mathbf{NP}$ using only simulation, we can also prove $\mathbf{P}^A = \mathbf{NP}^A$ for all A.
- If we can prove $\mathbf{P} \neq \mathbf{NP}$ using only simulation, we can also prove $\mathbf{P}^A \neq \mathbf{NP}^A$ for all A.
- But there exist oracles X and Y:
  - $\mathbf{P}^X \neq \mathbf{NP}^X$ (See Sipser p.378)
  - $\mathbf{P}^Y = \mathbf{NP}^Y$ (Proof: $\mathbf{NP}^{\text{QBF}} \subseteq \mathbf{NPSPACE} \subseteq \mathbf{PSPACE} \subseteq \mathbf{P}^{\text{QBF}}$)
- Diagonalization has its limits!
  It is not sufficent to simulate computation,
  we must analyze them $\rightarrow$ e.g. cicuit complexity.

Agenda

- oracles ✓
- oracles and **PH** ✓
- relativization and **P** vs. **NP** ✓
- alternation and **PH**

Alternation

Recall that

- Σ_2SAT $= \{\exists\vec{u_1}\forall\vec{u_2}.\varphi(\vec{u_1},\vec{u_2})$ | formula is true $\}$ is **NP**$^{\textbf{coNP}}$-complete
- SAT $= \{\exists\vec{u_1}.\varphi(\vec{u_1})$ | formula is true $\}$ is **NP**-complete
- VAL $= \{\forall\vec{u_1}.\varphi(\vec{u_1})$ | formula is true $\}$ is **coNP**-complete

- $\exists \sim$ existential certificate $\sim$ there is an accepting computation
- $\forall \sim$ universal certificate $\sim$ all computations are accepting

Alternation

Definition

An alternating Turing machine is a Turing machine where

- states are partitioned into existential (denoted $\exists$ or $\vee$) and universal (denoted $\forall$ or $\wedge$),
- configurations are labelled by the type of the current state,
- a configuration in the computation tree is accepting iff
  - it is $\exists$ and some of its successors is accepting,
  - it is $\forall$ and all its successors are accepting.

We define **ATIME**, **ASPACE**, **AP**, **APSPACE** etc. accordingly.

Alternation and PH

Let $\Sigma_i\mathbf{P}$ denote the set of languages decidable by ATM

- running in polynomial time,
- with initial state being existential, and
- such that on every run there are at most i maximal blocks of existential and of universal configurations.

Theorem

For all i, $\mathbf{\Sigma_i^p} = \Sigma_i\mathbf{P}$.

Power of alternation

Theorem

For $f(n) \geq n$, *we have*
$\mathbf{ATIME}(f(n)) \subseteq \mathbf{SPACE}(f(n)) \subseteq \mathbf{ATIME}(f^2(n))$.

For $f(n) \geq \log n$, *we have*
$\mathbf{ASPACE}(f(n)) = \mathbf{TIME}(2^{\mathcal{O}(f(n))})$.

Power of alternation

Theorem

For $f(n) \geq n$, we have
$\mathbf{ATIME}(f(n)) \subseteq \mathbf{SPACE}(f(n)) \subseteq \mathbf{ATIME}(f^2(n))$.

For $f(n) \geq \log n$, we have
$\mathbf{ASPACE}(f(n)) = \mathbf{TIME}(2^{\mathcal{O}(f(n))})$.

Corollary:
$\mathbf{L} \subseteq \mathbf{AL} = \mathbf{P} \subseteq \mathbf{AP} = \mathbf{PSPACE} \subseteq \mathbf{APSPACE} = \mathbf{EXP} \subseteq \mathbf{AEXP} \cdots$

Power of alternation: Proofs

- **ATIME**$(f(n)) \subseteq$ **SPACE**$(f(n))$

Power of alternation: Proofs

- **ATIME**$(f(n)) \subseteq$ **SPACE**$(f(n))$
  DFS on the tree + remember only decisions (not configurations)
- **SPACE**$(f(n)) \subseteq$ **ATIME**$(f^2(n))$

Power of alternation: Proofs

- **ATIME**$(f(n)) \subseteq$ **SPACE**$(f(n))$
  DFS on the tree + remember only decisions (not configurations)
- **SPACE**$(f(n)) \subseteq$ **ATIME**$(f^2(n))$
  like Savitch's theorem
- **ASPACE**$(f(n)) \subseteq$ **TIME**$(2^{\mathcal{O}(f(n))})$

Power of alternation: Proofs

- $\mathsf{ATIME}(f(n)) \subseteq \mathsf{SPACE}(f(n))$
  DFS on the tree + remember only decisions (not configurations)
- $\mathsf{SPACE}(f(n)) \subseteq \mathsf{ATIME}(f^2(n))$
  like Savitch's theorem
- $\mathsf{ASPACE}(f(n)) \subseteq \mathsf{TIME}(2^{\mathcal{O}(f(n))})$
  configuration graph + "attractor" construction
- $\mathsf{ASPACE}(f(n)) \supseteq \mathsf{TIME}(2^{\mathcal{O}(f(n))})$

Power of alternation: Proofs

- $\mathbf{ATIME}(f(n)) \subseteq \mathbf{SPACE}(f(n))$
  DFS on the tree + remember only decisions (not configurations)
- $\mathbf{SPACE}(f(n)) \subseteq \mathbf{ATIME}(f^2(n))$
  like Savitch's theorem
- $\mathbf{ASPACE}(f(n)) \subseteq \mathbf{TIME}(2^{O(f(n))})$
  configuration graph + "attractor" construction
- $\mathbf{ASPACE}(f(n)) \supseteq \mathbf{TIME}(2^{O(f(n))})$
  guess and check the tableaux of the computation
  (+ halting state on the left)

Further Reading

Alternation

- for a survey on alternation see *Chandra, Kozen, Stockmeyer* Alternation in Journal of the ACM 28(1), 1981.
- `http://portal.acm.org/citation.cfm?id=322243`

What have we learnt?

- the polynomial hierarchy can be defined in terms of certificates, recursively by oracles, or by bounded alternation
- diagonalization/simulation proof techniques have their limits
- alternation seems to add power:
  it moves us to the “next higher” class

Up next: time/space tradeoffs, **TISP**(f, g)

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 22, 2019

Lecture 11

Lower Bounds for SAT

Agenda

- big picture
- **TISP**
- lower bound for satisfiability

What is complexity all about?

- formalize the notion of computation
- resource consumption of computations
- depending on input size
- in the worst-case
- computing precise solutions

What is complexity all about?

- formalize the notion of computation
- resource consumption of computations
- depending on input size
- in the worst-case
- computing precise solutions

complexity classes

separation

lower bounds

Satisfiability

We cannot rule out that SAT could be solved in

- linear time or
- logarithmic space

Satisfiability

We cannot rule out that SAT could be solved in

- linear time or
- logarithmic space

Situation similar for many **NP**-complete problems.

What about restricting time and space simultaneously?

TISP

Definition (TISP)

Let $S, T : \mathbb{N} \to \mathbb{N}$ be constructible functions. A language $L \subseteq \{0,1\}^*$ is in the complexity class **TISP**$(T(n), S(n))$ if there exists a TM M deciding L in time $T(n)$ and space $S(n)$.

Note: **TISP**$(T(n), S(n)) \neq$ **DTIME**$(T(n)) \cap$ **SPACE**$(S(n))$

Agenda

- big picture ✓
- **TISP** ✓
- lower bound for satisfiability
- big picture

Lower Bound for Satisfiability

Theorem

SAT $\notin$ **TISP**$(n^{1.1}, n^{0.1})$.

In order to decide SAT we need

- either more than linear time
- or more than logarithmic space
- due to completeness this translates to any other problem in **NP**
- stronger results known (see further reading)

Proof – Big Picture

Proof is by contradiction. So assume

0. SAT $\in$ **TISP**$(n^{1.1}, n^{0.1})$
1. This implies **NTIME**$(n) \subseteq$ **TISP**$(n^{1.2}, n^{0.2})$
2. This implies **NTIME**$(n^{10}) \subseteq$ **TISP**(n^{12}, n^{02}) by padding
3. 1. also implies **NTIME**$(n) \subseteq$ **DTIME**$(n^{1.2})$
4. which implies $\mathbf{\Sigma_2}$**TIME**$(n^{8}) \subseteq$ **NTIME**$(n^{9.6})$
5. separately we can show **TISP**$(n^{12}, n^{2}) \subseteq$ $\mathbf{\Sigma_2}$**TIME**(n^{8})
6. (2,4,5) together establish **NTIME**$(n^{10}) \subseteq$ **NTIME**$(n^{9.6})$
   contradicting the non-deterministic time hierarchy theorem

Proof – Part 1

- can be proven by careful observation of the Cook-Levin reduction.
- problem decided in $\mathbf{NTIME}(T(n))$ can be formulated as satisfiability problem of size $T(n)\log(T(n))$
- every output bit of reduction computable in polylogarithmic time and space
- hence if $\mathrm{SAT} \in \mathbf{TISP}(n^{1.1}, n^{0.1})$ then $\mathbf{NTIME}(n) \subseteq \mathbf{TISP}(n^{1.2}, n^{0.2})$

Proof – Part 2 (padding)

- let $L \in \textbf{NTIME}(n^{10})$

Proof – Part 2 (padding)

- let $L \in \textbf{NTIME}(n^{10})$
- define $L' = \{x1^{|x|^{10}} \mid x \in L\}$

Proof – Part 2 (padding)

- let $L \in \mathbf{NTIME}(n^{10})$
- define $L' = \{x1^{|x|^{10}} \mid x \in L\}$
- then $L' \in \mathbf{NTIME}(n)$

Proof – Part 2 (padding)

- let $L \in \mathbf{NTIME}(n^{10})$
- define $L' = \{x1^{|x|^{10}} \mid x \in L\}$
- then $L' \in \mathbf{NTIME}(n)$
- by part 1 of proof: $L' \in \mathbf{TISP}(n^{1.2}, n^{0.2})$
- thus $L \in \mathbf{TISP}(n^{12}, n^2)$

Proof – Part 3

By definition of **TISP**.

Proof – Part 4

Definition

A language L is in $\mathbf{\Sigma_2 TIME}(n^8)$ iff there exists a TM M running in time $\mathcal{O}(n^8)$ and constants c, d such that

$$x \in L \text{ iff } \exists u \in \{0,1\}^{c|x|^8}.\ \forall v \in \{0,1\}^{d|x|^8}.\ M(x,u,v) = 1$$

Proof – Part 4

Definition

A language L is in $\mathbf{\Sigma_2 TIME}(n^8)$ iff there exists a TM M running in time $\mathcal{O}(n^8)$ and constants c, d such that

$$x \in L \text{ iff } \exists u \in \{0,1\}^{c|x|^8}.\ \forall v \in \{0,1\}^{d|x|^8}.\ M(x,u,v) = 1$$

- let $L \in \mathbf{\Sigma_2 TIME}(n^8)$

Proof – Part 4

Definition

A language L is in $\mathbf{\Sigma_2 TIME}(n^8)$ iff there exists a TM M running in time $\mathcal{O}(n^8)$ and constants c, d such that

$$x \in L \text{ iff } \exists u \in \{0,1\}^{c|x|^8}.\ \forall v \in \{0,1\}^{d|x|^8}.\ M(x,u,v) = 1$$

- let $L \in \mathbf{\Sigma_2 TIME}(n^8)$
- define $L' = \{(x,u) \mid \forall v \in \{0,1\}^{d|x|^8}.\ M(x,u,v) = 1\}$

Proof – Part 4

Definition

A language L is in $\mathbf{\Sigma_2 TIME}(n^8)$ iff there exists a TM M running in time $\mathcal{O}(n^8)$ and constants c, d such that

$$x \in L \text{ iff } \exists u \in \{0,1\}^{c|x|^8}.\ \forall v \in \{0,1\}^{d|x|^8}.\ M(x,u,v) = 1$$

- let $L \in \mathbf{\Sigma_2 TIME}(n^8)$
- define $L' = \{(x,u) \mid \forall v \in \{0,1\}^{d|x|^8}.\ M(x,u,v) = 1\}$
- hence $\overline{L'} \in \mathbf{NTIME}(n^8)$

Proof – Part 4

Definition

A language L is in $\mathbf{\Sigma_2 TIME}(n^8)$ iff there exists a TM M running in time $\mathcal{O}(n^8)$ and constants c, d such that

$$x \in L \text{ iff } \exists u \in \{0,1\}^{c|x|^8}.\ \forall v \in \{0,1\}^{d|x|^8}.\ M(x,u,v) = 1$$

- let $L \in \mathbf{\Sigma_2 TIME}(n^8)$
- define $L' = \{(x,u) \mid \forall v \in \{0,1\}^{d|x|^8}.\ M(x,u,v) = 1\}$
- hence $\overline{L'} \in \mathbf{NTIME}(n^8)$
- by premise we obtain $\overline{L'} \in \mathbf{DTIME}(n^{1.2*8})$ and also L'

Proof – Part 4

Definition

A language L is in $\mathbf{\Sigma_2 TIME}(n^8)$ iff there exists a TM M running in time $\mathcal{O}(n^8)$ and constants c, d such that

$$x \in L \text{ iff } \exists u \in \{0,1\}^{c|x|^8}.\ \forall v \in \{0,1\}^{d|x|^8}.\ M(x,u,v) = 1$$

- let $L \in \mathbf{\Sigma_2 TIME}(n^8)$
- define $L' = \{(x,u) \mid \forall v \in \{0,1\}^{d|x|^8}.\ M(x,u,v) = 1\}$
- hence $\overline{L'} \in \mathbf{NTIME}(n^8)$
- by premise we obtain $\overline{L'} \in \mathbf{DTIME}(n^{1.2*8})$ and also L'
- since $L = \{x \mid \exists u \in \{0,1\}^{c|x|^8}, (x,u) \in L'\}$ we obtain $L \in \mathbf{NTIME}(n^{9.6})$

Proof – Part 5

- let $L \in \textbf{TISP}(n^{12}, n^2)$

Proof – Part 5

- let $L \in \textbf{TISP}(n^{12}, n^2)$
- then there exists a TM M such that $x \in \{0,1\}^n$ is accepted iff there is a path of length n^{12} in the configuration graph from C_{start} to C_{accept}

Proof – Part 5

- let $L \in \mathbf{TISP}(n^{12}, n^2)$
- then there exists a TM M such that $x \in \{0,1\}^n$ is accepted iff there is a path of length n^{12} in the configuration graph from C_{start} to C_{accept}
- where each configuration takes space $\mathcal{O}(n^2)$
- this is the case iff
    - there exist configurations $C_0, \ldots, C_{n^6}$ such that
    - $C_0 = C_{start}$, $C_{n^6} = C_{accept}$
    - for all $1 \leq i \leq n^6$ C_{i+1} is reachable from C_i in n^6 steps

Proof – Part 5

- let $L \in \mathbf{TISP}(n^{12}, n^2)$
- then there exists a TM M such that $x \in \{0,1\}^n$ is accepted iff there is a path of length n^{12} in the configuration graph from C_{start} to C_{accept}
- where each configuration takes space $\mathcal{O}(n^2)$
- this is the case iff
  - there exist configurations $C_0, \ldots, C_{n^6}$ such that
  - $C_0 = C_{start}$, $C_{n^6} = C_{accept}$
  - for all $1 \leq i \leq n^6$ C_{i+1} is reachable from C_i in n^6 steps
- this implies $L \in \mathbf{\Sigma_2 TIME}(n^8)$
- which can be equivalently characterized using alternating TMs

Agenda

- big picture ✓
- **TISP** ✓
- lower bound for satisfiability ✓

Summary of today's result

- SAT cannot be decided in linear time and, simultaneously, logarithmic space
- neither can any other problem in **NP**
- lower bounds are hard
- nice combination of proof techniques
    - padding
    - reductions
    - splitting paths in the configuration graph
    - diagonalization

Further Reading

- AB, Theorem 5.11
- original lower bound by *Fortnow*, Time-space tradeoffs for satisfiability, CCC 1997.
- current record: SAT $\notin$ **TISP**$(n^c, n^{o(1)})$ for any $c < 2\cos(\pi/7)$
- by *R. Williams* Time-space tradeoffs for counting NP solutions modulo integers, CCC 2007.

Complexity Theory

Jan Křetínský

Based on slides by Michael Luttenberger

Technical University of Munich

Summer 2019

May 28, 2019

Lecture 12–13

Randomization and Polynomial Time

“Realistic computation somewhere between P and NP”

Agenda

- Motivation: From **NP** to a more realistic class by randomization
  - Choosing the certificate at random
  - Error reduction by rerunning
- Randomized poly-time with one-sided error: **RP**, **coRP**, **ZPP**
- Power of randomization with two-sided error: **PP**, **BPP**

Recap **P**

Definition (P)

For every $L \subseteq \{0,1\}^*$:
$L \in \mathbf{P}$ if there is a poly-time TM M such that for every $x \in \{0,1\}^*$:

$$x \in L \Leftrightarrow M(x) = 1.$$

- "poly-time TM M":
  - M deterministic
  - M outputs $\{0,1\}$
  - There is a polynomial $T(n)$ s.t. M halts on every x within $T(|x|)$ steps.
- Problems in **P** are deemed "tractable".

Recap NP

Theorem (Certificates)

For every $L \subseteq \{0,1\}^*$:
*$L \in$ **NP** if and only if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a poly-time TM M such that for every $x \in \{0,1\}^*$*

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} \ : \ M(x,u) = 1$$

- Certificate u: satisfying assignment, independent set, 3-coloring, etc.
- **NP** captures the class of possibly (not) tractable computations:
  - Don't know how to compute u in poly-time, but
  - if there is a u, then $|u|$ is polynomial in $|x|$, and
  - we can check in poly-time if a u is a certificate/solution.

Recap NP

Theorem (Certificates)

For every $L \subseteq \{0,1\}^*$:
*$L \in$ **NP** if and only if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a poly-time TM M such that for every $x \in \{0,1\}^*$*

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} \ : \ M(x,u) = 1$$

- Certificate u: satisfying assignment, independent set, 3-coloring, etc.
- **NP** captures the class of possibly (not) tractable computations:
  - Don't know how to compute u in poly-time, but
  - if there is a u, then $|u|$ is polynomial in $|x|$, and
  - we can check in poly-time if a u is a certificate/solution.
- NDTMs can check all $2^{p(|x|)}$ possible us in parallel.
- Seems unrealistic. Common conjecture: **P** $\neq$ **NP**.

Recap **NP**

Theorem (Certificates)

For every $L \subseteq \{0,1\}^*$*:*
$L \in \mathbf{NP}$ *if and only if there exists a polynomial* $p : \mathbb{N} \to \mathbb{N}$ *and a poly-time TM* M *such that for every* $x \in \{0,1\}^*$

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} \ : \ M(x,u) = 1$$

- Certificate u: satisfying assignment, independent set, 3-coloring, etc.
- **NP** captures the class of possibly (not) tractable computations:
    - Don't know how to compute u in poly-time, but
    - if there is a u, then $|u|$ is polynomial in $|x|$, and
    - we can check in poly-time if a u is a certificate/solution.
- NDTMs can check all $2^{p(|x|)}$ possible us in parallel.
- Seems unrealistic. Common conjecture: $\mathbf{P} \neq \mathbf{NP}$.
- Goal: Obtain from **NP** a more realistic class by randomization:

Choose u uniformly at random from $\{0,1\}^{p(|x|)}$.

Randomizing NP

Definition (Accept/Reject certificates and probabilities)

Fix some $L \in \mathbf{NP}$ decided by M using certificates u of length $p(\cdot)$:

$$A_{M,x} := \{u \in \{0,1\}^{p(|x|)} \mid M(x,u) = 1\} \text{ and } R_{M,x} := \{0,1\}^{p(|x|)} \setminus A_{M,x}.$$

Randomizing NP

Definition (Accept/Reject certificates and probabilities)

Fix some $L \in \mathbf{NP}$ decided by M using certificates u of length $p(\cdot)$:

$$A_{M,x} := \{u \in \{0,1\}^{p(|x|)} \mid M(x,u) = 1\} \text{ and } R_{M,x} := \{0,1\}^{p(|x|)} \setminus A_{M,x}.$$

- If we choose $u \in \{0,1\}^{p(|x|)}$ uniformly at random:
  - $A_{M,x}$ is the event that u “says accept x”.
  - $R_{M,x}$ is the event that u “says reject x”.

Randomizing NP

Definition (Accept/Reject certificates and probabilities)

Fix some $L \in \mathbf{NP}$ decided by M using certificates u of length $p(\cdot)$:

$$A_{M,x} := \{u \in \{0,1\}^{p(|x|)} \mid M(x,u) = 1\} \text{ and } R_{M,x} := \{0,1\}^{p(|x|)} \setminus A_{M,x}.$$

- If we choose $u \in \{0,1\}^{p(|x|)}$ uniformly at random:
  - $A_{M,x}$ is the event that u “says accept x”.
  - $R_{M,x}$ is the event that u “says reject x”.

Definition (Accept/Reject certificates and probabilities (cont’d))

$$\Pr[A_{M,x}] := \frac{|A_{M,x}|}{2^{p(|x|)}} \text{ and } \Pr[R_{M,x}] := \frac{|R_{M,x}|}{2^{p(|x|)}} = 1 - \Pr[A_{M,x}].$$

Randomizing NP

Definition (Accept/Reject certificates and probabilities)

Fix some $L \in \mathbf{NP}$ decided by M using certificates u of length $p(\cdot)$:

$$A_{M,x} := \{u \in \{0,1\}^{p(|x|)} \mid M(x,u) = 1\} \text{ and } R_{M,x} := \{0,1\}^{p(|x|)} \setminus A_{M,x}.$$

- If we choose $u \in \{0,1\}^{p(|x|)}$ uniformly at random:
  - $A_{M,x}$ is the event that u "says accept x".
  - $R_{M,x}$ is the event that u "says reject x".

Definition (Accept/Reject certificates and probabilities (cont'd))

$$\Pr[A_{M,x}] := \frac{|A_{M,x}|}{2^{p(|x|)}} \text{ and } \Pr[R_{M,x}] := \frac{|R_{M,x}|}{2^{p(|x|)}} = 1 - \Pr[A_{M,x}].$$

$L \in \mathbf{NP}$ iff $\forall x \in \{0,1\}^*$:

$$x \in L \Rightarrow \Pr[A_{M,x}] \geq 2^{-p(|x|)} \text{ and } x \notin L \Rightarrow \Pr[A_{M,x}] = 0.$$

Randomizing NP: Example SAT

- Input: CNF-formula ϕ with n variables.
- Output: Choose truth assignment $u \in \{0, 1\}^n$ uniformly at random.
  - If u satisfies ϕ, output yes, $\phi \in$ SAT.
  - Else, output probably, $\phi \notin$ SAT.

- If output is yes, $\phi \in$ SAT, then we know $\phi \in$ SAT for sure.
- But what if output is probably, $\phi \notin$ SAT?

Randomizing NP: Example SAT

- Input: CNF-formula ϕ with n variables.
- Output: Choose truth assignment $u \in \{0,1\}^n$ uniformly at random.
  - If u satisfies ϕ, output yes, $\phi \in$ SAT.
  - Else, output probably, $\phi \notin$ SAT.

- If output is yes, $\phi \in$ SAT, then we know $\phi \in$ SAT for sure.
- But what if output is probably, $\phi \notin$ SAT?
- Consider $\phi = x_1 \wedge x_2 \wedge \ldots \wedge x_n \in$ SAT:
  - Probability of probably, $\phi \notin$ SAT: $\Pr[R_{M,x}] = 1 - 2^{-n}$
  - Called false negative.

Randomizing NP: Example SAT

- Input: CNF-formula ϕ with n variables.
- Output: Choose truth assignment $u \in \{0, 1\}^n$ uniformly at random.
  - If u satisfies ϕ, output yes, $\phi \in$ SAT.
  - Else, output probably, $\phi \notin$ SAT.

- If output is yes, $\phi \in$ SAT, then we know $\phi \in$ SAT for sure.
- But what if output is probably, $\phi \notin$ SAT?
- Consider $\phi = x_1 \wedge x_2 \wedge \ldots \wedge x_n \in$ SAT:
  - Probability of probably, $\phi \notin$ SAT: $\Pr[R_{M,x}] = 1 - 2^{-n}$
  - Called false negative.
- If we run this algorithm r-times,
  prob. of false negative decreases to: $(1 - 2^{-n})^r \approx e^{-r/2^n}$.

Randomizing NP: Example SAT

- Input: CNF-formula ϕ with n variables.
- Output: Choose truth assignment $u \in \{0, 1\}^n$ uniformly at random.
  - If u satisfies ϕ, output yes, $\phi \in$ SAT.
  - Else, output probably, $\phi \notin$ SAT.

- If output is yes, $\phi \in$ SAT, then we know $\phi \in$ SAT for sure.
- But what if output is probably, $\phi \notin$ SAT?
- Consider $\phi = x_1 \wedge x_2 \wedge \ldots \wedge x_n \in$ SAT:
  - Probability of probably, $\phi \notin$ SAT: $\Pr[R_{M,x}] = 1 - 2^{-n}$
  - Called false negative.
- If we run this algorithm r-times,
  prob. of false negative decreases to: $(1 - 2^{-n})^r \approx e^{-r/2^n}$.
- Exponential number $r \sim 2^n$ required to reduce this to any tolerable error bound like $1/4$ or $1/10$.
- Not that helpful as SAT $\in$ **EXP** (zero prob. of false negative).

Randomizing NP: Conclusion

- Not enough to only choose certificate u at random,
  we need to require that $\Pr[A_{M,x}]$ is significantly larger than $2^{-p(|x|)}$;
  otherwise we'll stay in **NP**.

Randomizing NP: Conclusion

- Not enough to only choose certificate u at random,
  we need to require that $\Pr[A_{M,x}]$ is significantly larger than $2^{-p(|x|)}$;
  otherwise we'll stay in **NP**.
- Goal:
  Polynomial number $r(|x|)$ of reruns should make prob. of false negatives arbitrary small.

Randomizing NP: Conclusion

- Not enough to only choose certificate u at random,
  we need to require that $\Pr[A_{M,x}]$ is significantly larger than $2^{-p(|x|)}$;
  otherwise we'll stay in **NP**.
- Goal:
  Polynomial number $r(|x|)$ of reruns should make prob. of false negatives arbitrary small.
- This holds if $\Pr[A_{M,x}] \geq n^{-k}$ for some $k > 0$:

$$(1 - \Pr[A_{M,x}])^{c|x|^{k+d}} \geq \left(1 - 1/|x|^k\right)^{c|x|^{k+d}} \approx e^{-c|x|^d}$$

as $\lim_{m\to\infty}(1 - 1/m)^m = e^{-1}$.

Agenda

- Motivation: From **NP** to a more realistic class by randomization ✓
  - Choosing the certificate at random ✓
  - Error reduction by rerunning ✓
- Randomized poly-time with one-sided error: **RP**, **coRP**, **ZPP**
  - Definitions
  - Monte Carlo and Las Vegas algorithms
  - Examples: ZEROP and perfect matchings
- Power of randomization with two-sided error: **PP**, **BPP**

Definition of RP

Definition (Randomized P (RP))

$L \in$ **RP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $M(x, u)$ using certificates u of length $|u| = p(|x|)$ such that for every $x \in \{0, 1\}^*$

$$x \in L \Rightarrow \Pr[A_{M,x}] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr[A_{M,x}] = 0.$$

- **P** $\subseteq$ **RP** $\subseteq$ **NP**
- **coRP** $:= \{\overline{L} \mid L \in$ **RP**$\}$
- **RP** unchanged if we replace $\geq 3/4$ by $\geq n^{-k}$ or $\geq 1 - 2^{-n^k}$ ($k > 0$).

Definition of RP

Definition (Randomized P (RP))

$L \in$ **RP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $M(x, u)$ using certificates u of length $|u| = p(|x|)$ such that for every $x \in \{0, 1\}^*$

$$x \in L \Rightarrow \Pr[A_{M,x}] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr[A_{M,x}] = 0.$$

- **P** $\subseteq$ **RP** $\subseteq$ **NP**
- **coRP** $:= \{\overline{L} \mid L \in \textbf{RP}\}$
- **RP** unchanged if we replace $\geq 3/4$ by $\geq n^{-k}$ or $\geq 1 - 2^{-n^k}$ $(k > 0)$.
- Realistic model of computation? How to obtain random bits?
  - "Slightly random sources": see e.g. Papadimitriou p. 261

Definition of RP

Definition (Randomized P (RP))

$L \in$ **RP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $M(x, u)$ using certificates u of length $|u| = p(|x|)$ such that for every $x \in \{0, 1\}^*$

$$x \in L \Rightarrow \Pr[A_{M,x}] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr[A_{M,x}] = 0.$$

- **P** $\subseteq$ **RP** $\subseteq$ **NP**
- **coRP** $:= \{\overline{L} \mid L \in \textbf{RP}\}$
- **RP** unchanged if we replace $\geq 3/4$ by $\geq n^{-k}$ or $\geq 1 - 2^{-n^k}$ $(k > 0)$.
- Realistic model of computation? How to obtain random bits?
  - “Slightly random sources”: see e.g. Papadimitriou p. 261
- One-sided error probabiliy for **RP**:
  - False negatives: if $x \in L$, then $\Pr[R_{M,x}] \leq 1/4$.
  - If $M(x, u) = 1$, output $x \in L$; else output probably, $x \notin L$
  - Error reduction by rerunning a polynomial number of times.

coRP, ZPP

Lemma (coRP)

*$L \in$ **coRP** if and only if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $M(x, u)$ using certificates u of length $|u| = p(|x|)$ such that for every $x \in \{0, 1\}^*$*

$$x \in L \Rightarrow Pr[A_{M,x}] = 1 \text{ and } x \notin L \Rightarrow Pr[A_{M,x}] \leq 1/4.$$

- One-sided error probability for **coRP**:
  - False positives: if $x \notin L$, then $\Pr[A_{M,x}] \leq 1/4$.
  - If $M(x, u) = 1$, output probably, $x \in L$; else output $x \notin L$

coRP, ZPP

Lemma (coRP)

*$L \in$ **coRP** if and only if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $M(x, u)$ using certificates u of length $|u| = p(|x|)$ such that for every $x \in \{0, 1\}^*$*

$$x \in L \Rightarrow Pr[A_{M,x}] = 1 \text{ and } x \notin L \Rightarrow Pr[A_{M,x}] \leq 1/4.$$

- One-sided error probability for **coRP**:
  - False positives: if $x \notin L$, then $\Pr[A_{M,x}] \leq 1/4$.
  - If $M(x, u) = 1$, output probably, $x \in L$; else output $x \notin L$

Definition ("Zero Probability of Error"-P (ZPP))

$$\textbf{ZPP} := \textbf{RP} \cap \textbf{coRP}$$

- If $L \in$ **ZPP**, then we have both an **RP**- and a **coRP**-TM for L.

Agenda

- Motivation: From **NP** to a more realistic class by randomization ✓
- Randomized poly-time with one-sided error: **RP**, **coRP**, **ZPP**
  - Definitions ✓
  - Monte Carlo and Las Vegas algorithms
  - Examples: ZEROP and perfect matchings
- Power of randomization with two-sided error: **PP**, **BPP**

RP-algorithms

- Assume $L \in \mathbf{RP}$ decided by TM $M(\cdot, \cdot)$.
- Given input x:
  - Choose $u \in \{0, 1\}^{p(|x|)}$ uniformly at random.
  - Run $M(x, u)$.
  - If $M(x, u) = 1$, output: yes, $x \in L$.
  - If $M(x, u) = 0$, output: probably, $x \notin L$.
- Called Monte Carlo algorithm.

RP-algorithms

- Assume $L \in$ **RP** decided by TM $M(\cdot, \cdot)$.
- Given input x:
  - Choose $u \in \{0, 1\}^{p(|x|)}$ uniformly at random.
  - Run $M(x, u)$.
  - If $M(x, u) = 1$, output: yes, $x \in L$.
  - If $M(x, u) = 0$, output: probably, $x \notin L$.
- Called Monte Carlo algorithm.
- If we rerun this algorithm exactly k-times:
  - If $x \in L$, probability that at least once yes, $x \in L$

$$\geq 1 - (1 - 3/4)^k = 1 - 4^{-k}$$

  - but if $x \notin L$, we will never know for sure.

RP-algorithms

- Assume $L \in \mathbf{RP}$ decided by TM $M(\cdot,\cdot)$.
- Given input x:
  - Choose $u \in \{0,1\}^{p(|x|)}$ uniformly at random.
  - Run $M(x,u)$.
  - If $M(x,u) = 1$, output: yes, $x \in L$.
  - If $M(x,u) = 0$, output: probably, $x \notin L$.
- Called Monte Carlo algorithm.
- If we rerun this algorithm exactly k-times:
  - If $x \in L$, probability that at least once yes, $x \in L$

$$\geq 1 - (1 - 3/4)^k = 1 - 4^{-k}$$

  - but if $x \notin L$, we will never know for sure.
- Expected running time if we rerun till output yes, $x \in L$:
  - If $x \in L$:
    - Number of reruns geometrically distributed with success prob. $\geq 3/4$, i.e.,
    - the expected number of reruns is at most $4/3$.
    - Expected running time also polynomial.
  - If $x \notin L$:
    - We run forever.

ZPP-algorithms

- Assume $L \in$ **ZPP**.
- Then we have Monte Carlo algorithms for both $x \in L$ and $x \in \overline{L}$.
- Given x:
  - Run both algorithms once.
  - If both reply probably, then output don't know.
  - Otherwise forward the (unique) yes-reply.
- Called Las Vegas algorithm

ZPP-algorithms

- Assume $L \in$ **ZPP**.
- Then we have Monte Carlo algorithms for both $x \in L$ and $x \in \overline{L}$.
- Given x:
  - Run both algorithms once.
  - If both reply probably, then output don't know.
  - Otherwise forward the (unique) yes-reply.
- Called Las Vegas algorithm
- If we rerun this algorithm exactly k-times:
  - If $x \in L$ ($x \in \overline{L}$), probability that at least once yes, $x \in L$ (yes, $x \in \overline{L}$)

$$\geq 1 - (1 - 3/4)^k = 1 - 4^{-k}$$

ZPP-algorithms

- Assume $L \in$ **ZPP**.
- Then we have Monte Carlo algorithms for both $x \in L$ and $x \in \overline{L}$.
- Given x:
  - Run both algorithms once.
  - If both reply probably, then output don't know.
  - Otherwise forward the (unique) yes-reply.
- Called Las Vegas algorithm
- If we rerun this algorithm exactly k-times:
  - If $x \in L$ ($x \in \overline{L}$), probability that at least once yes, $x \in L$ (yes, $x \in \overline{L}$)

$$\geq 1 - (1 - 3/4)^k = 1 - 4^{-k}$$

- Expected running time if we rerun till output yes:
  - In both cases expected number of reruns at most $4/3$.
  - So, randomized algorithm which decides L in expected polynomial time.

ZPP-algorithms

- Assume $L \in$ **ZPP**.
- Then we have Monte Carlo algorithms for both $x \in L$ and $x \in \overline{L}$.
- Given x:
  - Run both algorithms once.
  - If both reply probably, then output don't know.
  - Otherwise forward the (unique) yes-reply.
- Called Las Vegas algorithm
- If we rerun this algorithm exactly k-times:
  - If $x \in L$ ($x \in \overline{L}$), probability that at least once yes, $x \in L$ (yes, $x \in \overline{L}$)

$$\geq 1 - (1 - 3/4)^k = 1 - 4^{-k}$$

- Expected running time if we rerun till output yes:
  - In both cases expected number of reruns at most $4/3$.
  - So, randomized algorithm which decides L in expected polynomial time.
- More on expected running time vs. exact running time later on.

Agenda

- Motivation: From **NP** to a more realistic class by randomization ✓
- Randomized poly-time with one-sided error: **RP**, **coRP**, **ZPP**
  - Definitions ✓
  - Monte Carlo and Las Vegas algorithms ✓
  - Examples: ZEROP and perfect matchings
- Power of randomization with two-sided error: **PP**, **BPP**

ZEROP

- Given: Multivariate polynomial $p(x_1, \ldots, x_k)$, not necessarily expanded, but evaluable in polynomial time.
- Wanted: Decide if $p(x_1, \ldots, x_k)$ is the zero polynomial.

$$\left|\begin{pmatrix} 0 & y^2 & xy \\ z & 0 & y \\ 0 & yz & xz \end{pmatrix}\right| = -y^2(z \cdot xz - 0) + xy(z \cdot yz - 0) = -xy^2z^2 + xy^2z^2 = 0$$

- ZEROP := “All zero polynomials evaluable in polynomial time”.
- E.g. determinant: substitute values for variables, then use Gauß-elemination.
- Not known to be in **P**.

Lemma (cf. Papadimitriou p. 243)

Let $p(x_1, \dots, x_k)$ be a nonzero polynomial with each variable x_i of degree at most d. Then for $M \in \mathbb{N}$:

$$\left|\{(x_1, \dots, x_k) \in \{0, 1, \dots, M-1\}^k \mid p(x_1, \dots, x_k) = 0\}\right| \leq kdM^{k-1}.$$

Lemma (cf. Papadimitriou p. 243)

Let $p(x_1, \ldots, x_k)$ be a nonzero polynomial with each variable x_i of degree at most d. Then for $M \in \mathbb{N}$:

$$\left|\{(x_1, \ldots, x_k) \in \{0, 1, \ldots, M-1\}^k \mid p(x_1, \ldots, x_k) = 0\}\right| \leq kdM^{k-1}.$$

Let $X_1, \ldots, X_k$ be independent random variables, each uniformly distributed on $\{0, 1, \ldots, M-1\}$. Then for $M = 4kd$:

$$p \notin \text{ZEROP} \Rightarrow \Pr\left[p(X_1, \ldots, X_k) = 0\right] \leq \frac{kdM^{k-1}}{M^k} = \frac{kd}{M} = \frac{1}{4}.$$

Lemma (cf. Papadimitriou p. 243)

Let $p(x_1, \ldots, x_k)$ be a nonzero polynomial with each variable x_i of degree at most d. Then for $M \in \mathbb{N}$:

$$\left|\{(x_1, \ldots, x_k) \in \{0, 1, \ldots, M-1\}^k \mid p(x_1, \ldots, x_k) = 0\}\right| \leq kdM^{k-1}.$$

Let $X_1, \ldots, X_k$ be independent random variables, each uniformly distributed on $\{0, 1, \ldots, M-1\}$. Then for $M = 4kd$:

$$p \notin \text{ZEROP} \Rightarrow \Pr\left[p(X_1, \ldots, X_k) = 0\right] \leq \frac{kdM^{k-1}}{M^k} = \frac{kd}{M} = \frac{1}{4}.$$

- So we can decide $p \in$ ZEROP in **coRP** if
  - we can evaluate $p(\cdot)$ in polynomial time, and
  - d is polynomial in the representation of p.

ZEROP

Lemma (cf. Papadimitriou p. 243)

Let $p(x_1, \ldots, x_k)$ be a nonzero polynomial with each variable x_i of degree at most d. Then for $M \in \mathbb{N}$:

$$\left|\{(x_1, \ldots, x_k) \in \{0, 1, \ldots, M-1\}^k \mid p(x_1, \ldots, x_k) = 0\}\right| \leq kdM^{k-1}.$$

Let $X_1, \ldots, X_k$ be independent random variables, each uniformly distributed on $\{0, 1, \ldots, M-1\}$. Then for $M = 4kd$:

$$p \notin \text{ZEROP} \Rightarrow \Pr\left[p(X_1, \ldots, X_k) = 0\right] \leq \frac{kdM^{k-1}}{M^k} = \frac{kd}{M} = \frac{1}{4}.$$

- So we can decide $p \in$ ZEROP in **coRP** if
  - we can evaluate $p(\cdot)$ in polynomial time, and
  - d is polynomial in the representation of p.
- See Arora p. 130 for work around if d is exponential
  - E.g. $p(x) = (\ldots((x-1)^2)^2 \ldots)^2$.

Perfect Matchings in Bipartite Graphs

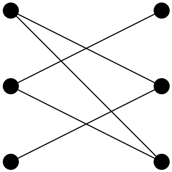
- Given: bipartite graph $G = (U, V, E)$ with

$$|U| = |V| = n \text{ and } E \subseteq U \times V$$

- Wanted: $M \subseteq E$ such that

$$\forall (u, v), (u', v') \in M : u \neq u' \wedge v \neq v' \quad \text{(matching)}$$

$$|M| = n \quad \text{(perfect)}$$



Perfect Matchings in Bipartite Graphs

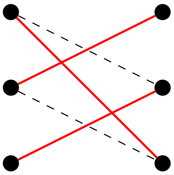
- Given: bipartite graph $G = (U, V, E)$ with

$$|U| = |V| = n \text{ and } E \subseteq U \times V$$

- Wanted: $M \subseteq E$ such that

$$\forall (u, v), (u', v') \in M : u \neq u' \wedge v \neq v' \quad \text{(matching)}$$

$$|M| = n \quad \text{(perfect)}$$



Perfect Matchings in Bipartite Graphs

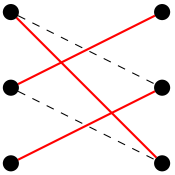
- Given: bipartite graph $G = (U, V, E)$ with

$$|U| = |V| = n \text{ and } E \subseteq U \times V$$

- Wanted: $M \subseteq E$ such that

$$\forall (u, v), (u', v') \in M : u \neq u' \wedge v \neq v' \quad \text{(matching)}$$

$$|M| = n \quad \text{(perfect)}$$



- Problem is known to be solvable in time $\mathcal{O}(n^5)$ (and better).
- So it is in **RP**.

Perfect Matchings in Bipartite Graphs

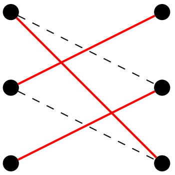
- Given: bipartite graph $G = (U, V, E)$ with

$$|U| = |V| = n \text{ and } E \subseteq U \times V$$

- Wanted: $M \subseteq E$ such that

$$\forall (u, v), (u', v') \in M : u \neq u' \wedge v \neq v' \quad \text{(matching)}$$

$$|M| = n \quad \text{(perfect)}$$



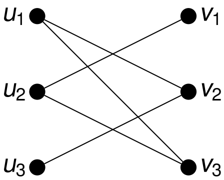
- Problem is known to be solvable in time $\mathcal{O}(n^5)$ (and better).
- So it is in **RP**.
- Still, some "easy" randomized algorithm relying on ZEROP.

Perfect Matchings in Bipartite Graphs

- For bipartite graph $G = (U, V, E)$ define square matrix M:

$$M_{ij} = \begin{cases} x_{ij} & \text{if } (u_i, v_j) \in E \\ 0 & \text{else .} \end{cases}$$

- Output:
  - "has perfect matching" if $\mathtt{det}(M) \notin \text{ZEROP}$
  - "might not have perfect matching" if $\mathtt{det}(M) \in \text{ZEROP}$



$$\left|\begin{pmatrix} 0 & x_{1,2} & x_{1,3} \\ x_{2,1} & 0 & x_{2,3} \\ 0 & x_{3,2} & 0 \end{pmatrix}\right| = x_{1,3}x_{2,1}x_{3,2}$$

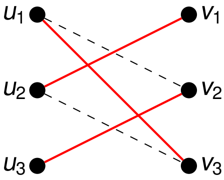
- Relies on Leibniz formula: $\mathtt{det}\, M = \sum_{\sigma \in S_n} \text{sgn}(\sigma) \prod_{i=1}^{n} M_{i,\sigma(i)}$.

Perfect Matchings in Bipartite Graphs

- For bipartite graph $G = (U, V, E)$ define square matrix M:

$$M_{ij} = \begin{cases} x_{ij} & \text{if } (u_i, v_j) \in E \\ 0 & \text{else .} \end{cases}$$

- Output:
  - "has perfect matching" if $\det(M) \notin$ ZEROP
  - "might not have perfect matching" if $\det(M) \in$ ZEROP



$$\left| \begin{pmatrix} 0 & x_{1,2} & x_{1,3} \\ x_{2,1} & 0 & x_{2,3} \\ 0 & x_{3,2} & 0 \end{pmatrix} \right| = x_{1,3} x_{2,1} x_{3,2}$$

- Relies on Leibniz formula: $\det M = \sum_{\sigma \in S_n} \mathrm{sgn}(\sigma) \prod_{i=1}^{n} M_{i,\sigma(i)}$.

Agenda

- Motivation: From **NP** to a more realistic class by randomization ✓
- Randomized poly-time with one-sided error: **RP**, **coRP**, **ZPP** ✓
  - Definitions ✓
  - Monte Carlo and Las Vegas algorithms ✓
  - Examples: ZEROP and perfect matchings ✓
- Power of randomization with two-sided error: **PP**, **BPP**
  - Enlarging **RP** by false negatives and false positives
  - Comparison: **NP**, **RP**, **coRP**, **ZPP**, **BPP**, **PP**
  - Probabilistic Turing machines
  - Expected running time
  - Error reduction for **BPP**
  - Some kind of derandomization for **BPP**
  - **BPP** in the polynomial hierarchy

Probability of error for both $x \in L$ and $x \notin L$

- **RP** obtained from **NP** by
  - choosing certificate u uniformly at random
  - requiring a fixed fraction of accept-certificates if $x \in L$

$$x \in L \Rightarrow \Pr[A_{M,x}] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr[A_{M,x}] = 0.$$

- **RP**-algorithms can only make errors for $x \in L$.

Probability of error for both $x \in L$ and $x \notin L$

- **RP** obtained from **NP** by
  - choosing certificate u uniformly at random
  - requiring a fixed fraction of accept-certificates if $x \in L$

$$x \in L \Rightarrow \Pr\left[A_{M,x}\right] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr\left[A_{M,x}\right] = 0.$$

- **RP**-algorithms can only make errors for $x \in L$.
- By allowing both errors for both cases, can we obtain a class that is
  - larger than **RP**,
  - but still more realistic than **NP**?

Probability of error for both $x \in L$ and $x \notin L$

- **RP** obtained from **NP** by
  - choosing certificate u uniformly at random
  - requiring a fixed fraction of accept-certificates if $x \in L$

$$x \in L \Rightarrow \Pr\left[A_{M,x}\right] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr\left[A_{M,x}\right] = 0.$$

- **RP**-algorithms can only make errors for $x \in L$.
- By allowing both errors for both cases, can we obtain a class that is
  - larger than **RP**,
  - but still more realistic than **NP**?
- Assume we change the definition of **RP** to:

$$x \in L \Leftrightarrow \Pr\left[A_{M,x}\right] \geq 3/4.$$

- Two-sided error probabilities:
  - False negatives: If $x \in L$: $\Pr\left[R_{M,x}\right] \leq 1/4$
  - False positives: If $x \notin L$: $\Pr\left[A_{M,x}\right] < 3/4$
  - Outputs: probably, $x \in L$ and probably, $x \notin L$

Probabilistic Polynomial Time (PP)

Definition (PP)

$L \in$ **PP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $M(x, u)$ using certificates u of length $|u| = p(|x|)$ such that for every $x \in \{0, 1\}^*$

$$x \in L \Leftrightarrow \Pr[A_{M,x}] \geq 3/4.$$

Probabilistic Polynomial Time (PP)

Definition (PP)

$L \in$ **PP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $M(x, u)$ using certificates u of length $|u| = p(|x|)$ such that for every $x \in \{0, 1\}^*$

$$x \in L \Leftrightarrow \Pr[A_{M,x}] \geq 3/4.$$

- **RP** $\subseteq$ **PP** $\subseteq$ **EXP**

Probabilistic Polynomial Time (PP)

Definition (PP)

$L \in$ **PP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $M(x, u)$ using certificates u of length $|u| = p(|x|)$ such that for every $x \in \{0, 1\}^*$

$$x \in L \Leftrightarrow \Pr[A_{M,x}] \geq 3/4.$$

- **RP** $\subseteq$ **PP** $\subseteq$ **EXP**
- One can show:
  - May replace $\geq$ by $>$.
  - May replace $3/4$ by $1/2$.
  - **PP** $=$ **coPP**

Probabilistic Polynomial Time (PP)

Definition (PP)

$L \in$ **PP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $M(x, u)$ using certificates u of length $|u| = p(|x|)$ such that for every $x \in \{0, 1\}^*$

$$x \in L \Leftrightarrow \Pr\left[A_{M,x}\right] \geq 3/4.$$

- **RP** $\subseteq$ **PP** $\subseteq$ **EXP**
- One can show:
  - May replace $\geq$ by $>$.
  - May replace $3/4$ by $1/2$.
  - **PP** $=$ **coPP**
- **PP**: "$x \in L$ iff x is accepted by a majority"
  - If $x \notin L$, then x is not accepted by a majority ($\neq$ a majority rejects x!)

Probabilistic Polynomial Time (PP)

Definition (PP)

$L \in$ **PP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $M(x, u)$ using certificates u of length $|u| = p(|x|)$ such that for every $x \in \{0, 1\}^*$

$$x \in L \Leftrightarrow \Pr[A_{M,x}] \geq 3/4.$$

- **RP** $\subseteq$ **PP** $\subseteq$ **EXP**
- One can show:
  - May replace $\geq$ by $>$.
  - May replace $3/4$ by $1/2$.
  - **PP** $=$ **coPP**
- **PP**: "$x \in L$ iff x is accepted by a majority"
  - If $x \notin L$, then x is not accepted by a majority ($\neq$ a majority rejects x!)
- Next: **PP** is at least as untractable as **NP**.

$\mathbf{NP} \subseteq \mathbf{PP}$

Theorem

$$\mathbf{NP} \subseteq \mathbf{PP}$$

- Assume TM $M(x, u)$ for $L \in \mathbf{NP}$ uses certificates u of length $p(|x|)$.
- Consider TM $N(x, w)$ with $|w| = p(|x|) + 2$:
  - If $w = 00u$, define $N(x, w) := M(x, u)$.
  - Else $N(x, w) = 1$ iff $w \neq 11 \ldots 1$.

NP $\subseteq$ PP

Theorem

$$\mathbf{NP} \subseteq \mathbf{PP}$$

- Assume TM $M(x, u)$ for $L \in \mathbf{NP}$ uses certificates u of length $p(|x|)$.
- Consider TM $N(x, w)$ with $|w| = p(|x|) + 2$:
  - If $w = 00u$, define $N(x, w) := M(x, u)$.
  - Else $N(x, w) = 1$ iff $w \neq 11\ldots1$.
- Choose w uniformly on $\{0, 1\}^{p(|x|)+2}$ at random:
  - If $x \in L$: $\Pr[A_{N,x}] \geq$
  - If $x \notin L$: $\Pr[A_{N,x}] =$

NP $\subseteq$ PP

Theorem

$$\mathbf{NP} \subseteq \mathbf{PP}$$

- Assume TM $M(x, u)$ for $L \in \mathbf{NP}$ uses certificates u of length $p(|x|)$.
- Consider TM $N(x, w)$ with $|w| = p(|x|) + 2$:
  - If $w = 00u$, define $N(x, w) := M(x, u)$.
  - Else $N(x, w) = 1$ iff $w \neq 11\ldots1$.
- Choose w uniformly on $\{0, 1\}^{p(|x|)+2}$ at random:
  - If $x \in L$: $\Pr[A_{N,x}] \geq 3/4 - 2^{-p(|x|)-2} + 2^{-p(|x|)-2} = 3/4$
  - If $x \notin L$: $\Pr[A_{N,x}] =$

NP $\subseteq$ PP

Theorem

$$\mathbf{NP} \subseteq \mathbf{PP}$$

- Assume TM $M(x, u)$ for $L \in \mathbf{NP}$ uses certificates u of length $p(|x|)$.
- Consider TM $N(x, w)$ with $|w| = p(|x|) + 2$:
  - If $w = 00u$, define $N(x, w) := M(x, u)$.
  - Else $N(x, w) = 1$ iff $w \neq 11 \ldots 1$.
- Choose w uniformly on $\{0, 1\}^{p(|x|)+2}$ at random:
  - If $x \in L$: $\Pr[A_{N,x}] \geq 3/4 - 2^{-p(|x|)-2} + 2^{-p(|x|)-2} = 3/4$
  - If $x \notin L$: $\Pr[A_{N,x}] = 3/4 - 2^{-p(|x|)-2} < 3/4$

“Bounded probability of error”-P (BPP)

- By the previous result:
  - **PP** does not seem to capture realistic computation.

“Bounded probability of error”-P (BPP)

- By the previous result:
  - **PP** does not seem to capture realistic computation.
- Proof relied on the dependency of the two error bounds:
  - We traded one-sided error probability

    $$x \in L \Rightarrow \Pr\left[A_{M,x}\right] \geq 2^{-p(|x|)} \text{ and } x \notin L \Rightarrow \Pr\left[A_{M,x}\right] = 0$$

    for two-sided error probability

    $$x \in L \Rightarrow \Pr\left[A_{M,x}\right] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr\left[A_{M,x}\right] < 3/4$$

    by adding enough accept-certificates, i.e.,

"Bounded probability of error"-P (BPP)

- By the previous result:
  - **PP** does not seem to capture realistic computation.
- Proof relied on the dependency of the two error bounds:
  - We traded one-sided error probability

    $$x \in L \Rightarrow \Pr[A_{M,x}] \geq 2^{-p(|x|)} \text{ and } x \notin L \Rightarrow \Pr[A_{M,x}] = 0$$

    for two-sided error probability

    $$x \in L \Rightarrow \Pr[A_{M,x}] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr[A_{M,x}] < 3/4$$

    by adding enough accept-certificates, i.e.,
  - we increased the probability for false positives,
  - while decreasing the probability for false negatives.

"Bounded probability of error"-P (BPP)

- By the previous result:
  - **PP** does not seem to capture realistic computation.
- Proof relied on the dependency of the two error bounds:
  - We traded one-sided error probability

    $$x \in L \Rightarrow \Pr[A_{M,x}] \geq 2^{-p(|x|)} \text{ and } x \notin L \Rightarrow \Pr[A_{M,x}] = 0$$

    for two-sided error probability

    $$x \in L \Rightarrow \Pr[A_{M,x}] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr[A_{M,x}] < 3/4$$

    by adding enough accept-certificates, i.e.,
  - we increased the probability for false positives,
  - while decreasing the probability for false negatives.
- Possible fix:
  - Require bounds on both error probabilities.
  - "Bounded error probability of error"-**P**

BPP

Definition (BPP)

$L \in$ **BPP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $M(x, u)$ using certificates u of length $|u| = p(|x|)$ such that for every $x \in \{0,1\}^*$

$$x \in L \Rightarrow \Pr[A_{M,x}] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr[R_{M,x}] \geq 3/4.$$

- **RP** $\subseteq$ **BPP** $=$ **coBPP** $\subseteq$ **PP**
  - Reminder: if $L \in$ **PP**, then $x \notin L \Rightarrow \Pr[A_{M,x}] < 3/4$.

BPP

Definition (BPP)

$L \in$ **BPP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $M(x, u)$ using certificates u of length $|u| = p(|x|)$ such that for every $x \in \{0, 1\}^*$

$$x \in L \Rightarrow \Pr[A_{M,x}] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr[R_{M,x}] \geq 3/4.$$

- **RP $\subseteq$ BPP $=$ coBPP $\subseteq$ PP**
    - Reminder: if $L \in$ **PP**, then $x \notin L \Rightarrow \Pr[A_{M,x}] < 3/4$.
- Two-sided error probabilities:
    - False negatives: If $x \in L$, then $\Pr[R_{M,x}] \leq 1/4$.
    - False positives: If $x \notin L$, then $\Pr[A_{M,x}] \leq 1/4$.
    - Outputs: probably, $x \in L$ and probably, $x \notin L$.
    - Error reduction to 2^{-n} by rerunning (later).

BPP

Definition (BPP)

$L \in$ **BPP** if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $M(x, u)$ using certificates u of length $|u| = p(|x|)$ such that for every $x \in \{0, 1\}^*$

$$x \in L \Rightarrow \Pr[A_{M,x}] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr[R_{M,x}] \geq 3/4.$$

- **RP** $\subseteq$ **BPP** $=$ **coBPP** $\subseteq$ **PP**
  - Reminder: if $L \in$ **PP**, then $x \notin L \Rightarrow \Pr[A_{M,x}] < 3/4$.
- Two-sided error probabilities:
  - False negatives: If $x \in L$, then $\Pr[R_{M,x}] \leq 1/4$.
  - False positives: If $x \notin L$, then $\Pr[A_{M,x}] \leq 1/4$.
  - Outputs: probably, $x \in L$ and probably, $x \notin L$.
  - Error reduction to 2^{-n} by rerunning (later).
- It is unknown whether **BPP** $=$ **NP** or even **BPP** $=$ **P**!
  - Under some non-trivial but “very reasonable” assumptions: **BPP** $=$ **P**!

BPP

Definition (BPP)

$L \in \mathbf{BPP}$ if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM $M(x, u)$ using certificates u of length $|u| = p(|x|)$ such that for every $x \in \{0, 1\}^*$

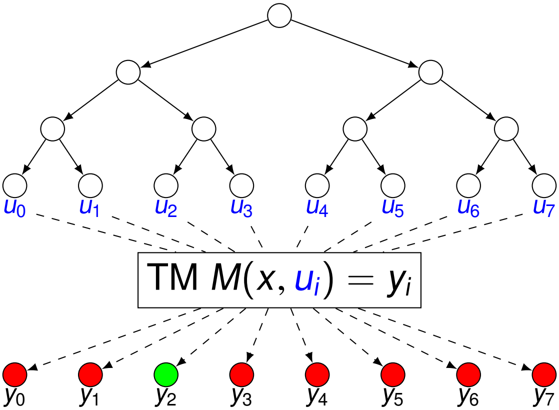
$$x \in L \Rightarrow \Pr[A_{M,x}] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr[R_{M,x}] \geq 3/4.$$

- $\mathbf{RP} \subseteq \mathbf{BPP} = \mathbf{coBPP} \subseteq \mathbf{PP}$
  - Reminder: if $L \in \mathbf{PP}$, then $x \notin L \Rightarrow \Pr[A_{M,x}] < 3/4$.
- Two-sided error probabilities:
  - False negatives: If $x \in L$, then $\Pr[R_{M,x}] \leq 1/4$.
  - False positives: If $x \notin L$, then $\Pr[A_{M,x}] \leq 1/4$.
  - Outputs: probably, $x \in L$ and probably, $x \notin L$.
  - Error reduction to 2^{-n} by rerunning (later).
- It is unknown whether $\mathbf{BPP} = \mathbf{NP}$ or even $\mathbf{BPP} = \mathbf{P}$!
  - Under some non-trivial but “very reasonable” assumptions: $\mathbf{BPP} = \mathbf{P}$!
- **BPP** = “most comprehensive, yet plausible notion of realistic computation” (Papadimitriou p. 259)

Agenda

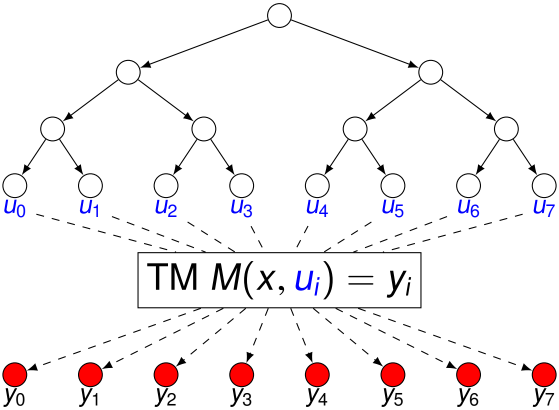
- Motivation: From **NP** to a more realistic class by randomization ✓
- Randomized poly-time with one-sided error: **RP**, **coRP**, **ZPP** ✓
- Power of randomization with two-sided error: **PP**, **BPP**
  - Enlarging **RP** by false negatives and false positives ✓
  - Comparison: **NP**, **RP**, **coRP**, **ZPP**, **BPP**, **PP**
  - Probabilistic Turing machines
  - Expected running time
  - Error reduction for **BPP**
  - Some kind of derandomization for **BPP**
  - **BPP** in the polynomial hierarchy

NP vs. RP vs. coRP vs. ZPP vs. BPP vs. PP



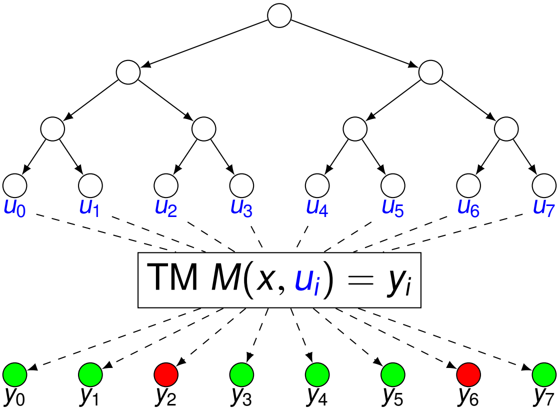
- $L \in$ **NP**:
  - if $x \in L$: at least one ●
  - if $x \notin L$: all ●

NP vs. RP vs. coRP vs. ZPP vs. BPP vs. PP



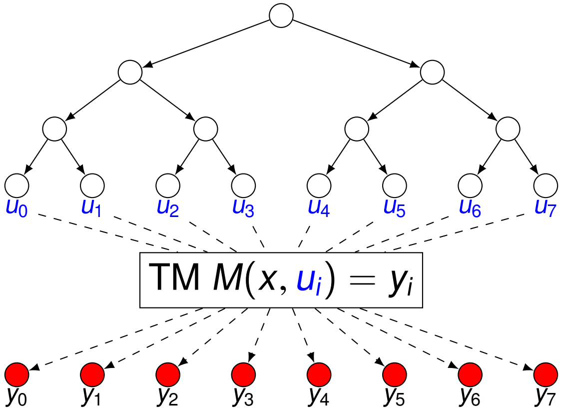
- $L \in$ **NP**:
  - if $x \in L$: at least one ●
  - if $x \notin L$: all ●

NP vs. RP vs. coRP vs. ZPP vs. BPP vs. PP

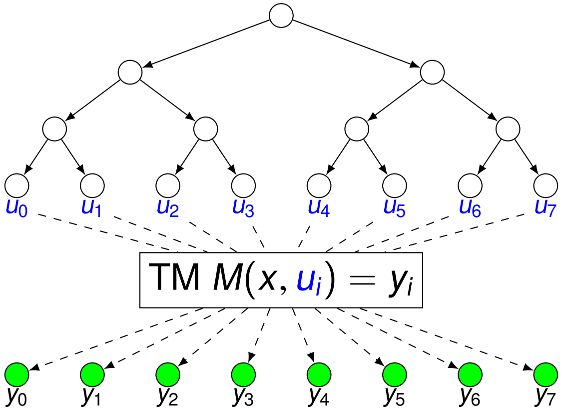


- $L \in$ **RP**:
  - if $x \in L$: at least 75% ●
  - if $x \notin L$: all ●

NP vs. RP vs. coRP vs. ZPP vs. BPP vs. PP

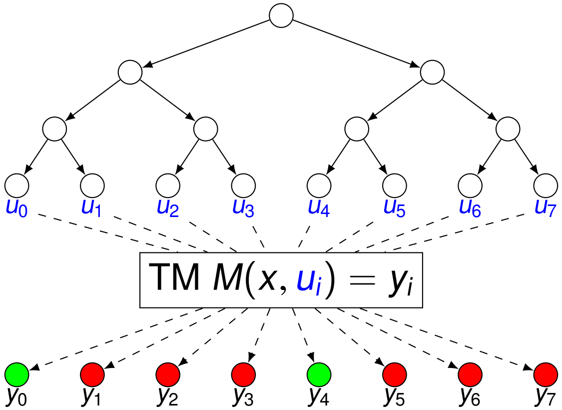


- $L \in$ **RP**:
  - if $x \in L$: at least 75% ●
  - if $x \notin L$: all ●



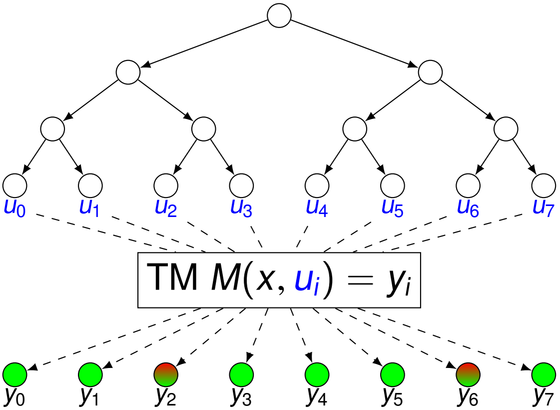
- $L \in$ **coRP**:
  - if $x \in L$: all ●
  - if $x \notin L$: at least 75% ●

NP vs. RP vs. coRP vs. ZPP vs. BPP vs. PP



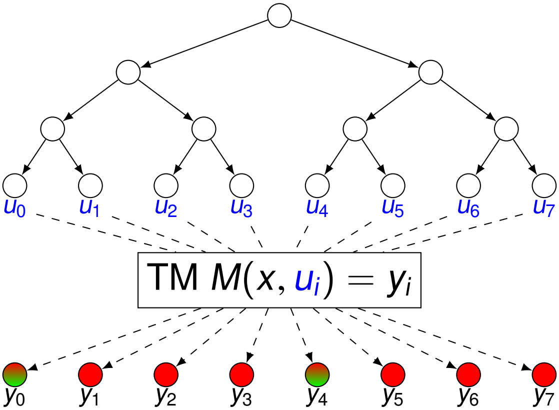
- $L \in$ **coRP**:
  - if $x \in L$: all ●
  - if $x \notin L$: at least 75% ●

NP vs. RP vs. coRP vs. ZPP vs. BPP vs. PP



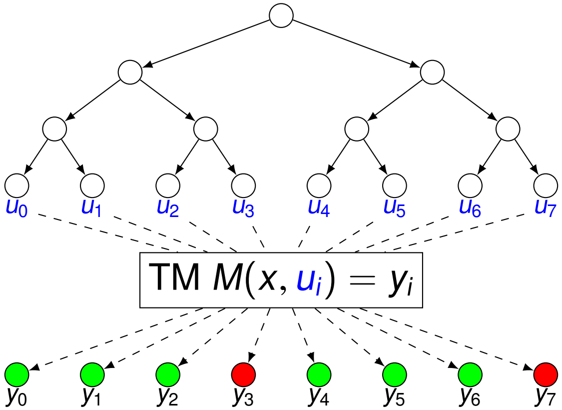
- $L \in$ **ZPP**:
  - if $x \in L$: no 🔴
  - if $x \notin L$: no 🟢

NP vs. RP vs. coRP vs. ZPP vs. BPP vs. PP



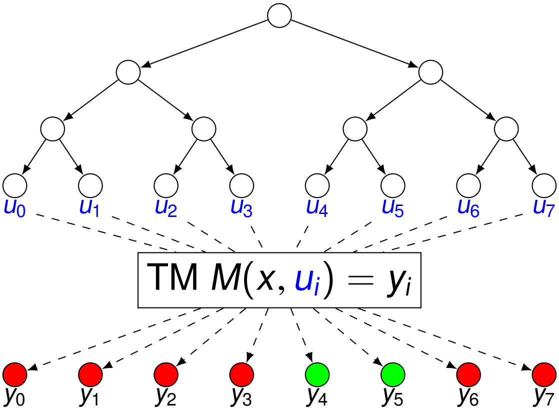
- $L \in$ **ZPP**:
  - if $x \in L$: no ●
  - if $x \notin L$: no ●

NP vs. RP vs. coRP vs. ZPP vs. BPP vs. PP



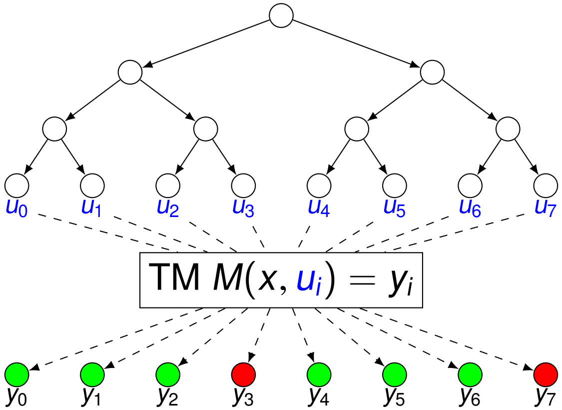
- $L \in$ **BPP**:
  - if $x \in L$: at least 75% ●
  - if $x \notin L$: at least 75% ●

NP vs. RP vs. coRP vs. ZPP vs. BPP vs. PP



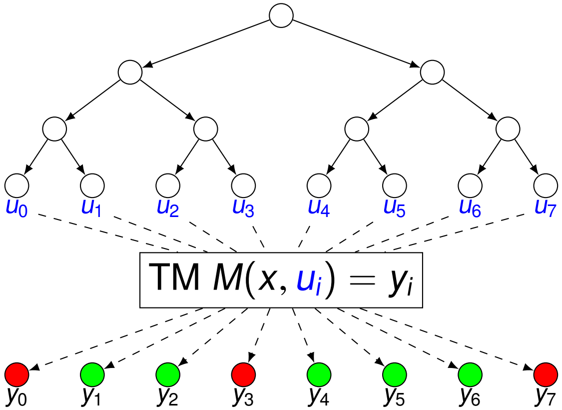
- $L \in$ **BPP**:
  - if $x \in L$: at least 75% 🟢
  - if $x \notin L$: at least 75% 🔴

NP vs. RP vs. coRP vs. ZPP vs. BPP vs. PP



- $L \in$ **PP**:
  - if $x \in L$: at least 75% ●
  - if $x \notin L$: less than 75% ●

NP vs. RP vs. coRP vs. ZPP vs. BPP vs. PP



- $L \in$ **PP**:
  - if $x \in L$: at least 75% ●
  - if $x \notin L$: less than 75% ●

Agenda

- Motivation: From **NP** to a more realistic class by randomization ✓
- Randomized poly-time with one-sided error: **RP**, **coRP**, **ZPP** ✓
- Power of randomization with two-sided error: **PP**, **BPP**
  - Enlarging **RP** by false negatives and false positives ✓
  - Comparison: **NP**, **RP**, **coRP**, **ZPP**, **BPP**, **PP** ✓
  - Probabilistic Turing machines
  - Expected running time
  - Error reduction for **BPP**
  - Some kind of derandomization for **BPP**
  - **BPP** in the polynomial hierarchy

Probabilistic Turing Machines

Definition (PTM)

We obtain from an NDTM $M = (\Gamma, Q, \delta_1, \delta_2)$ a probabilistic TM (PTM) by choosing in every computation step the transition function uniformly at random, i.e., any given run of M on x of length exactly l occurs with probability 2^{-l}.
A PTM runs in time $T(n)$ if the underlying NDTM runs in time $T(n)$, i.e., if M halts on x within at most $T(|x|)$ steps regardless of the random choices it makes.

Probabilistic Turing Machines

Definition (PTM)

We obtain from an NDTM $M = (\Gamma, Q, \delta_1, \delta_2)$ a probabilistic TM (PTM) by choosing in every computation step the transition function uniformly at random, i.e., any given run of M on x of length exactly l occurs with probability 2^{-l}.
A PTM runs in time $T(n)$ if the underlying NDTM runs in time $T(n)$, i.e., if M halts on x within at most $T(|x|)$ steps regardless of the random choices it makes.

Corollary

*$L \in$ **RP** iff there is a poly-time PTM M s.t. for all $x \in \{0,1\}^*$:*

$$x \in L \Rightarrow Pr[M(x) = 1] \geq 3/4 \textit{ and } x \notin L \Rightarrow Pr[M(x) = 1] = 0.$$

Probabilistic Turing Machines

Definition (PTM)

We obtain from an NDTM $M = (\Gamma, Q, \delta_1, \delta_2)$ a probabilistic TM (PTM) by choosing in every computation step the transition function uniformly at random, i.e., any given run of M on x of length exactly l occurs with probability 2^{-l}.
A PTM runs in time $T(n)$ if the underlying NDTM runs in time $T(n)$, i.e., if M halts on x within at most $T(|x|)$ steps regardless of the random choices it makes.

Corollary

*$L \in$ **coRP** iff there is a poly-time PTM M s.t. for all $x \in \{0,1\}^*$:*

$$x \in L \Rightarrow Pr[M(x) = 1] = 1 \text{ and } x \notin L \Rightarrow Pr[M(x) = 1] \leq 1/4.$$

Probabilistic Turing Machines

Definition (PTM)

We obtain from an NDTM $M = (\Gamma, Q, \delta_1, \delta_2)$ a probabilistic TM (PTM) by choosing in every computation step the transition function uniformly at random, i.e., any given run of M on x of length exactly l occurs with probability 2^{-l}.
A PTM runs in time $T(n)$ if the underlying NDTM runs in time $T(n)$, i.e., if M halts on x within at most $T(|x|)$ steps regardless of the random choices it makes.

Corollary

*$L \in$ **BPP** iff there is a poly-time PTM M s.t. for all $x \in \{0,1\}^*$:*

$$x \in L \Rightarrow Pr[M(x) = 1] \geq 3/4 \textit{ and } x \notin L \Rightarrow Pr[M(x) = 1] \leq 1/4.$$

Probabilistic Turing Machines

Definition (PTM)

We obtain from an NDTM $M = (\Gamma, Q, \delta_1, \delta_2)$ a probabilistic TM (PTM) by choosing in every computation step the transition function uniformly at random, i.e., any given run of M on x of length exactly l occurs with probability 2^{-l}.
A PTM runs in time $T(n)$ if the underlying NDTM runs in time $T(n)$, i.e., if M halts on x within at most $T(|x|)$ steps regardless of the random choices it makes.

Corollary

*$L \in$ **PP** iff there is a poly-time PTM M s.t. for all $x \in \{0,1\}^*$:*

$$x \in L \Rightarrow Pr[M(x) = 1] \geq 3/4 \textit{ and } x \notin L \Rightarrow Pr[M(x) = 1] < 3/4.$$

Agenda

- Motivation: From **NP** to a more realistic class by randomization ✓
- Randomized poly-time with one-sided error: **RP**, **coRP**, **ZPP** ✓
- Power of randomization with two-sided error: **PP**, **BPP**
  - Enlarging **RP** by false negatives and false positives ✓
  - Comparison: **NP**, **RP**, **coRP**, **ZPP**, **BPP**, **PP**✓
  - Probabilistic Turing machines✓
  - Expected running time
  - Error reduction for **BPP**
  - Some kind of derandomization for **BPP**
  - **BPP** in the polynomial hierarchy

Expected vs. Exact Running Time

- Recall: if $L \in$ **ZPP**
  - **RP**-algorithms for L and $\overline{L}$.
  - Rerun both algorithms on x until one outputs yes.
  - This decides L in expected polynomial time.
  - But might run infinitely long in the worst case.
- So, is expected time more powerful than exact time?

Expected Running Time

Definition (Expected running time of a PTM)

For a PTM M let $T_{M,x}$ be the random variable that counts the steps of a computation of M on x, i.e., $\Pr[T_{M,x} \leq t]$ is the probability that M halts on x within at most t steps.
We say that M runs in expected time $T(n)$ if $\mathbb{E}[T_{M,x}] \leq T(|x|)$ for every x.

Expected Running Time

Definition (Expected running time of a PTM)

For a PTM M let $T_{M,x}$ be the random variable that counts the steps of a computation of M on x, i.e., $\Pr[T_{M,x} \leq t]$ is the probability that M halts on x within at most t steps.
We say that M runs in expected time $T(n)$ if $\mathbb{E}[T_{M,x}] \leq T(|x|)$ for every x.

- Possibly infinite runs.

Expected Running Time

Definition (Expected running time of a PTM)

For a PTM M let $T_{M,x}$ be the random variable that counts the steps of a computation of M on x, i.e., $\Pr[T_{M,x} \leq t]$ is the probability that M halts on x within at most t steps.
We say that M runs in expected time $T(n)$ if $\mathbb{E}[T_{M,x}] \leq T(|x|)$ for every x.

- Possibly infinite runs.
- So, certificates would need to be unbounded.

Expected Running Time

Definition (Expected running time of a PTM)

For a PTM M let $T_{M,x}$ be the random variable that counts the steps of a computation of M on x, i.e., $\Pr[T_{M,x} \leq t]$ is the probability that M halts on x within at most t steps.
We say that M runs in expected time $T(n)$ if $\mathbb{E}[T_{M,x}] \leq T(|x|)$ for every x.

- Possibly infinite runs.
- So, certificates would need to be unbounded.

Definition (BPeP)

A language L is in **BPeP** if there is a polynomial $T : \mathbb{N} \to \mathbb{N}$ and a PTM M such that for every $x \in \{0,1\}^*$:

$$x \in L \Rightarrow \Pr[M(x) = 1] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr[M(x) = 0] \geq 3/4$$

and $\mathbb{E}[T_{M,x}] \leq T(|x|)$.

Expected Running Time

- Assume $L \in$ **BPeP**.
  - PTM M deciding L within expected running time $T(n)$.

Expected Running Time

- Assume $L \in$ **BPeP**.
  - PTM M deciding L within expected running time $T(n)$.
- Probability that M does more than k steps on input x:

$$\Pr\left[T_{M,x} \geq k\right] \leq \frac{\mathbb{E}\left[T_{M,x}\right]}{k} \leq \frac{T(|x|)}{k}$$

by Markov's inequality.

Expected Running Time

- Assume $L \in$ **BPeP**.
  - PTM M deciding L within expected running time $T(n)$.
- Probability that M does more than k steps on input x:

$$\Pr\left[T_{M,x} \geq k\right] \leq \frac{\mathbb{E}\left[T_{M,x}\right]}{k} \leq \frac{T(|x|)}{k}$$

  by Markov's inequality.
- So, for $k = 10T(|x|)$ (polynomial in $|x|$):

$$\Pr\left[T_{M,x} \geq 10T(|x|)\right] \leq 0.1$$

  for every input x.

Expected Running Time

- New algorithm $\tilde{M}$:
  - Simulate M for at most $10T(|x|)$ steps.
  - If simulation terminates, forward the reply of M.
  - Otherwise, choose reply uniformly at random.

Expected Running Time

- New algorithm $\tilde{M}$:
  - Simulate M for at most $10T(|x|)$ steps.
  - If simulation terminates, forward the reply of M.
  - Otherwise, choose reply uniformly at random.
- Runs in (exact) polynomial time.

Expected Running Time

- New algorithm $\tilde{M}$:
  - Simulate M for at most $10T(|x|)$ steps.
  - If simulation terminates, forward the reply of M.
  - Otherwise, choose reply uniformly at random.
- Runs in (exact) polynomial time.
- Error probabilities:
  - Assume $x \in L$.
  - If simulation halts:
  - Otherwise:

Expected Running Time

- New algorithm $\tilde{M}$:
  - Simulate M for at most $10T(|x|)$ steps.
  - If simulation terminates, forward the reply of M.
  - Otherwise, choose reply uniformly at random.
- Runs in (exact) polynomial time.
- Error probabilities:
  - Assume $x \in L$.
  - If simulation halts: $\leq 1/4$
  - Otherwise:

Expected Running Time

- New algorithm $\tilde{M}$:
  - Simulate M for at most $10T(|x|)$ steps.
  - If simulation terminates, forward the reply of M.
  - Otherwise, choose reply uniformly at random.
- Runs in (exact) polynomial time.
- Error probabilities:
  - Assume $x \in L$.
  - If simulation halts: $\leq 1/4$
  - Otherwise: $= 1/2$

Expected Running Time

- New algorithm $\tilde{M}$:
  - Simulate M for at most $10T(|x|)$ steps.
  - If simulation terminates, forward the reply of M.
  - Otherwise, choose reply uniformly at random.
- Runs in (exact) polynomial time.
- Error probabilities:
  - Assume $x \in L$.
  - If simulation halts: $\leq 1/4$
  - Otherwise: $= 1/2$
  - In total: $1/4 \cdot \underbrace{\Pr[T_{M,x} \leq 10T(|x|)]}_{\leq 1} + 1/2 \cdot \underbrace{(1 - \Pr[T_{M,x} \leq 10T(|x|)])}_{\leq 0.1} \leq 0.3$

Expected Running Time

- New algorithm $\tilde{M}$:
  - Simulate M for at most $10T(|x|)$ steps.
  - If simulation terminates, forward the reply of M.
  - Otherwise, choose reply uniformly at random.
- Runs in (exact) polynomial time.
- Error probabilities:
  - Assume $x \in L$.
  - If simulation halts: $\leq 1/4$
  - Otherwise: $= 1/2$
  - In total: $1/4 \cdot \underbrace{\Pr\left[T_{M,x} \leq 10T(|x|)\right]}_{\leq 1} + 1/2 \cdot \underbrace{\left(1 - \Pr\left[T_{M,x} \leq 10T(|x|)\right]\right)}_{\leq 0.1} \leq 0.3$

Lemma

$$\mathbf{BPP} = \mathbf{BPeP}$$

Lemma

*$L \in$ **ZPP** iff L is decided by some PTM in expected polynomial time.*

Agenda

- Motivation: From **NP** to a more realistic class by randomization ✓
- Randomized poly-time with one-sided error: **RP**, **coRP**, **ZPP** ✓
- Power of randomization with two-sided error: **PP**, **BPP**
  - Enlarging **RP** by false negatives and false positives ✓
  - Comparison: **NP**, **RP**, **coRP**, **ZPP**, **BPP**, **PP**✓
  - Probabilistic Turing machines✓
  - Expected running time✓
  - Error reduction for **BPP**
  - Some kind of derandomization for **BPP**
  - **BPP** in the polynomial hierarchy

Error reduction

- Consider: $L \in$ **RP**:
  - Probability for error after r reruns:
  - if $x \notin L$: $= 0$
  - if $x \in L$: $\leq 4^{-r}$, i.e., r-times probably, $x \notin L$.

Error reduction

- Consider: $L \in$ **RP**:
  - Probability for error after r reruns:
  - if $x \notin L$: $= 0$
  - if $x \in L$: $\leq 4^{-r}$, i.e., r-times probably, $x \notin L$.
- Similarly for $L \in$ **coRP** and $L \in$ **ZPP**.

Error reduction

- Consider: $L \in$ **RP**:
  - Probability for error after r reruns:
  - if $x \notin L$: $= 0$
  - if $x \in L$: $\leq 4^{-r}$, i.e., r-times probably, $x \notin L$.
- Similarly for $L \in$ **coRP** and $L \in$ **ZPP**.
- What if $L \in$ **BPP**?
  - We cannot wait for a yes
  - Instead use the majority.

Error reduction for BPP

Definition (BPP(f))

Let $f : \mathbb{N} \to \mathbb{Q}$ be a function.
$L \in \mathbf{BPP}(f)$ if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial-time TM M such that for every $x \in \{0,1\}^*$

$$x \in L \Rightarrow \Pr[A_{M,x}] \geq f(|x|) \text{ and } x \notin L \Rightarrow \Pr[R_{M,x}] \geq f(|x|).$$

Theorem (Error reduction for BPP)

For any $c > 0$:

$$\mathbf{BPP} = \mathbf{BPP}(1/2 + n^{-c})$$

- The longer the input, the less dominant the "majority" has to be.

Error reduction for BPP (Proof)

- Assume $L \in$ **BPP**, and
- Choose any $c > 0$.

Error reduction for BPP (Proof)

- Assume $L \in \mathbf{BPP}$, and
- Choose any $c > 0$.
- There exists certainly an n_0 s.t. for all $n \geq n_0$:

$$1/2 + n^{-c} \leq 3/4.$$

Error reduction for BPP (Proof)

- Assume $L \in \mathbf{BPP}$, and
- Choose any $c > 0$.
- There exists certainly an n_0 s.t. for all $n \geq n_0$:

$$1/2 + n^{-c} \leq 3/4.$$

- So: $L \cap \{0,1\}^{\geq n_0} \in \mathbf{BPP}(1/2 + n^{-c})$.

Error reduction for BPP (Proof)

- Assume $L \in \mathbf{BPP}$, and
- Choose any $c > 0$.
- There exists certainly an n_0 s.t. for all $n \geq n_0$:

$$1/2 + n^{-c} \leq 3/4.$$

- So: $L \cap \{0,1\}^{\geq n_0} \in \mathbf{BPP}(1/2 + n^{-c})$.
- Thus, $\mathbf{BPP}(1/2 + n^{-c})$-algorithm for L:
  - If $|x| < n_0$, decide $x \in L$ in $\mathbf{P}$ (error prob. $= 0$)
  - Else run **BPP**-algorithm (error prob. $\leq 1/4$)

Error reduction for BPP (Proof)

- Let $L \in \mathbf{BPP}(1/2 + n^{-c})$ for some $c > 0$.

Error reduction for BPP (Proof)

- Let $L \in \mathbf{BPP}(1/2 + n^{-c})$ for some $c > 0$.
- Run $1/2 + n^{-c}$-algorithm r-times on input x:
  - Outputs: $y = y_1 y_2 y_3 \ldots y_r$
  - with $y_i \in \{0, 1\}$ and $y_i = 1$ if output probably, $x \in L$
  - $Y_1 = \sum_{i=1}^{r} y_i$ number of 1s
  - $Y_0 = r - Y_1$ number of 0s

Error reduction for BPP (Proof)

- Let $L \in \mathbf{BPP}(1/2 + n^{-c})$ for some $c > 0$.
- Run $1/2 + n^{-c}$-algorithm r-times on input x:
  - Outputs: $y = y_1 y_2 y_3 \dots y_r$
  - with $y_i \in \{0, 1\}$ and $y_i = 1$ if output probably, $x \in L$
  - $Y_1 = \sum_{i=1}^{r} y_i$ number of 1s
  - $Y_0 = r - Y_1$ number of 0s
- Probability of $y_i = 1$ if $x \in L$, resp. $y_i = 0$ if $x \notin L$:

$$x \in L \;:\; \Pr[y_i = 1] \geq 1/2 + |x|^{-c} \text{ resp. } x \notin L \;:\; \Pr[y_i = 0] \geq 1/2 + |x|^{-c}$$

(y_i indepedent, Bernoulli distributed RVs)

Error reduction for BPP (Proof)

- Let $L \in \mathbf{BPP}(1/2 + n^{-c})$ for some $c > 0$.
- Run $1/2 + n^{-c}$-algorithm r-times on input x:
  - Outputs: $y = y_1 y_2 y_3 \dots y_r$
  - with $y_i \in \{0, 1\}$ and $y_i = 1$ if output probably, $x \in L$
  - $Y_1 = \sum_{i=1}^{r} y_i$ number of 1s
  - $Y_0 = r - Y_1$ number of 0s
- Probability of $y_i = 1$ if $x \in L$, resp. $y_i = 0$ if $x \notin L$:

$$x \in L \ : \ \Pr[y_i = 1] \geq 1/2 + |x|^{-c} \text{ resp. } x \notin L \ : \ \Pr[y_i = 0] \geq 1/2 + |x|^{-c}$$

(y_i indepedent, Bernoulli distributed RVs)

- Expected number of 1s for $x \in L$, resp. of 0s if $x \notin L$:

$$x \in L \ : \ \mathbb{E}[Y_1] \geq r/2 + r|x|^{-c} \text{ resp. } x \notin L \ : \ \mathbb{E}[Y_0] \geq r/2 + r|x|^{-c}$$

Error reduction for BPP (Proof)

- Let $L \in \mathbf{BPP}(1/2 + n^{-c})$ for some $c > 0$.
- Run $1/2 + n^{-c}$-algorithm r-times on input x:
  - Outputs: $y = y_1 y_2 y_3 \dots y_r$
  - with $y_i \in \{0, 1\}$ and $y_i = 1$ if output probably, $x \in L$
  - $Y_1 = \sum_{i=1}^{r} y_i$ number of 1s
  - $Y_0 = r - Y_1$ number of 0s
- Probability of $y_i = 1$ if $x \in L$, resp. $y_i = 0$ if $x \notin L$:

$$x \in L \ : \ \Pr[y_i = 1] \geq 1/2 + |x|^{-c} \text{ resp. } x \notin L \ : \ \Pr[y_i = 0] \geq 1/2 + |x|^{-c}$$

(y_i indepedent, Bernoulli distributed RVs)

- Expected number of 1s for $x \in L$, resp. of 0s if $x \notin L$:

$$x \in L \ : \ \mathbb{E}[Y_1] \geq r/2 + r|x|^{-c} \text{ resp. } x \notin L \ : \ \mathbb{E}[Y_0] \geq r/2 + r|x|^{-c}$$

- Intuitively, for e.g. $r = |x|^{c+d}$ for some $d \in \mathbb{N}$ we get:

$$x \in L \ : \ \mathbb{E}[Y_1 - Y_0] \geq 2|x|^{d} \text{ resp. } x \notin L \ : \ \mathbb{E}[Y_0 - Y_1] \geq 2|x|^{d}$$

i.e., expect significant majority in favor of correct answer.

Error reduction for BPP (Proof)

- Idea: let majority decide, i.e., output $x \in L$ iff $Y_1 > Y_0$.

Error reduction for BPP (Proof)

- Idea: let majority decide, i.e., output $x \in L$ iff $Y_1 > Y_0$.
- Assume $x \in L$ in the following
  - Case $x \notin L$ symmetric:
  - set $z_i := 1 - y_i$ and consider Y_0 instead of Y_1

Error reduction for BPP (Proof)

- Idea: let majority decide, i.e., output $x \in L$ iff $Y_1 > Y_0$.
- Assume $x \in L$ in the following
  - Case $x \notin L$ symmetric:
  - set $z_i := 1 - y_i$ and consider Y_0 instead of Y_1
- Probability that “majority” wrongly says $x \notin L$:

$$\Pr[Y_1 \leq Y_0] = \Pr[Y_1 \leq r/2]$$

Error reduction for BPP (Proof)

- Idea: let majority decide, i.e., output $x \in L$ iff $Y_1 > Y_0$.
- Assume $x \in L$ in the following
  - Case $x \notin L$ symmetric:
  - set $z_i := 1 - y_i$ and consider Y_0 instead of Y_1
- Probability that "majority" wrongly says $x \notin L$:

$$\Pr[Y_1 \leq Y_0] = \Pr[Y_1 \leq r/2]$$

- Chernoff bound: for $X \sim \mathrm{Bin}(n; p)$ with $\mu := \mathbb{E}[X]$ and $\delta \in (0, 1)$

$$\Pr[X \leq (1 - \delta)\mu] \leq e^{-\mu\delta^2/2}$$

Error reduction for BPP (Proof)

- Idea: let majority decide, i.e., output $x \in L$ iff $Y_1 > Y_0$.
- Assume $x \in L$ in the following
  - Case $x \notin L$ symmetric:
  - set $z_i := 1 - y_i$ and consider Y_0 instead of Y_1
- Probability that "majority" wrongly says $x \notin L$:

$$\Pr[Y_1 \le Y_0] = \Pr[Y_1 \le r/2]$$

- Chernoff bound: for $X \sim \mathrm{Bin}(n; p)$ with $\mu := \mathbb{E}[X]$ and $\delta \in (0, 1)$

$$\Pr[X \le (1-\delta)\mu] \le e^{-\mu\delta^2/2}$$

- Thus:

$$\Pr[Y_1 \le r/2] = \Pr[Y_1 \le (1 - (1 - r/(2\mu)))\mu] \le e^{-\mu\delta^2/2}$$

as long as $\delta := 1 - r/(2\mu) \in (0, 1)$.

Error reduction for BPP (Proof)

- Bounds on $\delta = 1 - r/(2\mu)$:

$$0 < \delta < 1 \Leftrightarrow 0 < r/2 < \mu \Leftarrow r/2 + r|x|^{-c} \leq \mu$$

- Thus, choose r s.t.

$$\Pr[Y_1 \leq r/2] \leq e^{-\mu\delta^2/2} \leq 1/4.$$

i.e.,

$$\mu\delta^2 \geq 2\log_e 4.$$

- With

$$\mu \geq r/2 + r|x|^{-c}$$

we obtain:

$$\mu\delta^2 = (\mu - r/2)(1-(r/2)/\mu)^2 \geq r|x|^{-c}\left(1 - \frac{r/2}{r/2 + r|x|^{-c}}\right)^2 = r \cdot \frac{|x|^{-3c}}{(1/2 + |x|^{-c})^2}$$

- So, choose $r \geq (\log_e 4) \cdot (|x|^{3c}/2 + 2|x|^{2c} + 2|x|^{c})$.

Error reduction for BPP (Proof)

- For $x \notin L$ we obtain analogously:

$$\Pr[Y_0 \leq Y_1] \leq 1/4 \text{ if } r \geq (|x|^{3c}/2 + 2|x|^{2c} + 2|x|^{c}).$$

- So, a polynomial number of rounds suffices to reduce error probability to at most $1/4$.
- Proof also yields:

Theorem (Error reduction for BPP)

For any $d > 0$:

$$\mathbf{BPP} = \mathbf{BPP}(1 - 2^{-n^d})$$

Error reduction for BPP (Proof)

- For $x \notin L$ we obtain analogously:

$$\Pr\left[Y_0 \leq Y_1\right] \leq 1/4 \text{ if } r \geq (|x|^{3c}/2 + 2|x|^{2c} + 2|x|^{c}).$$

- So, a polynomial number of rounds suffices to reduce error probability to at most $1/4$.
- Proof also yields:

Theorem (Error reduction for BPP)

For any $d > 0$:

$$\mathbf{BPP} = \mathbf{BPP}(1 - 2^{-n^d})$$

- Ex.: Show the theorem.

Agenda

- Motivation: From **NP** to a more realistic class by randomization ✓
- Randomized poly-time with one-sided error: **RP**, **coRP**, **ZPP** ✓
- Power of randomization with two-sided error: **PP**, **BPP**
  - Enlarging **RP** by false negatives and false positives ✓
  - Comparison: **NP**, **RP**, **coRP**, **ZPP**, **BPP**, **PP**✓
  - Probabilistic Turing machines✓
  - Expected running time✓
  - Error reduction for **BPP**✓
  - Some kind of derandomization for **BPP**
  - **BPP** in the polynomial hierarchy

Some Kind of Derandomization

Theorem

*Let $L \in$ **BPP** be decided by a poly-time TM $M(x, u)$ using certificates of poly-length $p(n)$.*
Then for every $n \in \mathbb{N}$ there exists a certificate u_n s.t. for all x with $|x| = n$:

$$x \in L \text{ iff } M(x, u_n) = 1.$$

Some Kind of Derandomization

Theorem

*Let $L \in$ **BPP** be decided by a poly-time TM $M(x, u)$ using certificates of poly-length $p(n)$.*
Then for every $n \in \mathbb{N}$ there exists a certificate u_n s.t. for all x with $|x| = n$:

$$x \in L \text{ iff } M(x, u_n) = 1.$$

- Error reduction: **BPP** $=$ **BPP**$(1 - 4^{-n})$
- For a given n let choose $u \in \{0, 1\}^{p(n)}$ uniformly at random.

Some Kind of Derandomization

Theorem

Let $L \in \mathbf{BPP}$ be decided by a poly-time TM $M(x, u)$ using certificates of poly-length $p(n)$.
Then for every $n \in \mathbb{N}$ there exists a certificate u_n s.t. for all x with $|x| = n$:

$$x \in L \text{ iff } M(x, u_n) = 1.$$

- Error reduction: $\mathbf{BPP} = \mathbf{BPP}(1 - 4^{-n})$
- For a given n let choose $u \in \{0, 1\}^{p(n)}$ uniformly at random.
- Let B_x be the event of bad certificates for x:

$$B_x := \{u \in \{0, 1\}^{p(|x|)} \mid x \in L \Leftrightarrow M(x, u) = 0\}.$$

Some Kind of Derandomization

Theorem

*Let $L \in$ **BPP** be decided by a poly-time TM $M(x, u)$ using certificates of poly-length $p(n)$.*
Then for every $n \in \mathbb{N}$ there exists a certificate u_n s.t. for all x with $|x| = n$:

$$x \in L \text{ iff } M(x, u_n) = 1.$$

- Error reduction: $\mathbf{BPP} = \mathbf{BPP}(1 - 4^{-n})$
- For a given n let choose $u \in \{0,1\}^{p(n)}$ uniformly at random.
- Let B_x be the event of bad certificates for x:

$$B_x := \{u \in \{0,1\}^{p(|x|)} \mid x \in L \Leftrightarrow M(x, u) = 0\}.$$

- $\Pr[B_x] \leq$

Some Kind of Derandomization

Theorem

Let $L \in \mathbf{BPP}$ be decided by a poly-time TM $M(x, u)$ using certificates of poly-length $p(n)$.
Then for every $n \in \mathbb{N}$ there exists a certificate u_n s.t. for all x with $|x| = n$:

$$x \in L \text{ iff } M(x, u_n) = 1.$$

- Error reduction: $\mathbf{BPP} = \mathbf{BPP}(1 - 4^{-n})$
- For a given n let choose $u \in \{0, 1\}^{p(n)}$ uniformly at random.
- Let B_x be the event of bad certificates for x:

$$B_x := \{u \in \{0, 1\}^{p(|x|)} \mid x \in L \Leftrightarrow M(x, u) = 0\}.$$

- $\Pr[B_x] \leq 4^{-n}$

Some Kind of Derandomization

Theorem

*Let $L \in$ **BPP** be decided by a poly-time TM $M(x, u)$ using certificates of poly-length $p(n)$.*
Then for every $n \in \mathbb{N}$ there exists a certificate u_n s.t. for all x with $|x| = n$:

$$x \in L \text{ iff } M(x, u_n) = 1.$$

- Error reduction: **BPP** $=$ **BPP**$(1 - 4^{-n})$
- For a given n let choose $u \in \{0, 1\}^{p(n)}$ uniformly at random.
- Let B_x be the event of bad certificates for x:

$$B_x := \{u \in \{0, 1\}^{p(|x|)} \mid x \in L \Leftrightarrow M(x, u) = 0\}.$$

- $\Pr[B_x] \leq 4^{-n}$
- $\Pr\left[\bigcup_{|x|=n} B_x\right] \leq$

Some Kind of Derandomization

Theorem

*Let $L \in$ **BPP** be decided by a poly-time TM $M(x, u)$ using certificates of poly-length $p(n)$.*
Then for every $n \in \mathbb{N}$ there exists a certificate u_n s.t. for all x with $|x| = n$:

$$x \in L \text{ iff } M(x, u_n) = 1.$$

- Error reduction: **BPP** $=$ **BPP**$(1 - 4^{-n})$
- For a given n let choose $u \in \{0,1\}^{p(n)}$ uniformly at random.
- Let B_x be the event of bad certificates for x:

$$B_x := \{u \in \{0,1\}^{p(|x|)} \mid x \in L \Leftrightarrow M(x, u) = 0\}.$$

- $\Pr[B_x] \leq 4^{-n}$
- $\Pr\left[\bigcup_{|x|=n} B_x\right] \leq \sum_{|x|=n} \Pr[B_x] \leq 2^n \cdot 4^{-n} = 2^{-n}$

Some Kind of Derandomization

Theorem

*Let $L \in$ **BPP** be decided by a poly-time TM $M(x, u)$ using certificates of poly-length $p(n)$.*
Then for every $n \in \mathbb{N}$ there exists a certificate u_n s.t. for all x with $|x| = n$:

$$x \in L \text{ iff } M(x, u_n) = 1.$$

- Error reduction: $\mathbf{BPP} = \mathbf{BPP}(1 - 4^{-n})$
- For a given n let choose $u \in \{0, 1\}^{p(n)}$ uniformly at random.
- Let B_x be the event of bad certificates for x:

$$B_x := \{u \in \{0, 1\}^{p(|x|)} \mid x \in L \Leftrightarrow M(x, u) = 0\}.$$

- $\Pr[B_x] \leq 4^{-n}$
- $\Pr\left[\bigcup_{|x|=n} B_x\right] \leq \sum_{|x|=n} \Pr[B_x] \leq 2^n \cdot 4^{-n} = 2^{-n}$
- $\Pr\left[\bigcap_{|x|=n} \overline{B}_x\right] \geq$

Some Kind of Derandomization

Theorem

*Let $L \in$ **BPP** be decided by a poly-time TM $M(x, u)$ using certificates of poly-length $p(n)$.*
Then for every $n \in \mathbb{N}$ there exists a certificate u_n s.t. for all x with $|x| = n$:

$$x \in L \text{ iff } M(x, u_n) = 1.$$

- Error reduction: **BPP** $=$ **BPP**$(1 - 4^{-n})$
- For a given n let choose $u \in \{0, 1\}^{p(n)}$ uniformly at random.
- Let B_x be the event of bad certificates for x:

$$B_x := \{u \in \{0, 1\}^{p(|x|)} \mid x \in L \Leftrightarrow M(x, u) = 0\}.$$

- $\Pr[B_x] \le 4^{-n}$
- $\Pr\left[\bigcup_{|x|=n} B_x\right] \le \sum_{|x|=n} \Pr[B_x] \le 2^n \cdot 4^{-n} = 2^{-n}$
- $\Pr\left[\bigcap_{|x|=n} \overline{B}_x\right] \ge 1 - 2^{-n} > 0$

Some Kind of Derandomization

Theorem

*Let $L \in$ **BPP** be decided by a poly-time TM $M(x, u)$ using certificates of poly-length $p(n)$.*
Then for every $n \in \mathbb{N}$ there exists a certificate u_n s.t. for all x with $|x| = n$:

$$x \in L \text{ iff } M(x, u_n) = 1.$$

- Error reduction: **BPP** $=$ **BPP**$(1 - 4^{-n})$
- For a given n let choose $u \in \{0, 1\}^{p(n)}$ uniformly at random.
- Let B_x be the event of bad certificates for x:

$$B_x := \{u \in \{0, 1\}^{p(|x|)} \mid x \in L \Leftrightarrow M(x, u) = 0\}.$$

- $\Pr[B_x] \leq 4^{-n}$
- $\Pr\left[\bigcup_{|x|=n} B_x\right] \leq \sum_{|x|=n} \Pr[B_x] \leq 2^n \cdot 4^{-n} = 2^{-n}$
- $\Pr\left[\bigcap_{|x|=n} \overline{B}_x\right] \geq 1 - 2^{-n} > 0$
- Seems unlikely for **NP**.

Agenda

- Motivation: From **NP** to a more realistic class by randomization ✓
- Randomized poly-time with one-sided error: **RP**, **coRP**, **ZPP** ✓
- Power of randomization with two-sided error: **PP**, **BPP**
  - Enlarging **RP** by false negatives and false positives ✓
  - Comparison: **NP**, **RP**, **coRP**, **ZPP**, **BPP**, **PP**✓
  - Probabilistic Turing machines✓
  - Expected running time✓
  - Error reduction for **BPP**✓
  - Some kind of derandomization for **BPP**✓
  - **BPP** in the polynomial hierarchy

BPP in the Polynomial Hierarchy PH

Theorem

$$\mathbf{BPP} \subseteq \mathbf{\Sigma_2^p} \cap \mathbf{\Pi_2^p}$$

- Reminder:
  - Definition of $L \in \mathbf{\Sigma_2^p}$:

    $$x \in L \text{ iff } \exists u \in \{0,1\}^{p(|x|)} \forall v \in \{0,1\}^{p(|x|)} : M(x,u,v) = 1.$$

  - Definition of $L \in \mathbf{\Pi_2^p}$:

    $$x \in L \text{ iff } \forall u \in \{0,1\}^{p(|x|)} \exists v \in \{0,1\}^{p(|x|)} : M(x,u,v) = 1.$$

BPP in the Polynomial Hierarchy PH

Theorem

$$\mathbf{BPP} \subseteq \mathbf{\Sigma_2^p} \cap \mathbf{\Pi_2^p}$$

- Reminder:
  - Definition of $L \in \mathbf{\Sigma_2^p}$:

$$x \in L \text{ iff } \exists u \in \{0,1\}^{p(|x|)} \forall v \in \{0,1\}^{p(|x|)} : M(x,u,v) = 1.$$

  - Definition of $L \in \mathbf{\Pi_2^p}$:

$$x \in L \text{ iff } \forall u \in \{0,1\}^{p(|x|)} \exists v \in \{0,1\}^{p(|x|)} : M(x,u,v) = 1.$$

- As $\mathbf{BPP} = \mathbf{coBPP}$ it suffices to show $\mathbf{BPP} \subseteq \mathbf{\Sigma_2^p}$:

$$L \in \mathbf{BPP} \Rightarrow \overline{L} \in \mathbf{BPP} \Rightarrow \overline{L} \in \mathbf{\Sigma_2^p} \Rightarrow L \in \mathbf{\Pi_2^p}$$

BPP in the Polynomial Hierarchy PH

- We use again that **BPP** $=$ **BPP**$(1 - 4^{-n})$.
- Let $p(\cdot)$ be the polynomial bounding the certificate length.
- Recall $A_{M,x}$: "accept-certificates"

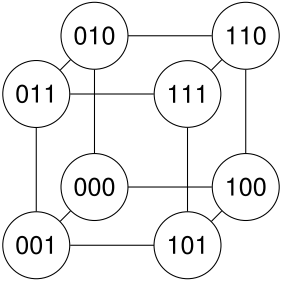
$$A_{M,x} := \{u \in \{0,1\}^{p(|x|)} \mid M(x,u) = 1\}$$

- Then

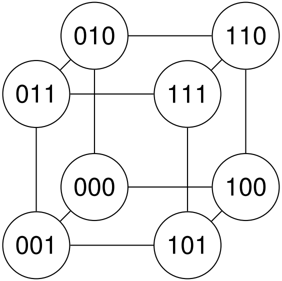
$$x \in L \Rightarrow |A_{M,x}| \geq (1 - 4^{-|x|})2^{p(|x|)} \text{ and } x \notin L \Rightarrow |A_{M,x}| \leq 4^{-n} \cdot 2^{p(|x|)}$$

- Need a formula to distinguish the two cases.

BPP in the Polynomial Hierarchy PH

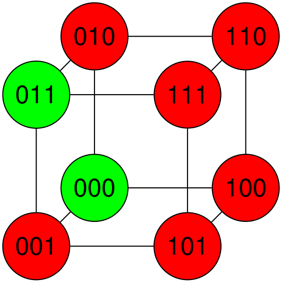


BPP in the Polynomial Hierarchy PH



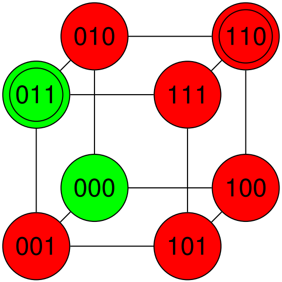
- Assume $|x| = 1$ and $p(|x|) = 3$,
- i.e., possible certificates in $\{0,1\}^3$.
- If $x \in L$, then $|A_{M,x}| \geq 3/4 \cdot 2^3 = 6$.
- If $x \notin L$, then $|A_{M,x}| \leq 1/4 \cdot 2^3 = 2$.

BPP in the Polynomial Hierarchy PH



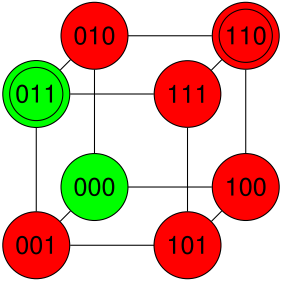
- Assume $x \notin L$, i.e., $|A_{M,x}| \leq 1/4 \cdot 8 = 2$

BPP in the Polynomial Hierarchy PH



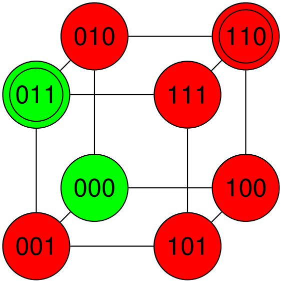
- Assume $x \notin L$, i.e., $|A_{M,x}| \leq 1/4 \cdot 8 = 2$
- Choose any $u_1, u_2 \in \{0, 1\}^3$.

BPP in the Polynomial Hierarchy PH



- Assume $x \notin L$, i.e., $|A_{M,x}| \leq 1/4 \cdot 8 = 2$
- Choose any $u_1, u_2 \in \{0,1\}^3$.
- By chance, we might hit $A_{M,x}$.

BPP in the Polynomial Hierarchy PH

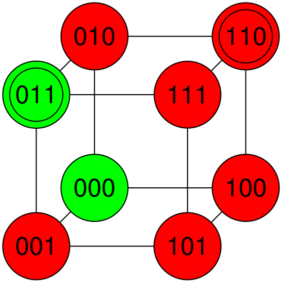


- Assume $x \notin L$, i.e., $|A_{M,x}| \leq 1/4 \cdot 8 = 2$
- Choose any $u_1, u_2 \in \{0,1\}^3$.
- By chance, we might hit $A_{M,x}$.
- Claim: But there is some $r \in \{0,1\}^3$ s.t.

$$\{u_1 \oplus r, u_2 \oplus r\} \cap A_{M,x} = \emptyset.$$

($\oplus$: bitwise xor)

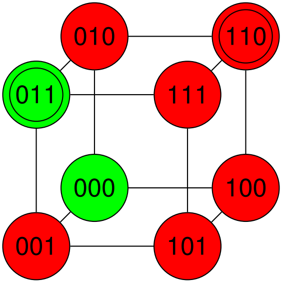
BPP in the Polynomial Hierarchy PH



- Note:

$$u_i \oplus r \in A_{M,x} \text{ iff } r \in A_{M,x} \oplus u_i.$$

BPP in the Polynomial Hierarchy PH



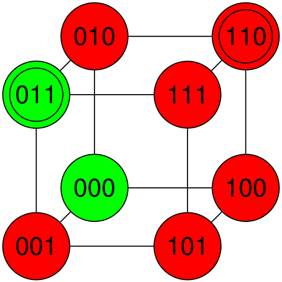
- Note:

$$u_i \oplus r \in A_{M,x} \text{ iff } r \in A_{M,x} \oplus u_i.$$

- So, choose

$$r \in \overline{A_{M,x} \oplus u_1 \cup A_{M,x} \oplus u_2} = \overline{\{000, 011\} \cup \{101, 110\}}.$$

BPP in the Polynomial Hierarchy PH



- Note:

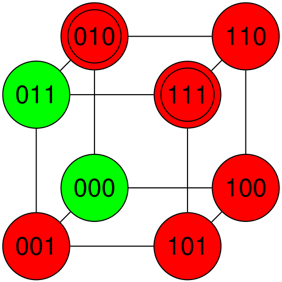
$$u_i \oplus r \in A_{M,x} \text{ iff } r \in A_{M,x} \oplus u_i.$$

- So, choose

$$r \in \overline{A_{M,x} \oplus u_1 \cup A_{M,x} \oplus u_2} = \overline{\{000, 011\} \cup \{101, 110\}}.$$

- E.g. $r = 001$.

BPP in the Polynomial Hierarchy PH



- Note:

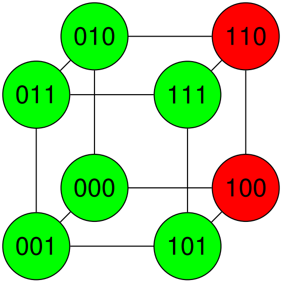
$$u_i \oplus r \in A_{M,x} \text{ iff } r \in A_{M,x} \oplus u_i.$$

- So, choose

$$r \in \overline{A_{M,x} \oplus u_1 \cup A_{M,x} \oplus u_2} = \overline{\{000, 011\} \cup \{101, 110\}}.$$

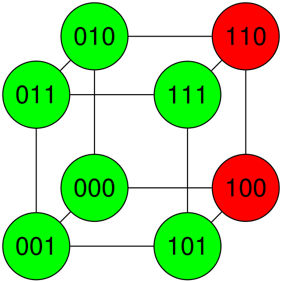
- E.g. $r = 001$.

BPP in the Polynomial Hierarchy PH



- Assume $x \in L$, i.e., $|A_{M,x}| \geq 6$.

BPP in the Polynomial Hierarchy PH

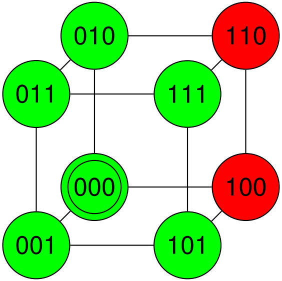


- Assume $x \in L$, i.e., $|A_{M,x}| \geq 6$.
- Claim: We can choose u_1, u_2 s.t. for any $r \in \{0,1\}^3$

$$\{u_1 \oplus r, u_2 \oplus r\} \cap A_{M,x} \neq \emptyset.$$

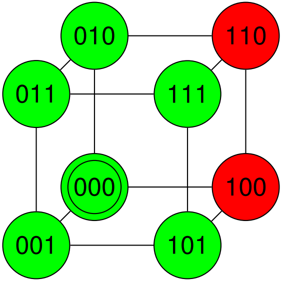
- Note: this is exactly the negation of the previous claim.

BPP in the Polynomial Hierarchy PH



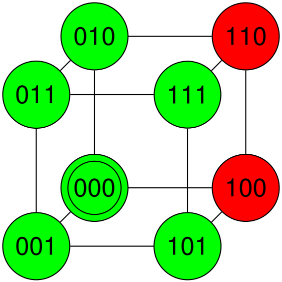
- E.g., take $u_1 = 000$.

BPP in the Polynomial Hierarchy PH



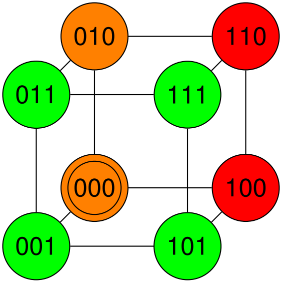
- E.g., take $u_1 = 000$.
- Then $u_1 \oplus r \in R_{M,x}$ iff $r \in u_1 \oplus R_{M,x} = \{100, 110\}$.

BPP in the Polynomial Hierarchy PH



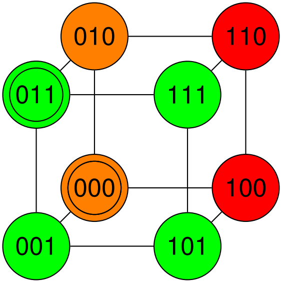
- E.g., take $u_1 = 000$.
- Then $u_1 \oplus r \in R_{M,x}$ iff $r \in u_1 \oplus R_{M,x} = \{100, 110\}$.
- So, take $u_2 \notin 100 \oplus R_{M,x} \cup 110 \oplus R_{M,x}$.

BPP in the Polynomial Hierarchy PH



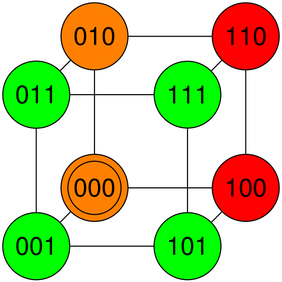
- E.g., take $u_1 = 000$.
- Then $u_1 \oplus r \in R_{M,x}$ iff $r \in u_1 \oplus R_{M,x} = \{100, 110\}$.
- So, take $u_2 \notin 100 \oplus R_{M,x} \cup 110 \oplus R_{M,x}$.

BPP in the Polynomial Hierarchy PH



- E.g., take $u_1 = 000$.
- Then $u_1 \oplus r \in R_{M,x}$ iff $r \in u_1 \oplus R_{M,x} = \{100, 110\}$.
- So, take $u_2 \notin 100 \oplus R_{M,x} \cup 110 \oplus R_{M,x}$.
- E.g., $u_2 = 011$.

BPP in the Polynomial Hierarchy PH

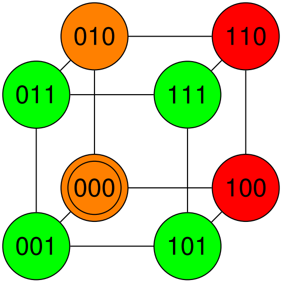


- Summary:

$$x \in L \cap \{0,1\}^1 \text{ iff } \exists u_1, u_2 \in \{0,1\}^3 \forall r \in \{0,1\}^3 : \bigvee_{i=1,2} u_i \oplus r \in A_{M,x}.$$

Reminder: $u_i \oplus r \in A_{M,x}$ iff $M(x, u_i \oplus r) = 1$.

BPP in the Polynomial Hierarchy PH



- Summary:

$$x \in L \cap \{0,1\}^1 \text{ iff } \exists u_1, u_2 \in \{0,1\}^3 \forall r \in \{0,1\}^3 \ : \bigvee_{i=1,2} u_i \oplus r \in A_{M,x}.$$

Reminder: $u_i \oplus r \in A_{M,x}$ iff $M(x, u_i \oplus r) = 1$.

- So, this is in $\mathbf{\Sigma_2^p}$.
- And works also for $|x| > 1$ and arbitrary $p(|x|)$.

BPP in the Polynomial Hierarchy PH

Claim:

Given x set $k := \lceil p(|x|)/|x| \rceil + 1$. Then:

$$x \in L \text{ iff } \exists u_1, \ldots, u_k \in \{0,1\}^{p(|x|)} \forall r \in \{0,1\}^{p(|x|)} : \bigvee_{i=1}^{k} M(x, u_i \oplus r) = 1.$$

- Note, the certificate $u_1 u_2 \ldots u_k$ has length polynomial in $|x|$.
- So, this formula represents a computation in $\mathbf{\Sigma_2^p}$.

BPP in the Polynomial Hierarchy PH

Claim:

Given x set $k := \lceil p(|x|)/|x| \rceil + 1$. Then:

$$x \in L \text{ iff } \exists u_1, \ldots, u_k \in \{0,1\}^{p(|x|)} \forall r \in \{0,1\}^{p(|x|)} : \bigvee_{i=1}^{k} M(x, u_i \oplus r) = 1.$$

- Assume $x \notin L$: To show there is always an r s.t.

$$\bigwedge_{i=1}^{k} r \oplus u_i \notin A_{M,x} \equiv r \notin \bigcup_{i=1}^{k} u_i \oplus A_{M,x}.$$

BPP in the Polynomial Hierarchy PH

Claim:

Given x set $k := \lceil p(|x|)/|x| \rceil + 1$. Then:

$$x \in L \text{ iff } \exists u_1, \ldots, u_k \in \{0,1\}^{p(|x|)} \forall r \in \{0,1\}^{p(|x|)} : \bigvee_{i=1}^{k} M(x, u_i \oplus r) = 1.$$

- Assume $x \notin L$: To show there is always an r s.t.

$$\bigwedge_{i=1}^{k} r \oplus u_i \notin A_{M,x} \equiv r \notin \bigcup_{i=1}^{k} u_i \oplus A_{M,x}.$$

- Size of the complement of this set:

$$\left| \bigcup_{i=1}^{k} u_i \oplus A_{M,x} \right| \leq \sum_{i=1}^{k} \left| u_i \oplus A_{M,x} \right| = k \left| A_{M,x} \right| \leq k 4^{-|x|} 2^{p(|x|)} < 2^{p(|x|)}.$$

BPP in the Polynomial Hierarchy PH

Claim:

Given x set $k := \lceil p(|x|)/|x| \rceil + 1$. Then:

$$x \in L \text{ iff } \exists u_1, \ldots, u_k \in \{0,1\}^{p(|x|)} \forall r \in \{0,1\}^{p(|x|)} : \bigvee_{i=1}^{k} M(x, u_i \oplus r) = 1.$$

- Assume $x \notin L$: To show there is always an r s.t.

$$\bigwedge_{i=1}^{k} r \oplus u_i \notin A_{M,x} \equiv r \notin \bigcup_{i=1}^{k} u_i \oplus A_{M,x}.$$

- Size of the complement of this set:

$$\left| \bigcup_{i=1}^{k} u_i \oplus A_{M,x} \right| \leq \sum_{i=1}^{k} \left| u_i \oplus A_{M,x} \right| = k \left| A_{M,x} \right| \leq k 4^{-|x|} 2^{p(|x|)} < 2^{p(|x|)}.$$

- So, this set cannot be empty no matter how we choose $u_1, \ldots, u_k$.

BPP in the Polynomial Hierarchy PH

Claim:

Given x set $k := \lceil p(|x|)/|x| \rceil + 1$. Then:

$$x \in L \text{ iff } \exists u_1, \ldots, u_k \in \{0,1\}^{p(|x|)} \forall r \in \{0,1\}^{p(|x|)} : \bigvee_{i=1}^{k} M(x, u_i \oplus r) = 1.$$

- Assume $x \in L$: To show there are $u_1, \ldots, u_k$ s.t.

$$\forall r : \bigvee_{i=1}^{k} u_i \oplus r \in A_{M,x} \equiv \neg \exists r : \bigwedge_{i=1}^{k} u_i \in r \oplus R_{M,x}.$$

BPP in the Polynomial Hierarchy PH

Claim:

Given x set $k := \lceil p(|x|)/|x| \rceil + 1$. Then:

$$x \in L \text{ iff } \exists u_1, \dots, u_k \in \{0,1\}^{p(|x|)} \forall r \in \{0,1\}^{p(|x|)} : \bigvee_{i=1}^{k} M(x, u_i \oplus r) = 1.$$

- Assume $x \in L$: To show there are $u_1, \dots, u_k$ s.t.

$$\forall r : \bigvee_{i=1}^{k} u_i \oplus r \in A_{M,x} \equiv \neg \exists r : \bigwedge_{i=1}^{k} u_i \in r \oplus R_{M,x}.$$

- Let $U_1, \dots, U_k$ be independent random variables, uniformly distributed on $\{0,1\}^{p(|x|)}$. (Doesn't matter if some coincide!)

BPP in the Polynomial Hierarchy PH

Claim:

Given x set $k := \lceil p(|x|)/|x| \rceil + 1$. Then:

$$x \in L \text{ iff } \exists u_1, \ldots, u_k \in \{0,1\}^{p(|x|)} \forall r \in \{0,1\}^{p(|x|)} : \bigvee_{i=1}^{k} M(x, u_i \oplus r) = 1.$$

- Assume $x \in L$: To show there are $u_1, \ldots, u_k$ s.t.

$$\forall r : \bigvee_{i=1}^{k} u_i \oplus r \in A_{M,x} \equiv \neg \exists r : \bigwedge_{i=1}^{k} u_i \in r \oplus R_{M,x}.$$

- Let $U_1, \ldots, U_k$ be independent random variables, uniformly distributed on $\{0,1\}^{p(|x|)}$. (Doesn't matter if some coincide!)
- For any given r: $\Pr\left[U_i \in r \oplus R_{M,x}\right] = \frac{|r \oplus R_{M,x}|}{2^{p(|x|)}} = \frac{|R_{M,x}|}{2^{p(|x|)}} \leq 4^{-n}$.

BPP in the Polynomial Hierarchy PH

Claim:

Given x set $k := \lceil p(|x|)/|x| \rceil + 1$. Then:

$$x \in L \text{ iff } \exists u_1, \ldots, u_k \in \{0,1\}^{p(|x|)} \forall r \in \{0,1\}^{p(|x|)} : \bigvee_{i=1}^{k} M(x, u_i \oplus r) = 1.$$

- Assume $x \in L$: To show there are $u_1, \ldots, u_k$ s.t.

$$\forall r : \bigvee_{i=1}^{k} u_i \oplus r \in A_{M,x} \equiv \neg \exists r : \bigwedge_{i=1}^{k} u_i \in r \oplus R_{M,x}.$$

- Let $U_1, \ldots, U_k$ be independent random variables, uniformly distributed on $\{0,1\}^{p(|x|)}$. (Doesn't matter if some coincide!)
- For any given r: $\Pr\left[U_i \in r \oplus R_{M,x}\right] = \frac{|r \oplus R_{M,x}|}{2^{p(|x|)}} = \frac{|R_{M,x}|}{2^{p(|x|)}} \leq 4^{-n}$.
- $\Pr\left[\bigwedge_{i=1}^{k} U_i \in r \oplus R_{M,x}\right] = \Pr\left[U_1 \in r \oplus R_{M,x}\right]^k \leq 4^{-kn}$.

BPP in the Polynomial Hierarchy PH

Claim:

Given x set $k := \lceil p(|x|)/|x| \rceil + 1$. Then:

$$x \in L \text{ iff } \exists u_1, \ldots, u_k \in \{0,1\}^{p(|x|)} \forall r \in \{0,1\}^{p(|x|)} : \bigvee_{i=1}^{k} M(x, u_i \oplus r) = 1.$$

- Assume $x \in L$: To show there are $u_1, \ldots, u_k$ s.t.

$$\forall r : \bigvee_{i=1}^{k} u_i \oplus r \in A_{M,x} \equiv \neg \exists r : \bigwedge_{i=1}^{k} u_i \in r \oplus R_{M,x}.$$

- Let $U_1, \ldots, U_k$ be independent random variables, uniformly distributed on $\{0,1\}^{p(|x|)}$. (Doesn't matter if some coincide!)
- For any given r: $\Pr\left[U_i \in r \oplus R_{M,x}\right] = \frac{|r \oplus R_{M,x}|}{2^{p(|x|)}} = \frac{|R_{M,x}|}{2^{p(|x|)}} \leq 4^{-n}$.
- $\Pr\left[\bigwedge_{i=1}^{k} U_i \in r \oplus R_{M,x}\right] = \Pr\left[U_1 \in r \oplus R_{M,x}\right]^k \leq 4^{-kn}$.
- $\Pr\left[\exists r : \bigwedge_{i=1}^{k} U_i \in r \oplus R_{M,x}\right] \leq \sum_{r \in \{0,1\}^*} 4^{-kn} = 2^{p(|x|)-2kn} < 1$.

BPP in the Polynomial Hierarchy PH

Claim:

Given x set $k := \lceil p(|x|)/|x| \rceil + 1$. Then:

$$x \in L \text{ iff } \exists u_1, \ldots, u_k \in \{0,1\}^{p(|x|)} \forall r \in \{0,1\}^{p(|x|)} \; : \; \bigvee_{i=1}^{k} M(x, u_i \oplus r) = 1.$$

- For both cases there is an n_0 s.t. the bounds hold for all x with $|x| > n_0$.
- $L \cap \{0,1\}^{\leq n_0}$ can be decided trivially in **P**.

Summary

- Obtain **RP** from **NP** by
  - choosing the certificate (transition function) uniformaly at random
  - requiring a bound on $\Pr[A_{M,x}]$ if $x \in L$ s.t.
  - error prob. can be reduced within a polynomial number of reruns.

Summary

- Obtain **RP** from **NP** by
  - choosing the certificate (transition function) uniformaly at random
  - requiring a bound on $\Pr[A_{M,x}]$ if $x \in L$ s.t.
  - error prob. can be reduced within a polynomial number of reruns.
- One-sided probability of error:
  - **RP**: false negatives
  - **coRP**: false postives
  - Monte Carlo algorithms: ZEROP $\in$ **coRP**, perfect matchings $\in$ **RP**

Summary

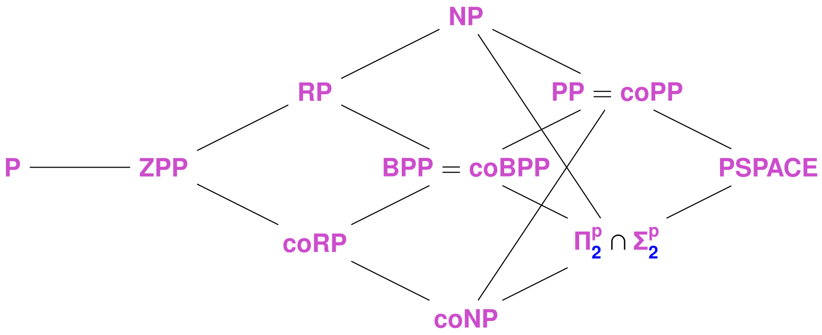
- Obtain **RP** from **NP** by
  - choosing the certificate (transition function) uniformaly at random
  - requiring a bound on $\Pr[A_{M,x}]$ if $x \in L$ s.t.
  - error prob. can be reduced within a polynomial number of reruns.
- One-sided probability of error:
  - **RP**: false negatives
  - **coRP**: false postives
  - Monte Carlo algorithms: ZEROP $\in$ **coRP**, perfect matchings $\in$ **RP**
- **ZPP** $:=$ **RP** $\cap$ **coRP** can be decided in expected polynomial time
  - Zero probability of error (if we wait for the definitive answer)
  - Las Vegas algorithms

Summary

- Obtained **PP** from **RP** by
  - allowing also for false positives
  - Error probabilities depend on each other: $\leq 1/4$ and $< 1 - 1/4$
  - **NP** $\subseteq$ **PP**: “**PP** allows for trading one error prob. for the other”

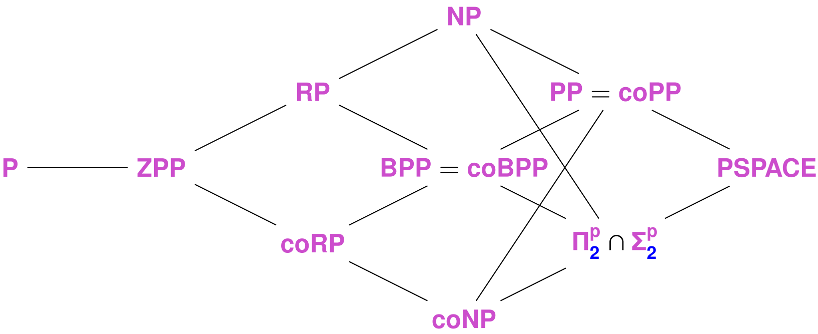
Summary

- Obtained **PP** from **RP** by
  - allowing also for false positives
  - Error probabilities depend on each other: $\leq 1/4$ and $< 1 - 1/4$
  - **NP** $\subseteq$ **PP**: "**PP** allows for trading one error prob. for the other"
- Obtained **BPP** from **PP** by
  - bounding both error prob. independently of each other.
  - Papadimitriou: "most comprehensive, yet plausible notion of realistic computation"
  - Conjecture: **BPP** $=$ **P**
  - Expected running time as powerful as exact running time.
  - One certificate u_n for all x with $|x| = n$.
  - Error reduction to 2^{-n^k} within a polynomial number of reruns.



- $\Pi_2^p \cap \Sigma_2^p \subseteq$ PP unknown.
- NP $\cup$ coNP $\subseteq$ PP known.

Summary



- Gödel Prize (1998) for Toda's theorem (1989): $\mathbf{PH} \subseteq \mathbf{P}^{\mathbf{PP}}$
  - $\mathbf{P}^{\mathbf{PP}}$: poly-time TMs having access to a **PP**-oracle.
  - If $\mathbf{PP} \subseteq \Sigma_k^p$ for some k, then $\mathbf{PH} = \Sigma_k^p$.
  - If $\mathbf{PP} \subseteq \mathbf{PH}$, then **PH** collapses at some finite level as **PP** has complete problems (see exercises).

Syntactic and Semantic Complexity Classes

- Just mentioned: **PP** has complete probems
  - $\phi \in$ MAJSAT iff at least $2^{n-1} + 1$ satisfying assignments of 2^n possible (see exercises).
- Unknown if there are complete problems for **ZPP**, **RP**, **BPP**.

Syntactic and Semantic Complexity Classes

- Just mentioned: **PP** has complete probems
  - $\phi \in$ MAJSAT iff at least $2^{n-1} + 1$ satisfying assignments of 2^n possible (see exercises).
- Unknown if there are complete problems for **ZPP**, **RP**, **BPP**.
- Reason to believe that there are none:
  - **P**, **NP**, **coNP** are syntatic complexity classes (complete problems).
  - **ZPP**, **RP**, **coRP**, **BPP** are semantic complexity classes.
- Example:
  - **NP**:

    $$x \in L \Leftrightarrow \Pr[A_{M,x}] > 0.$$

    Every poly-time TM $M(x, u)$ defines a language in **NP**.
  - **BPP**:

    $$x \in L \Rightarrow \Pr[A_{M,x}] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr[R_{M,x}] \geq 3/4.$$

    Not every poly-time TM $M(x, u)$ defines a language in **BPP**.

Syntactic and Semantic Complexity Classes

- Just mentioned: **PP** has complete probems
  - $\phi \in$ MAJSAT iff at least $2^{n-1} + 1$ satisfying assignments of 2^n possible (see exercises).
- Unknown if there are complete problems for **ZPP**, **RP**, **BPP**.
- Reason to believe that there are none:
  - **P**, **NP**, **coNP** are syntatic complexity classes (complete problems).
  - **ZPP**, **RP**, **coRP**, **BPP** are semantic complexity classes.
- Example:
  - **NP**:

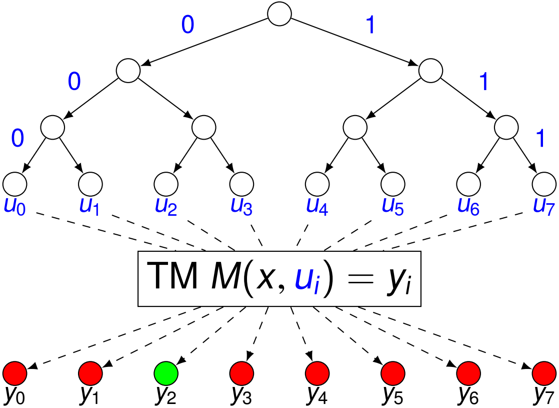
$$x \in L \Leftrightarrow \Pr[A_{M,x}] > 0.$$

    Every poly-time TM $M(x, u)$ defines a language in **NP**.
  - **BPP**:

$$x \in L \Rightarrow \Pr[A_{M,x}] \geq 3/4 \text{ and } x \notin L \Rightarrow \Pr[R_{M,x}] \geq 3/4.$$

    Not every poly-time TM $M(x, u)$ defines a language in **BPP**.
- Ex.: What about **PP**?

Leaf languages



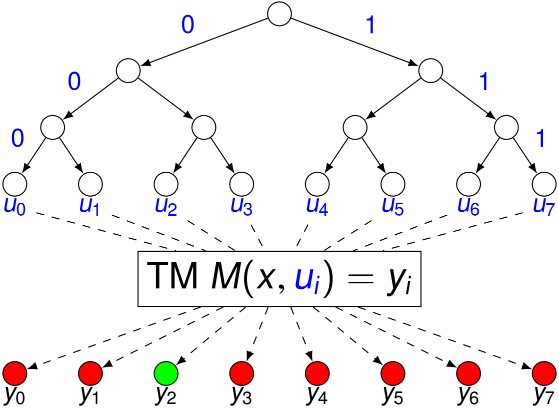
Definition

For a poly-time $M(x, u)$ using certificates $u \in \{0, 1\}^{p(|x|)}$ set

$$L_M(x) := y_0 y_1 \ldots y_{2^{p(|x|)}-1} \text{ with } y_i = M(x, u_i) \text{ and } (u_i)_2 = i$$

The leaf-language of M is then $L_M := \{L_M(x) \mid x \in \{0, 1\}^*\}$.

Leaf languages

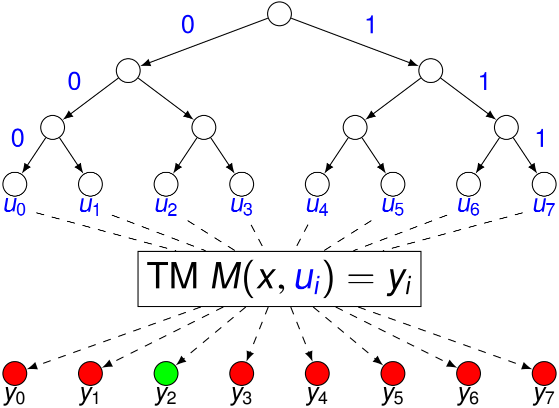


Definition (cont'd)

For $A, R \subseteq \{0,1\}^*$ with $A \cap R = \emptyset$ the class $\mathbf{C}[A, R]$ consists of all language L for which there is a TM $M(x, u)$ s.t. $\forall x \in \{0,1\}^*$:

$$x \in L \Rightarrow L_M(x) \in A \text{ and } x \notin L \Rightarrow L_M(x) \in R.$$

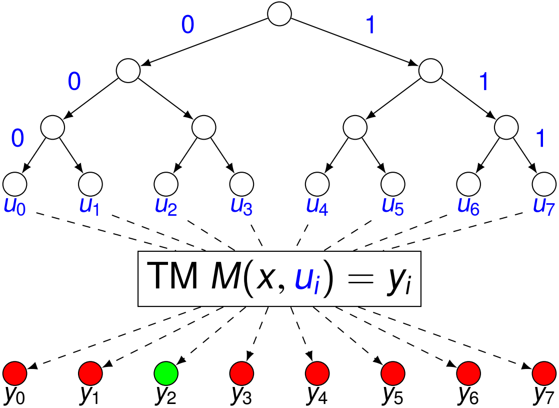
Leaf languages



Definition (cont'd)

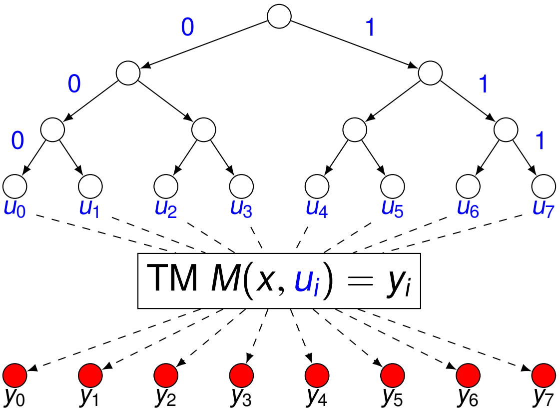
C$[A, R]$ is called syntactic if $A \cup R = \{0, 1\}^*$, otherwise it is called semantic.

Leaf languages



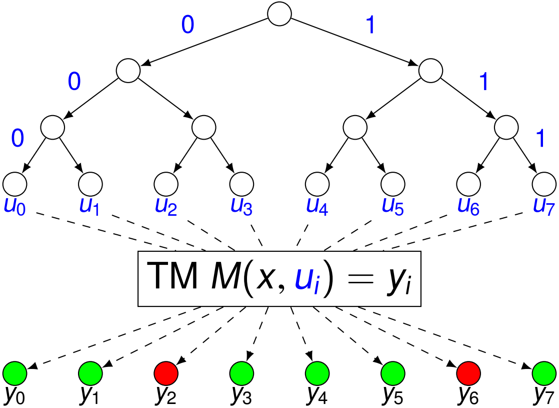
- **NP** $=$ **C**$[(0+1)^*1(0+1)^*, 0^*]$

Leaf languages



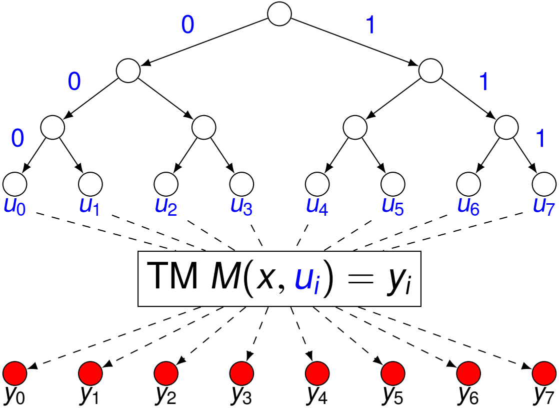
- **NP** $=$ **C**$[(0+1)^*1(0+1)^*, 0^*]$

Leaf languages



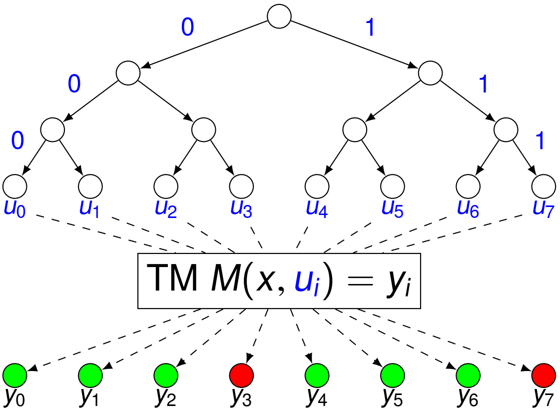
- **NP** $= \mathbf{C}[(0+1)^*1(0+1)^*, 0^*]$
- **RP** $= \mathbf{C}[\{w \in \{0,1\}^* \mid \frac{\#_1 w}{\#_0 w} \geq 3\}, 0^*]$

Leaf languages



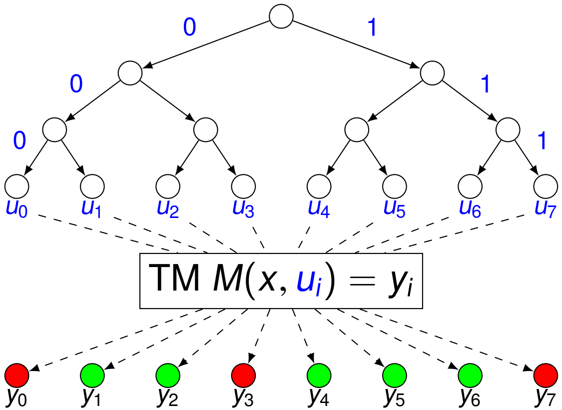
- **NP** $= \mathbf{C}[(0+1)^*1(0+1)^*, 0^*]$
- **RP** $= \mathbf{C}[\{w \in \{0,1\}^* \mid \frac{\#_1 w}{\#_0 w} \geq 3\}, 0^*]$

Leaf languages



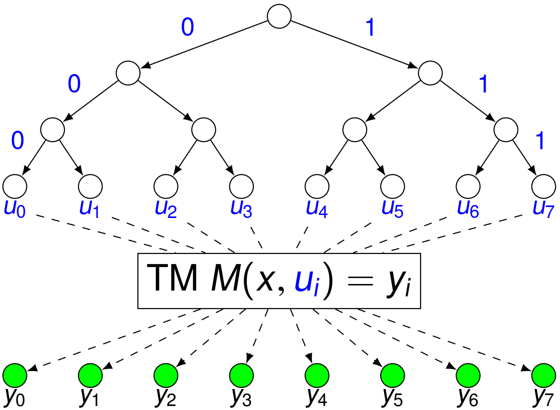
- **NP** $= \mathbf{C}[(0+1)^*1(0+1)^*, 0^*]$
- **RP** $= \mathbf{C}[\{w \in \{0,1\}^* \mid \frac{\#_1 w}{\#_0 w} \geq 3\}, 0^*]$
- **PP** $= \mathbf{C}[\{w \in \{0,1\}^* \mid \frac{\#_1 w}{\#_0 w} \geq 3\}, \{w \in \{0,1\}^* \mid \frac{\#_1 w}{\#_0 w} < 3\}]$

Leaf languages



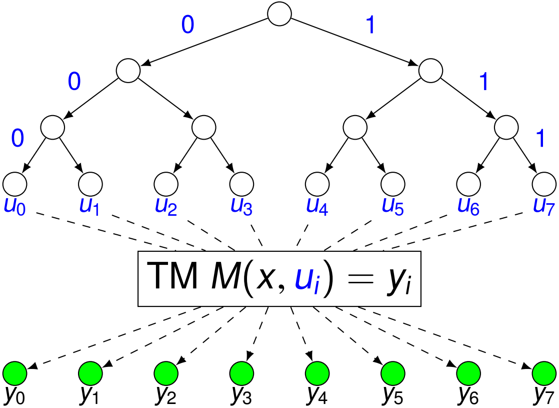
- **NP** $= \mathbf{C}[(0+1)^*1(0+1)^*, 0^*]$
- **RP** $= \mathbf{C}[\{w \in \{0,1\}^* \mid \frac{\#_1 w}{\#_0 w} \geq 3\}, 0^*]$
- **PP** $= \mathbf{C}[\{w \in \{0,1\}^* \mid \frac{\#_1 w}{\#_0 w} \geq 3\}, \{w \in \{0,1\}^* \mid \frac{\#_1 w}{\#_0 w} < 3\}]$

Leaf languages



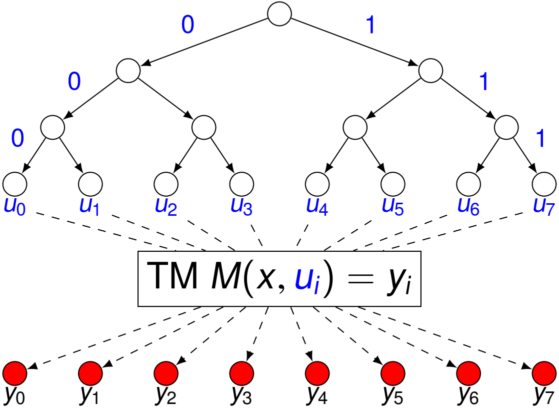
- What about **P**?

Leaf languages



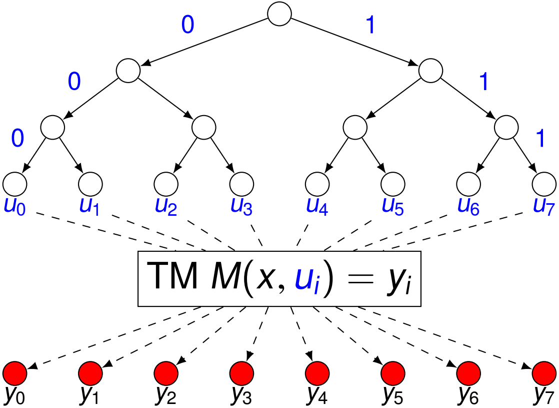
- What about **P**?
- $\mathbf{P} = \mathbf{C}[1(0+1)^*, 0(0+1)^*]$.

Leaf languages



- What about **P**?
- $\mathbf{P} = \mathbf{C}[1(0+1)^*, 0(0+1)^*]$.

Leaf languages



- What about **P**?
- **P** $=$ **C**$[1(0+1)^*, 0(0+1)^*]$.
- Certificate $0\dots0$ can always be used (compare this to **BPP**)

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 28, 2019

Lecture 14

Interactive Proofs

Overview

NP certificates or proof of membership

Overview

NP certificates or proof of membership

$\downarrow$

RP proofs chosen at random

Overview

NP	certificates or proof of membership
$\downarrow$	
RP	proofs chosen at random
$\downarrow$	
IP	interactive proofs between a prover and a verifier

Example: job interview, interactive vs. fixed questions

Agenda

- interactive proof examples
  - socks
  - graph coloring
  - graph non-isomorphism
- definition of interactive proof complexity
  - **IP**
  - public coins: **AM**

Different socks

Example

P wants to convince V that she has a red sock and a yellow sock.
V is blind and has a coin.

Interactive Proof

1. P tells V which sock is red
2. V holds red sock in her right hand, left sock in her yellow hand
3. P turns away from V
4. V tosses a coin
   4.1 heads: keep socks
   4.2 tails: switch socks
5. V asks P where the red sock is

Observations

- If P tells the truth (different colors), she will always answer correctly
- If P lies

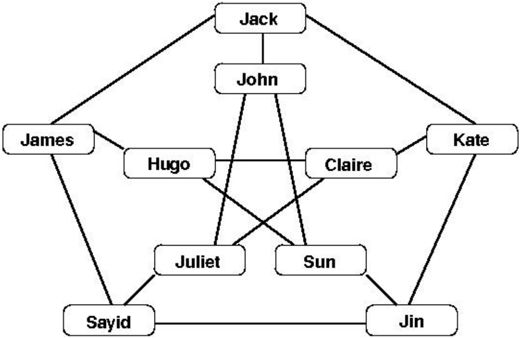
Observations

- If P tells the truth (different colors), she will always answer correctly
- If P lies
  - she can only answer correctly with probability $1/2$
  - after k rounds, she gets caught lying with probability $1 - 2^{-k}$

Observations

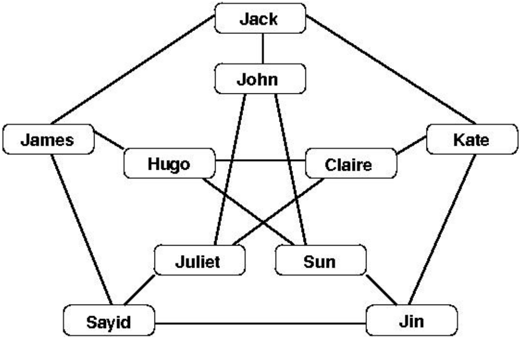
- If P tells the truth (different colors), she will always answer correctly
- If P lies
  - she can only answer correctly with probability $1/2$
  - after k rounds, she gets caught lying with probability $1 - 2^{-k}$
- random choices are crucial
- P has more computational power (vision) than V
- P must not see V's coin (private coin)

Graph 3-Coloring



- P claims: *G* is 3-colorable
- How can she prove it to V?

Graph 3-Coloring



- P claims: G is 3-colorable
- How can she prove it to V?
- provide certificate (since $3-\mathrm{Col} \in$ **NP**), V checks it
- possible for all $L \in$ **NP** with one round if P has **NP** power

What if actual coloring should be secret?

- given a graph (V, E) with $|V| = n$
- P claims 3-colorability
- P wants to convince V of coloring $c : V \to C \quad (= \{R, G, B\})$

What if actual coloring should be secret?

- given a graph (V, E) with $|V| = n$
- P claims 3-colorability
- P wants to convince V of coloring $c : V \to C \quad (= \{R, G, B\})$

1. P randomly picks a permutation $\pi : C \to C$ and puts $\pi(c(v_i))$ in envelope i for each $1 \leq i \leq n$
2. V randomly picks edge (u_i, u_j) and opens envelopes i and j to find colors c_i and c_j
3. V accepts iff $c_i \neq c_j$

Observations

- the protocol has two rounds
- a round is an uninterrupted sequence of messages from one party

Observations

- the protocol has two rounds
- a round is an uninterrupted sequence of messages from one party
- if G is not 3-colorable, P will be caught lying after $\mathcal{O}(n^3)$ rounds with probability $1 - 2^{-n}$

Observations

- the protocol has two rounds
- a round is an uninterrupted sequence of messages from one party
- if G is not 3-colorable, P will be caught lying after $\mathcal{O}(n^3)$ rounds with probability $1 - 2^{-n}$
- V learns nothing about the actual coloring

$\Rightarrow$ zero-knowledge protocol

- by reductions, all **NP** languages have ZK protocols

Observations

- the protocol has two rounds
- a round is an uninterrupted sequence of messages from one party
- if G is not 3-colorable, P will be caught lying after $\mathcal{O}(n^3)$ rounds with probability $1 - 2^{-n}$
- V learns nothing about the actual coloring

$\Rightarrow$ zero-knowledge protocol

- by reductions, all **NP** languages have ZK protocols
- private coins

Graph Non-Isomorphism

- **NP** languages have succinct, deterministic proofs
- **coNP** languages possibly don't
- graph isomorphism, GI, is in **NP**
- hence GNI $= \{\langle G_1, G_2 \rangle \mid G_1 \ncong G_2\}$ is in **coNP**
- GNI has a succinct interactive proof

Interactive Proof for GNI

given: graphs G_1, G_2

- **V** pick $i \in_R \{1, 2\}$, random permutation π
- **V** use π to permute nodes of G_i to obtain graph H
- **V** send H to P
- **P** check which of G_1, G_2 was used to obtain H
- **P** let G_j be that graph and send j to V
- **V** accept iff $i = j$

Intuition

- same idea as for socks protocol
- P has unlimited computational power
- if $G_1 \cong G_2$ then P answers correctly with probability at most $1/2$
- probability can be improved by sequential or parallel repetition
- if $G_1 \ncong G_2$ then P answers correctly with probability 1
- privacy of coins crucial

Agenda

- interactive proof examples ✓
  - socks ✓
  - graph coloring ✓
  - graph non-isomorphism ✓
- definition of interactive proof complexity
  - **IP**
  - public coins: **AM**

Interaction

Definition (Interaction)

Let $f, g : \{0,1\}^* \rightarrow \{0,1\}^*$ be functions and $k \geq 0$ an integer that may depend on the input size. A k-round interaction of f and g on input $x \in \{0,1\}^*$ is the sequence $\langle f, g\rangle(x)$ of strings $a_1, \ldots, a_k \in \{0,1\}^*$ defined by

$$\begin{array}{rcll} a_1 & = & f(x) & \\ a_2 & = & g(x, a_1) & \\ & \cdots & & \\ a_{2i+1} & = & f(x, a_1, \ldots, a_{2i}) & \text{for } 2i < k \\ a_{2i+2} & = & g(x, a_1, \ldots, a_{2i+1}) & \text{for } 2i+1 < k \end{array}$$

The output of f at the end of the interaction is defined by $out_f\langle f, g\rangle(x) = f(x, a_1, \ldots, a_k)$ and assumed to be in $\{0,1\}$.

This is a deterministic interaction, we need to add randomness.

Adding Randomness

Definition (IP)

For an integer $k \geq 1$ that may depend on the input size, a language L is in **IP**$[k]$, if there is a probabilistic polynomial-time TM V that can have a k-round interaction with a function $P : \{0,1\}^* \to \{0,1\}^*$ such that

- Completeness
  $x \in L \implies \exists P.Pr[out_V\langle V, P\rangle(x) = 1] \geq 2/3$
- Soundness
  $x \notin L \implies \forall P.Pr[out_V\langle V, P\rangle(x) = 1] \leq 1/3$

We define **IP** $= \bigcup_{c \geq 1}$ **IP**$[n^c]$.

- V has access to a random variable $r \in_R \{0,1\}^m$
- e.g. $a_1 = f(x, r)$ and $a_3 = f(x, a_1, r)$
- g cannot see r

$\Rightarrow$ $out_V\langle V, P\rangle(x)$ is a random variable where all probabilities are over the choice of r

Arthur-Merlin Protocols

Definition (AM)

- For every k the complexity class **AM**$[k]$ is defined as the subset of **IP**$[k]$ obtained when the verfier's messages are random bits only and also the only random bits used by V.
- **AM** $=$ **AM**$[2]$

Such an interactive proof is called an Arthur-Merlin proof or a public coin proof.

Agenda

- interactive proof examples ✓
  - socks ✓
  - graph coloring ✓
  - graph non-isomorphism ✓
- definition of interactive proof complexity
  - **IP** ✓
  - public coins: **AM** ✓

Basic Properties

- **NP** $\subseteq$ **IP**
- for every polynomial $p(n)$ the acceptance bounds in the definition of **IP** can be changes to
  - $2^{-p(n)}$for soundness
  - $1 - 2^{-p(n)}$ for completeness
- the requirement for completeness can be changed to require probability 1 yielding perfect completeness
- perfect soundness collapses **IP** to **NP**

What have we learnt?

- **IP**[k]: languages that have k-round interactive proofs
- interaction and randomization possibly add power
  - randomization alone: **BPP** (possibly equals **P**)
  - deterministic interaction: **NP**

  ⇒ interactive proofs more succinct
- prover has unlimited computational power
- verifier is a **BPP** machine (poly-time with coins)
- coins can be private or public
- zero-knowledge protocols do exist for all **NP** languages
- soundness and completeness thresholds can be adapted

What's next?

- **AM**$[2] =$ **AM**$[k]$ — AM hierarchy collapses
- **AM**$[k+2] =$ **IP**$[k]$ — private coins don't help
- if graph isomorphism is **NP**-complete, the polynomial hierarchy collapses
- **IP** $=$ **PSPACE**

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 28, 2019

Lecture 15

Public Coins and Graph (Non)Isomorphism

Goal and Plan

Goal

- understand public coins and their relation to private coins
- get a reason why graph isomorphism might not be **NP**-complete

Goal and Plan

Goal

- understand public coins and their relation to private coins
- get a reason why graph isomorphism might not be **NP**-complete

Plan

- show that graph non-isomorphism has a two round Arthur-Merlin proof; formally: GNI $\in$ **AM**[2]
- show that this implies GI is not **NP**-complete unless $\Sigma_2^p = \Pi_2^p$

Agenda

- **IP** and **AM** – recap
- graph non-isomorphism as a problem about set sizes
- tool: pairwise independent hash functions
- an **AM**[2] protocol for GNI
- improbability of **NP**-completeness of GI

IP

Definition (IP)

For an integer $k \geq 1$ that may depend on the input size, a language L is in **IP**$[k]$, if there is a probabilistic polynomial-time TM V that can have a k-round interaction with a function $P : \{0,1\}^* \to \{0,1\}^*$ such that

- Completeness
  $x \in L \implies \exists P.Pr[out_V\langle V, P\rangle(x) = 1] \geq 2/3$
- Soundness
  $x \notin L \implies \forall P.Pr[out_V\langle V, P\rangle(x) = 1] \leq 1/3$

We define **IP** $= \bigcup_{c \geq 1}$ **IP**$[n^c]$.

- V has access to a random variable $r \in_R \{0,1\}^m$
- e.g. $a_1 = f(x, r)$ and $a_3 = f(x, a_1, r)$
- g cannot see r

$\Rightarrow$ $out_V\langle V, P\rangle(x)$ is a random variable where all probabilities are

AM

Definition (AM)

- For every k the complexity class **AM**$[k]$ is defined as the subset of **IP**$[k]$ obtained when the verfier's messages are random bits only and also the only random bits used by V.
- **AM** $=$ **AM**$[2]$

Such an interactive proof is called an Arthur-Merlin proof or a public coin proof.

Agenda

- **IP** and **AM** – recap ✓
- graph non-isomorphism as a problem about set sizes
- tool: pairwise independent hash functions
- an **AM**[2] protocol for GNI
- improbability of **NP**-completeness of GI

Recasting GNI

- let G_1, G_2 be graphs with nodes $\{1, \ldots, n\}$ each
- we define a set S such that
  - if $G_1 \cong G_2$ then $|S| = n!$
  - if $G_1 \ncong G_2$ then $|S| = 2n!$

Recasting GNI

- let G_1, G_2 be graphs with nodes $\{1, \ldots, n\}$ each
- we define a set S such that
  - if $G_1 \cong G_2$ then $|S| = n!$
  - if $G_1 \ncong G_2$ then $|S| = 2n!$
- idea: S is the set of graphs that are isomorphic to G_1 OR to G_2
- if $G_1 \cong G_2$, this set is small, otherwise not

Recasting GNI

- let G_1, G_2 be graphs with nodes $\{1, \ldots, n\}$ each
- we define a set S such that
  - if $G_1 \cong G_2$ then $|S| = n!$
  - if $G_1 \ncong G_2$ then $|S| = 2n!$
- idea: S is the set of graphs that are isomorphic to G_1 OR to G_2
- if $G_1 \cong G_2$, this set is small, otherwise not
- problem: automorphisms
  - an automorphism of G_1 is a permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that $\pi(G) = G$
  - all automorphisms of graph G written $aut(G)$

The infamous set S

$$S = \{(H, \pi) \mid H \cong G_1 \text{ or } H \cong G_2, \pi \in aut(H)\}$$

The infamous set S

$$S = \{(H, \pi) \mid H \cong G_1 \text{ or } H \cong G_2, \pi \in aut(H)\}$$

- to convince the verifier that $G_1 \not\cong G_2$ the prover has to convince the verifier that $|S| = 2n!$ rather than $n!$
- that is the verifier should accept with high probability if $|S| \geq K$ for some K
- it should reject if $|S| \leq \frac{K}{2}$

Agenda

- **IP** and **AM** – recap ✓
- graph non-isomorphism as a problem about set sizes ✓
- tool: pairwise independent hash functions
- an **AM**[2] protocol for GNI
- improbability of **NP**-completeness of GI

Hash functions

- goal: store a set $S \subseteq \{0,1\}^m$ to efficiently answer membership $x \in S$
- S could change dynamically
- $|S|$ much smaller than 2^m, possibly around 2^k for $k \leq m$

Hash functions

- goal: store a set $S \subseteq \{0,1\}^m$ to efficiently answer membership $x \in S$
- S could change dynamically
- $|S|$ much smaller than 2^m, possibly around 2^k for $k \leq m$
- to create a hash table of size 2^k
  - select a hash function $h : \{0,1\}^m \to \{0,1\}^k$
  - store x at $h(x)$
- collision: $h(x) = h(y)$ for $x \neq y$
- choosing hash functions randomly from a collection, one can expect h to be almost bijective if $|S| \approx 2^k$

Pairwise independent hash functions

Definition

Let $\mathcal{H}_{m,k}$ be a collection of functions from $\{0,1\}^m$ to $\{0,1\}^k$. We say that $\mathcal{H}_{m,k}$ is pairwise independent if

- for every $x \neq x' \in \{0,1\}^m$ and
- for every $y, y' \in \{0,1\}^k$ and

$Pr_{h \in_R \mathcal{H}_{m,k}}[h(x) = y \wedge h(x') = y'] = 2^{-2k}$

- when h is choosen randomly $(h(x), h(x'))$ is distributed uniformly over $\{0,1\}^k \times \{0,1\}^k$
- such collections exist
- here: we only assume the existence

Agenda

- **IP** and **AM** – recap ✓
- graph non-isomorphism as a problem about set sizes ✓
- tool: pairwise independent hash functions ✓
- an **AM**[2] protocol for GNI
- improbability of **NP**-completeness of GI

Goldwasser-Sipser Set Lower Bound Protocol

- $S \subseteq \{0,1\}^m$
- both parties know a K
- prover wants to convince verifier that $|S| \geq K$
- verifier rejects with high probability if $|S| \leq \frac{K}{2}$
- let k be an integer such that $2^{k-2} < K \leq 2^{k-1}$

Goldwasser-Sipser Set Lower Bound Protocol

The following protocol has two rounds and uses public coins!

V
- randomly choose $h : \{0,1\}^m \to \{0,1\}^k$ from a pairwise independent collection of hash functions $\mathcal{H}_{m,k}$
- randomly choose $y \in \{0,1\}^k$
- send h and y to prover

P
- find an $x \in S$ such that $h(x) = y$
- send x to V together with a certificate of membership of x in S

V if $h(x) = y$ and $x \in S$ accept; otherwise reject

Why the protocol works?

Intuition: If S is big enough (non-isomorphic case) then the prover has a good chance to find a pre-image.

Why the protocol works?

Intuition: If S is big enough (non-isomorphic case) then the prover has a good chance to find a pre-image.

Formally:

- show that there exists a $\hat{p}$ such that
  - if $|S| \geq K$ then $Pr[\exists x \in S.h(x) = y]$ is greater than $\frac{3}{4}\hat{p}$
  - if $|S| \leq \frac{K}{2}$ then $Pr[\exists x \in S.h(x) = y]$ is lower than $\frac{\hat{p}}{2}$
- this is a probability gap which can be amplified by repetition
- one can choose $\hat{p} = \frac{K}{2^k}$
  - soundness: easy (not enough elements even if injective)
  - completeness: by inclusion-exclusion principle
    $\geq \sum_x Pr[h(x) = y] - \frac{1}{2} \sum_{x \neq x} Pr[h(x) = y, h(x') = y]$
    by pairwise independence $\frac{|S|}{2^k} - \frac{|S|^2}{2^{2k+1}} \geq \frac{3}{4}\hat{p}$

Putting it together

AM[2] public coin protocol for GNI

- compute S (automorphisms) as above
- prover and verifier run set lower bound protocol several times
- verifier accepts by majority vote
- using Chernoff bounds, this gives the desired completeness and soundness probabilities
- observe: only a constant number of iterations necessary which can be executed in parallel

$\Rightarrow$ number of rounds stays at 2

Details: Arora-Barak, section 8.2

Agenda

- **IP** and **AM** – recap ✓
- graph non-isomorphism as a problem about set sizes ✓
- tool: pairwise independent hash functions ✓
- an **AM**[2] protocol for GNI ✓
- improbability of **NP**-completeness of GI

Graph Isomorphism

Theorem

If GI $= \{\langle G_1, G_2\rangle \mid G_1 \cong G_2\}$ *is* **NP**-*complete then* $\mathbf{\Sigma_2^p} = \mathbf{\Pi_2^p}$.

Proof idea ($\mathbf{\Sigma_2^p} \subseteq \mathbf{\Pi_2^p}$):

- $\exists\vec{x}\forall\vec{y}\ \varphi(x, y)$ equivalent to
- $\exists\vec{x}\ g(x) \in$ GNI equivalent to (GNI $\in$ **AM**)
- $\exists\vec{x}\forall\vec{r}\exists\vec{m}\ A(g(x), r, m) = 1$ equivalent to
- $\forall\vec{r}\exists\vec{x}\exists\vec{m}\ A(g(x), r, m) = 1$
  (perfect completeness $\Longrightarrow$ satisfiable
  soundness with 2^{-n} $\Longrightarrow$ single string r)

What have we learnt?

- graph isomorphism is not **NP**-complete unless the (polynomial) hierarchy collapses
- public coins are as expressive as private coins
  - proof of GNI $\in$ **AM**[2] generalizes to **IP**$[k] =$ **AM**$[k+2]$ (without proof)
  - one can also show **AM**$[k] =$ **AM**$[k+1]$ for $k \geq 2$ (collapse)
    intuitively **AM** more powerful than **MA**, because in **AM** Merlin gets to look at the random bits before deciding on his answer
  - also not shown: perfect completeness for **AM**
- Goldwasser-Sipser set lower bound protocol (in **AM**[2])
- hash functions as a useful tool

Up next: **IP** $=$ **PSPACE**

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 28, 2019

Lecture 16

IP $=$ PSPACE

Goal and Plan

Goal

- **IP** $=$ **PSPACE**

Plan

1. **PSPACE** $\subseteq$ **IP** by showing QBF $\in$ **IP**
2. **IP** $\subseteq$ **PSPACE** by computing optimal prover strategies in polynomial space

Agenda

- arithmetization of Boolean formulas
- arithmetization of quantified formulas by linearization
- interactive protocol for QBF

Agenda

- arithmetization of Boolean formulas
- arithmetization of quantified formulas by linearization
- interactive protocol for QBF

Afternoon

- optimal prover strategy to show **IP** $\subseteq$ **PSPACE**
- a note on graph isomorphism
- summary: interactive proofs incl. further reading and context
- outlook: approximation and PCP theorem

Proof Idea

Show that QBF $\in$ **IP**.

This implies **PSPACE** $\subseteq$ **IP** because

Proof Idea

Show that QBF $\in$ **IP**.

This implies **PSPACE** $\subseteq$ **IP** because

- QBF is **PSPACE**-complete
- **IP** closed under polynomial reductions

Proof Idea

Show that QBF $\in$ **IP**.

This implies **PSPACE** $\subseteq$ **IP** because

- QBF is **PSPACE**-complete
- **IP** closed under polynomial reductions

Technique
Turn formulas into polynomials, similar to reduction from 3SAT to ILP: arithmetization.

Setting

- let $\Phi = Q_1 x_1 \dots Q_n x_n \varphi(x_1, \dots, x_n)$ be a quantified boolean formula, where φ is in 3CNF with m clauses
- Φ is either true or false
- running example: $\Phi_= = \forall x \exists y \ (x \vee \overline{y}) \wedge (\overline{x} \vee y)$, where the body is written $\varphi_=$
- deciding truth value of Φ is **PSPACE**-complete

Observation

- $x \wedge y$ is satisfied iff $x \cdot y = 1$ for $x, y \in \{0, 1\}$

Observation

- $x \wedge y$ is satisfied iff $x \cdot y = 1$ for $x, y \in \{0, 1\}$
- $\overline{x}$ is satisfied iff $1 - x = 1$

Observation

- $x \wedge y$ is satisfied iff $x \cdot y = 1$ for $x, y \in \{0, 1\}$
- $\overline{x}$ is satisfied iff $1 - x = 1$
- $x \vee y$ is satisfied iff $x + y \geq 1$

Observation

- $x \wedge y$ is satisfied iff $x \cdot y = 1$ for $x, y \in \{0, 1\}$
- $\overline{x}$ is satisfied iff $1 - x = 1$
- $x \vee y$ is satisfied iff $x + y \geq 1$
- note that $x \vee y \equiv x \wedge \overline{y} \vee \overline{x} \wedge y \vee x \wedge y$

Observation

- $x \wedge y$ is satisfied iff $x \cdot y = 1$ for $x, y \in \{0, 1\}$
- $\overline{x}$ is satisfied iff $1 - x = 1$
- $x \vee y$ is satisfied iff $x + y \geq 1$
- note that $x \vee y \equiv x \wedge \overline{y} \vee \overline{x} \wedge y \vee x \wedge y$

$\Rightarrow$ $x \vee y$ is satisfied iff $x + y - xy = 1$

Arithmetization of Boolean formulas

For Boolean formula $\varphi(x_1, \ldots, x_n)$ we define $ari_\varphi(x_1, \ldots, x_n)$ such that $\varphi(x_1, \ldots, x_n)$ is satisfied iff $ari_\varphi(x_1, \ldots, x_n)$ is 1 for satisfying assignment of x_i to true/false and the corresponding x_i.

Arithmetization of Boolean formulas

$$\begin{aligned}
ari_{x_i}(x_1,\ldots,x_n) &= x_i \\
ari_{\overline{\varphi}}(x_1,\ldots,x_n) &= 1 - ari_{\varphi}(x_1,\ldots,x_n) \\
ari_{\varphi_1\wedge\varphi_2}(x_1,\ldots,x_n) &= ari_{\varphi_1}(x_1,\ldots,x_n)\cdot ari_{\varphi_2}(x_1,\ldots,x_n) \\
ari_{\varphi_1\vee\varphi_2}(x_1,\ldots,x_n) &= ari_{\varphi_1}(x_1,\ldots,x_n)+ari_{\varphi_2}(x_1,\ldots,x_n) \\
&\quad -ari_{\varphi_1}(x_1,\ldots,x_n)\cdot ari_{\varphi_2}(x_1,\ldots,x_n)
\end{aligned}$$

Arithmetization of Boolean formulas

$$
\begin{aligned}
ari_{x_i}(x_1,\dots,x_n) &= x_i \\
ari_{\overline{\varphi}}(x_1,\dots,x_n) &= 1 - ari_{\varphi}(x_1,\dots,x_n) \\
ari_{\varphi_1 \wedge \varphi_2}(x_1,\dots,x_n) &= ari_{\varphi_1}(x_1,\dots,x_n) \cdot ari_{\varphi_2}(x_1,\dots,x_n) \\
ari_{\varphi_1 \vee \varphi_2}(x_1,\dots,x_n) &= ari_{\varphi_1}(x_1,\dots,x_n) + ari_{\varphi_2}(x_1,\dots,x_n) \\
&\quad - ari_{\varphi_1}(x_1,\dots,x_n) \cdot ari_{\varphi_2}(x_1,\dots,x_n)
\end{aligned}
$$

Example

$\varphi_= = (x \vee \overline{y}) \wedge (\overline{x} \vee y)$

$$
\begin{aligned}
ari_{\varphi_=}(x,y) &= (x + (1-y) - x(1-y)) \cdot ((1-x) + y - (1-x)y) \\
&= (1 - y + xy) \cdot (1 - x + xy) \\
&= 1 - x - y + 3xy - xy^2 - x^2y + x^2y^2 \\
&=: f_=(x,y)
\end{aligned}
$$

Observation

- degree of arithmetization is $\leq 3m$
- crucial for polynomial representation of formulas

What about quantification?

Intuition

- universal quantification corresponds to conjunction corresponds to multiplication
- existential quantification corresponds to disjunction corresponds to addition

What about quantification?

Intuition

- universal quantification corresponds to conjunction corresponds to multiplication
- existential quantification corresponds to disjunction corresponds to addition

- $ari_{\forall x_i.\varphi}(x_1, \ldots, x_i, \ldots, x_n)$ equals $ari_\varphi(x_1, \ldots, 0, \ldots, x_n) \cdot ari_\varphi(x_1, \ldots, 1, \ldots, x_n)$
- $ari_{\exists x_i.\varphi}(x_1, \ldots, x_i, \ldots, x_n)$ equals $ari_\varphi(x_1, \ldots, 0, \ldots, x_n) + ari_\varphi(x_1, \ldots, 1, \ldots, x_n) - ari_\varphi(x_1, \ldots, 0, \ldots, x_n) \cdot ari_\varphi(x_1, \ldots, 1, \ldots, x_n)$

Running Example

Example

$$
\begin{aligned}
ari_{\Phi_=}(x, y) &= ari_{\exists y.\varphi_=}(0, y) \cdot ari_{\exists y.\varphi_=}(1, y) \\
&= (f_=(0,0) + f_=(0,1) - f_=(0,0) f_=(0,1)) \cdot \ldots \\
&= \ldots \\
&= 1
\end{aligned}
$$

Lessons learnt

- $\Phi_=$ is true
- degree of polynomial might get exponential in n
- coefficients too

Lessons learnt

- $\Phi_=$ is true
- degree of polynomial might get exponential in n
- coefficients too

Rescue

- over $\{0, 1\}$ we have $x^c = x$
- gives rise to linearization
- to get rid of large coefficients: compute over some sufficiently small finite field

Agenda

- arithmetization of Boolean formulas ✓
- arithmetization of quantified formulas by linearization
- interactive protocol for QBF

Linearization

Linearization means reducing all exponents in polynomial to 1.

- $L_y(f(x,y)) = f(x,1) \cdot y + f(x,0) \cdot (1-y)$
- $L_y(f(x,y)$ is linear in y
- $L_y(f(x,y))$ is equivalent to $f(x,y)$ over $\{0,1\}^2$

Example

$$
\begin{aligned}
L_y(f_=(x,y)) &= L_y(1-x-y+3xy-xy^2-x^2y+x^2y^2) \\
&= (1-y)(1-x)+y\cdot(-x+3x-x-x^2+x^2) \\
&= 1-x-y+2xy
\end{aligned}
$$

General form

$$\begin{aligned} L_j(f(x_1,\ldots,x_j,\ldots,x_n)) &= f(x_1,\ldots,1,\ldots,x_k)x_j \\ &+ f(x_1,\ldots,0,\ldots,x_k)(1-x_j) \end{aligned}$$

Arithmetization

1. arithmetize Boolean body of formula
2. linearize all variables
3. for innermost quantifier apply $ari_\forall x$ (resp. $ari_\exists x$)
4. repeat from 2.

Recursive definition of general arithmetization

$$f_{n,n}(x_1, \ldots, x_n) := ari_\varphi(x_1, \ldots, x_n)$$

$$f_{i,j}(x_1, \ldots, x_i) = L_{j+1}(f_{i,j+1}(x_1, \ldots, x_i))$$

$$f_{i,i}(x_1, \ldots, x_i) := f_{i+1,0}(x_1, \ldots, x_i, 0) f_{i+1,0}(x_1, \ldots, x_i, 1)$$

if x_{i+1} universal

$$f_{i,i}(x_1, \ldots, x_i) := f_{i+1,0}(x_1, \ldots, x_i, 0) + f_{i+1,0}(x_1, \ldots, x_i, 1) - f_{i+1,0}(x_1, \ldots, x_i, 0) f_{i+1,0}(x_1, \ldots, x_i, 1)$$

if x_{i+1} existential

Observations

- there are $\mathcal{O}(n^2)$ functions $f_{\cdot,\cdot}$
- functions $f_{n,\cdot}$ have degree at most $3m$
- all other functions have degree of each variable at most 2
- $f_{0,0} = 1$ iff $\Phi \in$ QBF

Agenda

- arithmetization of Boolean formulas ✓
- arithmetization of quantified formulas by linearization ✓
- interactive protocol for QBF

Protocol intuition

- V accepts if $f_{0,0} = 1$
- P needs to convince V of that fact by iterating over all $f_{i,j}$
- V challenges P by choosing random values from a finite field
- P inserts these values into polynomials and return linear function
- V checks that functions adhere to recursive scheme

Initialization

- verifier and prover agree on prime p such that $12|\Phi|^2 < p \in \mathcal{O}(|\Phi|^2)$
- all polynomials will be computed in $\mathbb{Z}/p\mathbb{Z}$
- this is a range, where linear functions can be polynomially represented and evaluated
- start: P sends $f_{0,0}$, the prime and the primality proof
- if $f_{0,0} = 1$ then iterate from $i = 1$ and $j = 0$ until both reach n; otherwise reject

$\Rightarrow$ $\mathcal{O}(n^2)$ rounds

Quantor case $j = 0$

- V asks for $f_{i,0}(r_1, \ldots, r_{i-1}, x_i)$
- P sends $f_{i,0}(r_1, \ldots, r_{i-1}, x_i)$
- if x_i is universally quantified, V checks whether

$$f_{i,0}(r_1, \ldots, r_{i-1}, 0) f_{i,0}(r_1, \ldots, r_{i-1}, 1) \equiv_p f_{i-1,i-1}(r_1, \ldots, r_{i-1})$$

- if x_i is existentially quantified, V checks

$$f_{i,0}(r_1, \ldots, r_{i-1}, 0) + f_{i,0}(r_1, \ldots, r_{i-1}, 1) - f_{i,0}(r_1, \ldots, r_{i-1}, 0) f_{i,0}(r_1, \ldots, r_{i-1}, 1) \equiv_p f_{i-1,i-1}(r_1, \ldots, r_{i-1})$$

- V picks random number $r_i \in \mathbb{Z}/p\mathbb{Z}$ and set j to 1

Linearization case $j > 0$

- V asks for $f_{i,j}(r_1, \ldots, x_j, \ldots, r_i)$
- P sends $f_{i,j}(r_1, \ldots, x_j, \ldots, r_i)$
- V checks

$$\begin{gathered}(1 - r_j) f_{i,j}(r_1, \ldots, 0, \ldots, r_i) + \\ r_j f_{i,j}(r_1, \ldots, 1, \ldots, r_i) \\ \equiv_p \\ f_{i,j-1}(r_1, \ldots, r_i)\end{gathered}$$

- V picks r_j at random and increases j (or sets j to 0 and increases i)

Finally . . .

P tests whether

$$ari_{\varphi}(r_1, \ldots, r_n) \equiv_p f_{n,n}(r_1, \ldots, r_n)$$

Observations

- P only sends linear functions
- total message length still polynomial
- V can compute linear functions in $\mathbb{Z}/p\mathbb{Z}$
- if $\Phi \in$ QBF P can always convince V by sending correct polynomials

$\Rightarrow$ perfect completeness

- we have public coins

What if $\Phi \notin$ QBF?

An honest prover admits this fact.

A cheating prover can try to send forged polynomials $g_{i,j}(x)$ instead of $f_{i,j}(x_1, \ldots, x, \ldots, x_i)$.

For soundness P must fail to convince V with high probability.

Soundness

- P can cheat in round (i,j) iff $f_{i,j}(x_1, \ldots, x, \ldots, x_i) - g_{i,j}(x) \equiv_p 0$
- that is: iff V by chance picks a root r_k of a polynomial

Soundness

- P can cheat in round (i,j) iff $f_{i,j}(x_1, \ldots, x, \ldots, x_i) - g_{i,j}(x) \equiv_p 0$
- that is: iff V by chance picks a root r_k of a polynomial
- probability to do so in round (i,j) is $q_{i,j} \leq deg(f_{i,j})/p$ since polynomials of degree n have at most n roots

Soundness

- P can cheat in round (i,j) iff $f_{i,j}(x_1, \ldots, x, \ldots, x_i) - g_{i,j}(x) \equiv_p 0$
- that is: iff V by chance picks a root r_k of a polynomial
- probability to do so in round (i, j) is $q_{i,j} \leq deg(f_{i,j})/p$ since polynomials of degree n have at most n roots
- $f_{n,\cdot}$ have degree at most $3m$
- $f_{i<n,\cdot}$ have degree at most 2
- there are $(n+1)(n+2)/2$ polynomials, $n+1$ large ones

$$
\begin{aligned}
Pr[\text{P cheats}] &\leq \Sigma_{i=1}^{n} \Sigma_{j=0}^{i} q_{i,j} \\
&\leq \frac{3m(n+1)}{p} + \frac{2n(n+1)}{2p} \\
&\leq \frac{4|\Phi|^2}{p} \\
&\leq 1/3
\end{aligned}
$$

Agenda

- arithmetization of Boolean formulas ✓
- arithmetization of quantified formulas by linearization ✓
- interactive protocol for QBF ✓

Agenda

- arithmetization of Boolean formulas ✓
- arithmetization of quantified formulas by linearization ✓
- interactive protocol for QBF ✓

Afternoon

- optimal prover strategy to show **IP** $\subseteq$ **PSPACE**
- a note on grpah isomorphism
- summary: interactive proofs incl further reading and context
- outlook: approximation and PCP theorem
- evaluation

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 28, 2019

Lecture 17

IP $=$ PSPACE (2)

Goal and Plan

Goal

- **IP** $=$ **PSPACE**

Plan

1. **PSPACE** $\subseteq$ **IP** by showing QBF $\in$ **IP** $\checkmark$
2. **IP** $\subseteq$ **PSPACE** by computing optimal prover strategies in polynomial space

Agenda

- optimal prover strategy to show **IP** $\subseteq$ **PSPACE**
- summary and further reading
- outlook: approximation and PCP theorem

Definition recap

L is in **IP** iff

1. there exists a polynomial p and
2. there exists a poly-time, randomized verifier V

Definition recap

L is in **IP** iff

1. there exists a polynomial p and
2. there exists a poly-time, randomized verifier V

such that for all words $x \in \{0,1\}^*$ holds

- if $x \in L$ then there exists a prover P such that $Pr[out_V\langle P, V\rangle(x) = 1] \geq 2/3$
- if $x \notin L$ then for all provers P holds that $Pr[out_V\langle P, V\rangle(x) = 1] \leq 1/3$

Definition recap

L is in **IP** iff

1. there exists a polynomial p and
2. there exists a poly-time, randomized verifier V

such that for all words $x \in \{0,1\}^*$ holds

- if $x \in L$ then there exists a prover P such that $Pr[out_V\langle P, V\rangle(x) = 1] \geq 2/3$
- if $x \notin L$ then for all provers P holds that $Pr[out_V\langle P, V\rangle(x) = 1] \leq 1/3$

Moreover, the following is bounded by $p(|x|)$

- the number of random bits chosen by V
- the number of rounds
- the length of each message

Optimal Prover

Let $L \in$ **IP** be arbitrary, we need to show that $L \in$ **PSPACE**.

Optimal Prover

Let $L \in$ **IP** be arbitrary, we need to show that $L \in$ **PSPACE**.

We know that there exist V and p according to definition on previous slide.

Optimal Prover

Let $L \in$ **IP** be arbitrary, we need to show that $L \in$ **PSPACE**.

We know that there exist V and p according to definition on previous slide.

For $x \in \{0, 1\}^n$, we need to compute in polynomial space whether $x \in L$ or $x \notin L$.

Optimal Prover

Let $L \in$ **IP** be arbitrary, we need to show that $L \in$ **PSPACE**.
We know that there exist V and p according to definition on previous slide.
For $x \in \{0,1\}^n$, we need to compute in polynomial space whether $x \in L$ or $x \notin L$.

$$z := \max_P \{Pr[out_V\langle P, V\rangle(x) = 1] \mid P \text{ is any prover for } L\}$$

Optimal Prover

Let $L \in$ **IP** be arbitrary, we need to show that $L \in$ **PSPACE**.
We know that there exist V and p according to definition on previous slide.
For $x \in \{0,1\}^n$, we need to compute in polynomial space whether $x \in L$ or $x \notin L$.

$$z := \max_P \{Pr[out_V\langle P, V\rangle(x) = 1] \mid P \text{ is any prover for } L\}$$

z is acceptance probability of optimal prover, inducing the error probability.

Optimal Prover

Let $L \in$ **IP** be arbitrary, we need to show that $L \in$ **PSPACE**.
We know that there exist V and p according to definition on previous slide.
For $x \in \{0,1\}^n$, we need to compute in polynomial space whether $x \in L$ or $x \notin L$.

$$z := \max_P \{Pr[out_V\langle P, V\rangle(x) = 1] \mid P \text{ is any prover for } L\}$$

z is acceptance probability of optimal prover, inducing the error probability.

- if $z \leq 1/3$ then $x \notin L$
- if $z \geq 2/3$ then $x \in L$
- since $L \in$ **IP** other z cannot occur
- maximum taken over finitely many provers for a given x

Recursive computation of z

If we can compute z in polynomial space, we are done.

Recursive computation of z

If we can compute z in polynomial space, we are done.

Recursive algorithm:

- simulate V branching on
  - each random choice of V
  - each possible response of P
- count
  - accepting branches produced by P's optimal response
  - total number of branches
- ratio is z

Doable in polynomial space?

- recursion depth: $p(n)$
- total number of branches: $p(n)^{p(n)}$

$\Rightarrow$ requires polynomially many bits only

- can manage both counters and current branch with a **PSPACE** machine

Agenda

- optimal prover strategy to show **IP** $\subseteq$ **PSPACE** ✓
- summary and further reading
- outlook: approximation and PCP theorem

Summary

- IP $=$ PSPACE
- PSPACE has short interactive proofs (certificates)
- proof of IP $\supseteq$ PSPACE also showed that we can have
  - public coins
  - perfect completeness

  for each $L \in$ IP
- interaction plus randomization seem to add power, whereas each in isolation seemingly does not

Further Reading

- interactive proofs defined in 1985 by *Goldwasser, Micali, Rackoff*. The knowledge complexity of interactive proof systems. SIAM Journal on Computing archive. Volume 18 (1)(1989).
- public coins: *L. Babai* Trading group theory for randomness. STOC 1985.
- survey book: *Oded Goldreich* Computational Complexity. A Conceptual Perspective. `http://www.wisdom.weizmann.ac.il/˜oded/cc-drafts.html`

Further Reading

- *Adi Shamir*. IP=PSPACE. Journal of the ACM v.39 n.4, p.878-880.
- outline here followed lecture notes from Brown university: A detailed proof that IP=PSPACE. `http://www.cs.brown.edu/courses/gs019/papers/ip.pdf`
- also nice: Michael Sipser's book Introduction to the Theory of Computation
- essentially covered 8.1 and 8.2 from Arora-Barak book
- an entertaining survey about the development in the beginning of the 90s by *L. Babai*. Transparent proofs and limits to approximations. First European Congress of Mathematicians. 1994.

Outlook

In the beginning of the 90s a lot of things happened quickly...

- Shamir proved that **IP** $=$ **PSPACE**
- one can also allow multiple provers which leads to the complexity class **MIP**
- one accepts only if provers agree
- **MIP** $=$ **NEXP**
- lead to the notion of **PCP**$[q, r]$, where one checks only r entries in a table of answer/query pairs of size 2^q
- it was then shown that **PCP**$[poly, poly]$ $=$ **NEXP** and **PCP**$[\log n, \mathcal{O}(1)]$ $=$ **NP**
- which yields strong results about approximation of **NP**-complete problems
- for instance: consider a $7/8$ approximation of 3SAT

Block structure of lecture

- basic complexity classes
- probabilistic TMs and randomization
- interactive proofs
- approximations and PCP
- parallelization
  - **NC**
  - circuits
  - descriptive complexity

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

May 28, 2019

Lecture 18

Approximation

Approximations

Goal

- decision $\rightarrow$ optimization
- formal definition of approximation
- hardness of approximation

Plan

- vertex cover: VC
- set cover: SC
- travelling salesman problem: TSP

Planes

Example

Given a set of airports, S, assign gas stations to a smallest subset, C, where planes can cover at most two legs without re-filling.

Formal model

- airports $\sim$ nodes in a graph
- legs $\sim$ undirected edges
- find a smallest set of nodes that covers all edges
- important problem in networks

Vertex Cover

Definition (Cover)

Let $G = (V, E)$ be an undirected graph. A set $C \subseteq V$ is a cover of S if

$$\forall (u, v) \in E.\ u \in C \ \vee \ v \in C$$

Decision problem

$$\mathrm{VC} = \{\langle G, k\rangle \mid G \text{ has a cover } C \text{ and } |C| \leq k\}$$

Optimization problem Min – VC

- given: $G = (V, E)$ undirected
- find: a minimal cover C

MinVC is NP-hard

Observation

- C is a cover iff $V \setminus C$ is an independent set.
- C is a minimal cover iff $V \setminus C$ is a maximal independent set.

Proof

- $\forall(u, v).\ u \in C \lor v \in C$

$\Leftrightarrow$ $\forall(u, v).\ u \notin V \setminus C \lor v \notin V \setminus C$

$\Leftrightarrow$ $\neg\exists(u, v).\ u \in V \setminus C \land v \in V \setminus C$

Some optimization problems

- many decision problems we have seen have optimization versions
- both minimization and maximization
- algorithms return best solution with respect to optimization parameter ρ

Examples

problem	min/max	parameter
3SAT	max	number of satisfiable clauses
Indset	max	size of independent set
VC	min	size of cover

Approximation

Computing precise solutions is often **NP**-hard for decision and optimization.

Instead of optimal solutions, in practice it often suffices to come up with approximations.

Definition (ρ-approximation)

A ρ-approximation for a minimization (maximization) problem with optimal solution O, returns a solution that is $\leq \rho O$ ($\geq \rho O$).

Note: ρ may depend on input size.

VC **approximation algorithm**

1. $C \leftarrow \emptyset$
2. **while** C **not** a cover
3. **pick** $(u, v) \in E$ s.t. $u, v \notin C$
4. $C \leftarrow C \cup \{u, v\}$
5. **return** C

Theorem

Algorithm runs in polynomial time and returns a 2-approximation.

Proof Edges picked contain no common vertices. Optimal vertex cover must contain at least one of the nodes, where the algorithm adds both.

Teams

Example

All your friends belong to one or several teams. You want to invite all of them but team-wise. What is the least number of invitations necessary?

Set Cover

- given: finite set U and a family $\mathcal{F}$ of subsets that covers U: $\bigcup \mathcal{F} \supseteq U$
- find: a smallest family $C \subseteq \mathcal{F}$ that covers U

Set Cover is NP-hard

Proof by reduction from vertex cover.

- let $G = (V, E)$ be an undirected graph
- $f(G) = (E, \mathcal{F})$
- $\mathcal{F} = \{E_v \mid v \in V\}$
- $E_v = \{\{u, v\} \in E\}$

Greedy algorithm for SC

1. $C \leftarrow \emptyset$, $U' \leftarrow U$
2. **while** $U' \neq \emptyset$
3. **pick** $S \in \mathcal{F}$ **maximizing** $|S \cap U'|$
4. $C \leftarrow C \cup \{S\}$
5. $U' \leftarrow U' \setminus S$
6. **return** C

- greedy algorithms pick the best local options
- algorithm runs in polynomial time

Roadmap

Just seen

- vertex cover
- 2-approximation algorithm for VC
- set cover
- approximation algorithm

Up next

- show that algorithm is a $\ln n$ approximation
- show that algorithm is a $\ln |S|$ approximation for largest set S
- TSP

What is the approximation ratio?

Need to compare result returned by algorithm with the unknown optimal solution

Observation If U has a k cover, then every subset of U has a k cover too!

Consequence Each step of greedy algorihm covers at least $1/k$ of the uncovered elements!

First bound: $\ln n$

- let $S_1, \ldots, S_t$ be the sequence of sets picked by algorithm
- let U_i be U' after i stages (uncovered)
- observe: $|U_{i+1}| = |U_i \setminus S_{i+1}| \leq |U_i|(1 - 1/k)$
- hence: $|U_{ik}| \leq |U_0|(1 - 1/k)^{ik} \leq \frac{|U|}{e^i}$
- thus $e^{\frac{t-1}{k}} \leq \frac{|U|}{|U_{t-1}|} \leq n$
- therefore: $t \leq k \ln(n) + 1$

Note: The bound depends on the input length. We say that the greedy algorithm approximates SC to within a logarithmic factor.

Better bound: $\ln|S|$

Theorem

Greedy algorithm approximates the optimal set cover to within a factor of $H(\max\{|S| \mid S \in \mathcal{F}\})$ *where* $H(n) = \Sigma_{i=1}^{n} \frac{1}{i}$

Proof

- imagine a price to be paid by each team
- at each stage 1 euro has to be paid by newly invited team members, split evenly
- $t \leq$ total amount paid

X for each $S \in \mathcal{F}$ selected by the greedy algorithm the total amount paid by its members is at most $\ln|S|$

$\Rightarrow$ the total amount paid (hence t) is less than $k \cdot \ln|S|$ for the largest S selected

Proof of (X)

For an arbitrary set S at any stage of the algorithm holds:

- if m members are uncovered, the algorithm chooses a subset covering at least m elements

$\Rightarrow$ each will pay $\leq 1/m$

- members pay the most, if they are covered one by one

$\Rightarrow$ harmonic series

Travelling Salesman Problem

Example (TSP)

Given a complete, weighted, undirected graph $G = (V, E)$ with non-negative weights. Find a Hamiltonian cycle of minimal cost.

Theorem

TSP *is* **NP***-hard.*

Proof: Reduce from Hamilton cycle (HC) by giving a large weight to non-edges.

Roadmap

Just seen

- NP-hard optimization problems
- approximation to within a certain factor
- complexity of approximation for any factor?

Up next

- approximation algorithm for special case of TSP
- Inapproximability results

Triangle Equality Instance

In practice, TSP is applied on graphs that satisfy the triangle inequality:

$$\forall u, v, w \in V. c(u, v) \leq c(u, w) + c(w, v)$$

Approximation algorithm for such *geographical* graphs

1. find minimum spanning tree T_G for $G = (V, E)$
2. traverse along depth-first search of T_G, jump over visited nodes

- algorithm is polynomial
- 2-approximation
  - $c(T_G) \leq$ minimal tour
  - algorithm traversal costs $2 \cdot c(T_G)$ since jumping over costs at most the sum of traversed edges

Roadmap

Just seen

- special TSP instance with polynomial 2-approximation

Up next

- show it is **NP**-hard to approximate general TSP to within any factor $\rho \geq 1$
- introduce gap version of TSP

gap-TSP

Given a complete, weighted, undirected graph $G = (V, E)$ and some constant $h \geq 1$.

Definition (gap-TSP)

A solution to the gap problem, $\text{gap} - \text{TSP}[|V|, h|V|]$, is an algorithm that return

YES if there exists a Hamiltonian cycle of cost $< |V|$

NO if all Hamiltonian cycles have cost $> h|V|$

For all other cases, it may return either yes or no.

Observation: An efficient *h*-approximation for TSP decides $\text{gap} - \text{TSP}[C, hC]$ for any C.

gap-TSP is NP-hard

Theorem

For any $h \geq 1$, HC $\leq_p$ gap $-$ TSP$[|V|, h|V|]$

Proof: Like HC $\leq_P$ TSP, where non-edge weights are $h|V|$.

$\Rightarrow$ Approximating TSP to within any factor is **NP**-hard.

What have we learnt?

- some NP-hard decision problems have optimization problems that can be efficiently approximated
  - vertex cover within factor 2
  - set cover within a logarithmic factor
  - geographical travelling salesman problem within factor 2
- some other problems are even NP-hard to approximate, for instance, general TSP
- gap problems are a useful tool to establish inapproximablity

Further Reading

Two books on approximation algorithms

- *Dorit Hochbaum*, Approximation Algorithms for NP-Hard Problems, PWS Publishing.
- *Vijay Vazirani*, Approximation algorithms, Springer.

Lecture Notes

Slides are adapted from a CC course by *Muli Safra*:

`http://www.cs.tau.ac.il/~safra/Complexity/Complexity.htm`

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

June 3, 2019

Lecture 19

Hardness of Approximation

Recap: optimization

- many decision problems we have seen have optimization versions
- both minimization and maximization
- algorithms return best solution with respect to optimization parameter ρ

Recap: optimization

- many decision problems we have seen have optimization versions
- both minimization and maximization
- algorithms return best solution with respect to optimization parameter ρ

Examples

problem	min/max	parameter
3SAT	max	fraction of satisfiable clauses
Indset	max	size of independent set
VC	min	size of cover

Recap: approximation results

- vertex cover has a 2-approximation
  - possibly **NP**-hard to approximate to within $2-\epsilon$ for all $\epsilon > 0$
  - currently known: **NP**-hard to approximate to within $10\sqrt{5}-21$;
  - *I. Dinur, S. Safra*, The importance of being biased, STOC 2002.

Recap: approximation results

- vertex cover has a 2-approximation
  - possibly **NP**-hard to approximate to within $2 - \epsilon$ for all $\epsilon > 0$
  - currently known: **NP**-hard to approximate to within $10\sqrt{5} - 21$;
  - *I. Dinur, S. Safra*, The importance of being biased, STOC 2002.
- set cover has a $\ln n$ approximation
  - this is optimal; it is **NP**-hard to approximate to within $(1 - \epsilon) \ln n$
  - *U. Feige*, A threshold of $\ln n$ for approximating set cover, STOC 1996.

Recap: approximation results

- vertex cover has a 2-approximation
  - possibly **NP**-hard to approximate to within $2 - \epsilon$ for all $\epsilon > 0$
  - currently known: **NP**-hard to approximate to within $10\sqrt{5} - 21$;
  - *I. Dinur, S. Safra*, The importance of being biased, STOC 2002.
- set cover has a $\ln n$ approximation
  - this is optimal; it is **NP**-hard to approximate to within $(1 - \epsilon)\ln n$
  - *U. Feige*, A threshold of $\ln n$ for approximating set cover, STOC 1996.
- TSP also hard to approximate to within any $1 + \epsilon$

Polynomial time approximation schemes

A problem has a polynomial time approximation scheme if for all $\epsilon > 0$ it can be efficiently approximated to within a factor of $1 - \epsilon$ for maximization and $1 + \epsilon$ for minimization.

Polynomial time approximation schemes

A problem has a polynomial time approximation scheme if for all $\epsilon > 0$ it can be efficiently approximated to within a factor of $1 - \epsilon$ for maximization and $1 + \epsilon$ for minimization.

Examples

- knapsack
- bin packing
- subset sum
- a number of other scheduling problems

Polynomial time approximation schemes

A problem has a polynomial time approximation scheme if for all $\epsilon > 0$ it can be efficiently approximated to within a factor of $1 - \epsilon$ for maximization and $1 + \epsilon$ for minimization.

Examples

- knapsack
- bin packing
- subset sum
- a number of other scheduling problems

Which **NP**-complete problems do have PTAS? Which don't? How to prove results on previous slide?

Recap: gap – TSP$[|V|, h|V|]$

An algorithm to solve the gap problem needs to:

- if G has a shortest tour of length $< |V|$ then G is accepted by the gap algorithm
- if the shortest tour of G is $> h|V|$ then G is rejected
- otherwise: don't care

Recap: gap $-$ TSP$[|V|, h|V|]$

An algorithm to solve the gap problem needs to:

- if G has a shortest tour of length $< |V|$ then G is accepted by the gap algorithm
- if the shortest tour of G is $> h|V|$ then G is rejected
- otherwise: don't care

Theorem: For any $h \geq 1$ gap $-$ TSP$[|V|, h|V|]$ is **NP**-hard by reduction from Hamiltonian cycle

Recap: gap – TSP$[|V|, h|V|]$

An algorithm to solve the gap problem needs to:

- if G has a shortest tour of length $< |V|$ then G is accepted by the gap algorithm
- if the shortest tour of G is $> h|V|$ then G is rejected
- otherwise: don't care

Theorem: For any $h \geq 1$ gap – TSP$[|V|, h|V|]$ is **NP**-hard by reduction from Hamiltonian cycle

$\Rightarrow$ It is **NP**-hard to approximate TSP to within any factor $h \geq 1$.

Recap: gap – TSP$[|V|, h|V|]$

An algorithm to solve the gap problem needs to:

- if G has a shortest tour of length $< |V|$ then G is accepted by the gap algorithm
- if the shortest tour of G is $> h|V|$ then G is rejected
- otherwise: don't care

Theorem: For any $h \geq 1$ gap – TSP$[|V|, h|V|]$ is **NP**-hard by reduction from Hamiltonian cycle

$\Rightarrow$ It is **NP**-hard to approximate TSP to within any factor $h \geq 1$.

The reduction is called gap-producing.

Agenda

- gap $-$ 3SAT$[\rho, 1]$
- 7/8 approximation for max3SAT
- PCP theorem: hardness of approximation view
- gap-preserving reductions
- hardness of approximating Indset and VC

gap-3SAT$[\rho, 1]$

- $\text{gap} - 3\text{SAT}[\rho, 1]$ is the gap version of max3SAT which computes the largest fraction of satisfiable clauses
- a 3CNF with m clauses is accepted if it is satisfiable
- it is rejected if $< \rho \cdot m$ clauses are satisfiable
- until 1992 it was an open problem whether max3SAT could be approximated to within any factor $> 7/8$
- why 7/8?

A 7/8 approximation of max3SAT

Theorem

For all 3CNF with exactly three independent literals per clause, there exists an assignment that satisfies $\geq 7/8$ *of the clauses.*

A 7/8 approximation of max3SAT

Theorem

For all 3CNF with exactly three independent literals per clause, there exists an assignment that satisfies $\geq 7/8$ of the clauses.

Proof

- for a random assignment let Y_i be the random variable that is true if clause C_i is true under the assignment
- then $N = \Sigma_{i=1}^{m} Y_i$ is the number of satisfied clauses
- $E[Y_i] = 7/8$ for all i

$\Rightarrow$ $E[N] = 7/8 \cdot m$

- by the law of average (probabilistic method basic principle) there must exist an assignment that makes 7/8 of the clauses true

A 7/8 approximation of max3SAT

Theorem

For all 3CNF with exactly three independent literals per clause, there exists an assignment that satisfies $\geq 7/8$ of the clauses.

Proof

- for a random assignment let Y_i be the random variable that is true if clause C_i is true under the assignment
- then $N = \Sigma_{i=1}^{m} Y_i$ is the number of satisfied clauses
- $E[Y_i] = 7/8$ for all i

$\Rightarrow$ $E[N] = 7/8 \cdot m$

- by the law of average (probabilistic method basic principle) there must exist an assignment that makes 7/8 of the clauses true

Can we do any better than 7/8?

No!

Theorem

For every $\epsilon > 0$ gap $-$ 3SAT$[7/8 + \epsilon, 1]$ is **NP***-hard.*

- this is a PCP theorem by *J. Håstad*, Some optimal inapproximability results, STOC 1997.
- as a consequence, if there exists a $7/8 + \epsilon$ approximation of max3SAT then $\mathbf{P} = \mathbf{NP}$
- we will later prove a much weaker PCP theorem

Agenda

- $\text{gap}-\text{3SAT}[\rho, 1]$ ✓
- 7/8 approximation for max3SAT ✓
- PCP theorem: hardness of approximation view
- gap-preserving reductions
- hardness of approximating Indset and VC

THE PCP theorem

Håstads result is one in a series of inapproximability results based on the PCP theorem.

Theorem (PCP: hardness of approximation)

There exists a $\rho < 1$ *such that* $\mathrm{gap} - 3\mathrm{SAT}[\rho, 1]$ *is* **NP***-hard.*

- Safra: *One of the deepest and most complicated proofs in computer science with a matching impact.*
- original proof in two papers:
  - *Arora, Safra*, Probabilistic checking of proofs, FOCS 92
  - *Arora, Lund, Motwani, Sudan, Szegedy*, Proof verification and the hardness of approximations, FOCS 92.
- virtually all inapproximability results depend on the PCP theorem and the notion of gap preserving reductions by Papadimitriou and Yannakakis

Probabilistically checkable proofs

- the PCP theorem is equivalent to the statement $\mathbf{NP} = \mathbf{PCP}[\log n, 1]$
- PCP stands for probabilistically checkable proofs and is related to interactive proofs and $\mathbf{MIP} = \mathbf{NEXP}$
- equivalence of two views shown in next lecture
- $\mathbf{NP} = \mathbf{PCP}[poly(n), 1]$ shown after that

Agenda

- gap – 3SAT[ρ, 1] ✓
- 7/8 approximation for max3SAT ✓
- PCP theorem: hardness of approximation view ✓
- gap-preserving reductions
- hardness of approximating Indset and VC

Gap-producing and preserving reductions

PCP theorem states that for every $L \in \mathbf{NP}$ there exists a gap-producing reduction f to $\text{gap} - 3\text{SAT}[\rho, 1]$:

- $x \in L \implies f(x)$ is satisfiable
- $x \notin L \implies$ less than ρ of the $f(x)$'s clauses can be satisfied at the same time

Gap-producing and preserving reductions

PCP theorem states that for every $L \in \mathbf{NP}$ there exists a gap-producing reduction f to $\text{gap} - 3\text{SAT}[\rho, 1]$:

- $x \in L \implies f(x)$ is satisfiable
- $x \notin L \implies$ less than ρ of the $f(x)$'s clauses can be satisfied at the same time

Observation

- in order to show inapproximability of other problems, we want to preserve gaps by reductions

$\text{gap} - 3\text{SAT}[\rho, 1] \leq_{gap} \text{gap} - \text{IS}[\rho, 1]$

Consider the proof of 3SAT $\leq_p$ Indset (nodes are satisfying assignments for each clause, edges between incompatible ones).

The reduction f used there is actually gap-preserving, we write

$$\text{gap} - 3\text{SAT}[\rho, 1] \leq_{gap} \text{gap} - \text{IS}[\rho, 1]$$

- if 3CNF ψ with m clauses is satisfiable then graph $f(\psi)$ has an independent set of size m
- if less than ρ of ψ's clauses can be satisfied, the largest independent set has less than $\rho \cdot m$ nodes
- hence: if we can approximate Indest to within ρ, then we can approximate max3SAT to within ρ, then we can decide any $L \in$ **NP**

What about vertex cover?

The same reduction f from independent set can be used to show hardness of approximating vertex cover to within $(7-\rho)/6$ for the same ρ used in max3SAT and Indset.

What about vertex cover?

The same reduction f from independent set can be used to show hardness of approximating vertex cover to within $(7 - \rho)/6$ for the same ρ used in max3SAT and Indset.

- ψ satisfiable

$\Rightarrow$ $f(\psi)$ has i.s. of size m

$\Rightarrow$ $f(\psi)$ has a v.c. of size $6m$

What about vertex cover?

The same reduction f from independent set can be used to show hardness of approximating vertex cover to within $(7-\rho)/6$ for the same ρ used in max3SAT and Indset.

- ψ satisfiable

$\Rightarrow$ $f(\psi)$ has i.s. of size m

$\Rightarrow$ $f(\psi)$ has a v.c. of size $6m$

- only $\rho \cdot m$ of ψ's clauses satisfiable

$\Rightarrow$ $f(\psi)$ has largest i.s. smaller than ρm

$\Rightarrow$ $f(\psi)$ has smallest v.c. of size larger than $(7-\rho)m$

Independent set vs. vertex cover

- For both independent set and vertex cover, we know that there exist a $\rho < 1$ such that neither can be approximated to within ρ (resp. $1/\rho$)

Independent set vs. vertex cover

- For both independent set and vertex cover, we know that there exist a $\rho < 1$ such that neither can be approximated to within ρ (resp. $1/\rho$)
- optimal solutions are intimately related: if *vc* is the smallest vertex cover and *is* the largest independent set then $vc = is - n$

Independent set vs. vertex cover

- For both independent set and vertex cover, we know that there exist a $\rho < 1$ such that neither can be approximated to within ρ (resp. $1/\rho$)
- optimal solutions are intimately related: if vc is the smallest vertex cover and is the largest independent set then $vc = is - n$
- but: approximation is different; using the ρ app. for independent set, yields a $\frac{n-\rho \cdot is}{n-is}$ approximation for set cover

Independent set vs. vertex cover

- For both independent set and vertex cover, we know that there exist a $\rho < 1$ such that neither can be approximated to within ρ (resp. $1/\rho$)
- optimal solutions are intimately related: if *vc* is the smallest vertex cover and *is* the largest independent set then $vc = is - n$
- but: approximation is different; using the ρ app. for independent set, yields a $\frac{n-\rho \cdot is}{n-is}$ approximation for set cover
- for independent set we can show **NP**-hardness of approximation to within any factor $\rho < 1$ by gap amplification

Gap amplification

- given instance $G = (V, E)$
- construct $G' = (V \times V, E')$ where

$$E' = \{(u, v), (u', v') \mid (u, u') \in E \vee (v, v') \in E\}$$

- if $I \subseteq V$ is an i.s. of G then $I \times I$ is an i.s. of G'; hence $opt(G') \geq opt(G)^2$
- if I' is an optimal i.s. in G' with vertices $(u_1, v_1), \ldots, (u_j, v_j)$ then the u_i and the v_i are each i.s. in G with at most $opt(G)$ vertices; hence $opt(G') \leq opt(G)^2$
- hence i.s. is also hard to approximate within ρ^2
- this can be done any constant k times to obtain the result

What have we learnt?

- 7/8 approximation for max3SAT
- PCP theorem: hardness of approximating max3SAT
- gap-preserving reductions to obtain more inapproximability results
- **NP**-hardness of approximating Indset to within any $\rho < 1$
- **NP**-hardness of approximating VC to within some $\rho > 1$ (yet unknown)
- but: many **NP**-complete problems can still be approximated to within any factor $1 + \epsilon$

Up next

- PCP: hardness of approximation vs. prob. checkable proofs
- proof of a weaker PCP theorem

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

June 3, 2019

Lecture 20

Probabilistically checkable proofs

Goal and plan

Goal

- understand probabilistically checkable proofs,
- know some examples, and
- see the relation (in fact, equivalence) between PCP and hardness of approximation

Plan

- PCP for GNI
- definition: intuition and formalization
- PCP theorem and some obvious consequences
- tool: a more general 3SAT, constraint satisfaction CSP
- PCP theorem $\implies$ gapCSP$[\rho, 1]$ is **NP**-hard
- gapCSP$[\rho, 1]$ is **NP**-hard $\implies$ PCP theorem

PCP: an intuition

What does probabilistically checkable mean?

PCP: an intuition

What does probabilistically checkable mean?

- you want to verify correctness of a proof by only looking at a few bits of it

PCP: an intuition

What does probabilistically checkable mean?

- you want to verify correctness of a proof by only looking at a few bits of it

Which proofs?

PCP: an intuition

What does probabilistically checkable mean?

- you want to verify correctness of a proof by only looking at a few bits of it

Which proofs?

- typically membership in a language

PCP: an intuition

What does probabilistically checkable mean?

- you want to verify correctness of a proof by only looking at a few bits of it

Which proofs?

- typically membership in a language

Why should I care?

PCP: an intuition

What does probabilistically checkable mean?

- you want to verify correctness of a proof by only looking at a few bits of it

Which proofs?

- typically membership in a language

Why should I care?

- because it gives you a tool to prove hardness of approximation

How can it be done?

How can it be done?

Example

- Susan picks some $0 \leq n \leq 10$, Matt wants to know which n
- problem: his vision is blurred, he only sees up to ± 5

How can it be done?

Example

- Susan picks some $0 \leq n \leq 10$, Matt wants to know which n
- problem: his vision is blurred, he only sees up to ± 5

Solution

- Matt: Hey, Susan, why don't you show me $100 \cdot n$ instead?

Can you say this more formally?

- blurred vision $\sim$ we cannot see all bits of a proof

$\Rightarrow$ we can check only a few bits

- proofs can be spread out such that wrong proofs are wrong everywhere
- the definition of PCP will require existence of a proof only
- a correct proof must always be accepted (completeness 1)
- a wrong proof must be rejected with high probability (soundness ρ)

Does it work for real problems?

Does it work for real problems?

- yes, here is a PCP for graph non-isomorphism
- we use our familiar notion of verifier and prover
- albeit both face some limitations (later)

PCP for GNI

Input: graphs G_0, G_1 with n nodes

Verifier	Proof π

PCP for GNI

Input: graphs G_0, G_1 with n nodes

Verifier	Proof π
	• an array π indexed by all graphs with n nodes • $\pi[H]$ contains a if $H \cong G_a$ • otherwise 0 or 1

PCP for GNI

Input: graphs G_0, G_1 with n nodes

Verifier	Proof π
• picks $b \in \{0,1\}$ at random • picks random permutation $\sigma : [n] \to [n]$ • asks for $b' = \pi[\sigma(G_b)]$ • accepts iff $b' = b$	• an array π indexed by all graphs with n nodes • $\pi[H]$ contains a if $H \cong G_a$ • otherwise 0 or 1

Analysis

- $|\pi|$ is exponential in n
- verifier asks for only one bit
- verifier needs $\mathcal{O}(n)$ random bits
- verifier is a polynomial time TM
- if π is correct, the verifier always accepts
- if π is wrong (e.g. because $G_0 \cong G_1$, then verifier accepts with probability $1/2$

Agenda

- PCP for GNI ✓
- definition: intuition and formalization
- PCP theorem and some obvious consequences
- tool: a more general 3SAT, constraint satisfaction CSP
- PCP theorem $\implies$ gapCSP$[\rho, 1]$ is **NP**-hard
- gapCSP$[\rho, 1]$ is **NP**-hard $\implies$ PCP theorem

PCP system for $L \subseteq \{0,1\}^*$

Input: word $x \in \{0,1\}^n$

Verifier	Prover
1. pick $r(n)$ random bits 2. pick $q(n)$ positions/bits in π 3. based on x and random bits, compute $\Phi : \{0,1\}^{q(n)} \to \{0,1\}$ 4. after receiving proof bits $\pi_1, \ldots, \pi_{q(n)}$ output $\Phi(\pi_1, \ldots, \pi_{q(n)})$	• creates a proof π that $x \in L$ • $\lvert\pi\rvert \in 2^{r(n)} q(n)$ • on request, sends bits of π

- V is a polynomial-time TM
- if $x \in L$ then there exists a proof π s.t. V always accepts
- if $x \notin L$ then V accepts with probability $\leq 1/2$ for all proofs π

PCP$[r(n), q(n)]$

Definition

A language $L \in \{0,1\}^*$ is in PCP$[r(n), q(n)]$ iff there exists a PCP system with $c \cdot r(n)$ random bits and $d \cdot q(n)$ queries for constants $c, d > 0$.

PCP$[r(n), q(n)]$

Definition

A language $L \in \{0,1\}^*$ is in PCP$[r(n), q(n)]$ iff there exists a PCP system with $c \cdot r(n)$ random bits and $d \cdot q(n)$ queries for constants $c, d > 0$.

Theorem (THE PCP theorem)

PCP$[\log n, 1]$ = NP.

Observations

- GNI $\in$ **PCP**$[poly(n), 1]$
- the soundness parameter is arbitrary and can be amplified by repetition
- **PCP**$[0, 0]$

Observations

- GNI $\in$ **PCP**$[poly(n), 1]$
- the soundness parameter is arbitrary and can be amplified by repetition
- **PCP**$[0, 0] =$ **P**
- **PCP**$[0, \log(n)]$

Observations

- GNI $\in$ **PCP**$[poly(n), 1]$
- the soundness parameter is arbitrary and can be amplified by repetition
- **PCP**$[0, 0] =$ **P**
- **PCP**$[0, \log(n)] =$ **P**
- **PCP**$[0, poly(n)]$

Observations

- GNI $\in$ **PCP**$[poly(n), 1]$
- the soundness parameter is arbitrary and can be amplified by repetition
- **PCP**$[0, 0] =$ **P**
- **PCP**$[0, \log(n)] =$ **P**
- **PCP**$[0, poly(n)] =$ **NP**
- **PCP**$[r(n), q(n)] \subseteq$ **NTIME**$(2^{\mathcal{O}(r(n))}q(n))$

Observations

- GNI $\in$ **PCP**$[poly(n), 1]$
- the soundness parameter is arbitrary and can be amplified by repetition
- **PCP**$[0, 0] =$ **P**
- **PCP**$[0, \log(n)] =$ **P**
- **PCP**$[0, poly(n)] =$ **NP**
- **PCP**$[r(n), q(n)] \subseteq$ **NTIME**$(2^{\mathcal{O}(r(n))}q(n))$

$\Rightarrow$ **PCP**$[\log n, 1] \subseteq$ **NP**

- every problem in **NP** has a polynomial sized proof (certificate), of which we need to check only a constant number of bits
- for 3SAT (and hence for all!) as low as 3!

More remarks

- the Cook-Levin reduction does not suffice to prove the PCP theorem
  - because of soundness
  - even for $x \notin L$, almost all clauses are satisfiable
  - because they describe acceptable computations

More remarks

- the Cook-Levin reduction does not suffice to prove the PCP theorem
  - because of soundness
  - even for $x \notin L$, almost all clauses are satisfiable
  - because they describe acceptable computations
- PCP is inherently different from **IP**
  - proofs can be exponential in PCP
  - PCP: restrictions on queries and random bits
  - IP: restrictions on total message length

$\Rightarrow$ **PCP**$[poly(n), poly(n)] \supseteq$ **IP** $=$ **PSPACE** (in fact equal to **NEXP**)

Agenda

- PCP for GNI ✓
- definition: intuition and formalization ✓
- PCP theorem and some obvious consequences ✓
- tool: a more general 3SAT, constraint satisfaction CSP
- PCP theorem $\implies$ gapCSP[ρ, 1] is **NP**-hard
- gapCSP[ρ, 1] is **NP**-hard $\implies$ PCP theorem

Constraint satisfaction

3SAT	qCSP
• n Boolean variables	• n Boolean variables
• m clauses	• m general constraints
• each clause has 3 variables	• each constraint is over q variables

CSP remarks

- one can define the fraction of simultaneously satisfiable clauses just as for max3SAT
- each constraint represents a function $\{0,1\}^q \to \{0,1\}$
- we may assume that all variables are used: $n \leq qm$

$\Rightarrow$ a qCSP instance can be represented using $mq\log(n)2^q$ bits (polynomial in n, m)

gap-CSP

Definition

$\text{gap}-\text{qCSP}[\rho, 1]$ is **NP**-hard if for every $L \in$ **NP** there is a gap-producing reduction f such that

- $x \in L \implies f(x)$ is satisfiable
- $x \notin L \implies$ at most ρ constraints of $f(x)$ are satisfiable (at the same time)

Agenda

- PCP for GNI ✓
- definition: intuition and formalization ✓
- PCP theorem and some obvious consequences ✓
- tool: a more general 3SAT, constraint satisfaction CSP ✓
- PCP theorem $\implies$ gapCSP$[\rho, 1]$ is **NP**-hard
- gapCSP$[\rho, 1]$ is **NP**-hard $\implies$ PCP theorem

PCP $\Leftrightarrow$ Hardness of approximation

Theorem

The following two statements are equivalent.

- $\mathbf{NP} = \mathbf{PCP}[\log n, 1]$
- *there exist* $0 < \rho < 1$ *and* $q \in \mathbb{N}$ *such that* $\mathrm{gap} - \mathrm{qCSP}[\rho, 1]$ *is* **NP**-*hard.*

PCP $\Leftrightarrow$ Hardness of approximation

Theorem

The following two statements are equivalent.

- $\mathbf{NP} = \mathbf{PCP}[\log n, 1]$
- *there exist* $0 < \rho < 1$ *and* $q \in \mathbb{N}$ *such that* $\text{gap} - \text{qCSP}[\rho, 1]$ *is* **NP**-*hard.*

- this formalizes the equivalence of probabilistically checkable proofs and hardness of approximation
- this is why the PCP theorem was a breakthrough in inapproximability
- gap preservation from CSP to 3SAT is not hard but omitted

$\Rightarrow$

- show that there is a gap-producing reduction f from 3SAT to $\mathrm{gap}-\mathrm{qCSP}[1/2, 1]$

$\Rightarrow$

- show that there is a gap-producing reduction f from 3SAT to $\text{gap}-\text{qCSP}[1/2, 1]$
- by PCP, 3SAT has PCP system with poly. time verifier V, a constant q queries, using $c \log n$ random bits

$\Rightarrow$

- show that there is a gap-producing reduction f from 3SAT to $\mathrm{gap} - \mathrm{qCSP}[1/2, 1]$
- by PCP, 3SAT has PCP system with poly. time verifier V, a constant q queries, using $c \log n$ random bits
- define $f(x) = \{\psi_r : \{0,1\}^q \to \{0,1\} \mid r \in \{0,1\}^{c \log n}\}$ such that
- $\psi_r(b_1, \ldots, b_q) = 1$ if V accepts the bits from proof π given by r

$\Rightarrow$

- show that there is a gap-producing reduction f from 3SAT to $\text{gap} - \text{qCSP}[1/2, 1]$
- by PCP, 3SAT has PCP system with poly. time verifier V, a constant q queries, using $c \log n$ random bits
- define $f(x) = \{\psi_r : \{0,1\}^q \to \{0,1\} \mid r \in \{0,1\}^{c \log n}\}$ such that
- $\psi_r(b_1, \ldots, b_q) = 1$ if V accepts the bits from proof π given by r
- $f(x)$ is a qCSP of size $2^{c \log n} \in \mathcal{O}(n)$, representable and computable in poly time

$\Rightarrow$

- show that there is a gap-producing reduction f from 3SAT to $\text{gap} - \text{qCSP}[1/2, 1]$
- by PCP, 3SAT has PCP system with poly. time verifier V, a constant q queries, using $c \log n$ random bits
- define $f(x) = \{\psi_r : \{0,1\}^q \to \{0,1\} \mid r \in \{0,1\}^{c \log n}\}$ such that
- $\psi_r(b_1, \ldots, b_q) = 1$ if V accepts the bits from proof π given by r
- $f(x)$ is a qCSP of size $2^{c \log n} \in \mathcal{O}(n)$, representable and computable in poly time
- if $x \in$ 3SAT then there exists proof π s.t. $f(x)$ is satisfiable

$\Rightarrow$

- show that there is a gap-producing reduction f from 3SAT to $\mathrm{gap}-\mathrm{qCSP}[1/2, 1]$
- by PCP, 3SAT has PCP system with poly. time verifier V, a constant q queries, using $c \log n$ random bits
- define $f(x) = \{\psi_r : \{0,1\}^q \to \{0,1\} \mid r \in \{0,1\}^{c \log n}\}$ such that
- $\psi_r(b_1, \ldots, b_q) = 1$ if V accepts the bits from proof π given by r
- $f(x)$ is a qCSP of size $2^{c \log n} \in \mathcal{O}(n)$, representable and computable in poly time
- if $x \in$ 3SAT then there exists proof π s.t. $f(x)$ is satisfiable
- if $x \notin$ 3SAT then all proofs π satisfy at most $1/2$ of $f(x)$'s constraints

$\Rightarrow$

- show that there is a gap-producing reduction f from 3SAT to $\text{gap} - \text{qCSP}[1/2, 1]$
- by PCP, 3SAT has PCP system with poly. time verifier V, a constant q queries, using $c \log n$ random bits
- define $f(x) = \{\psi_r : \{0,1\}^q \to \{0,1\} \mid r \in \{0,1\}^{c \log n}\}$ such that
- $\psi_r(b_1, \ldots, b_q) = 1$ if V accepts the bits from proof π given by r
- $f(x)$ is a qCSP of size $2^{c \log n} \in \mathcal{O}(n)$, representable and computable in poly time
- if $x \in$ 3SAT then there exists proof π s.t. $f(x)$ is satisfiable
- if $x \notin$ 3SAT then all proofs π satisfy at most $1/2$ of $f(x)$'s constraints

$\Rightarrow$ f is gap-producing

$\Longleftarrow$

- show that for $L \in \mathbf{NP}$, there exists a PCP system

$\Leftarrow$

- show that for $L \in \mathbf{NP}$, there exists a PCP system
- by assumption there is a gap-producing reduction f from L to $\text{gap} - \text{qCSP}[\rho, 1]$ for some q and ρ

$\Leftarrow$

- show that for $L \in \mathbf{NP}$, there exists a PCP system
- by assumption there is a gap-producing reduction f from L to gap $-$ qCSP$[\rho, 1]$ for some q and ρ
  - for $x \in L$: $f(x)$ is satisfiable qCSP $\{\psi_i\}_{i=1}^{m}$
  - for $x \notin L$ at most ρm constraints satisfiable

$\Leftarrow$

- show that for $L \in \mathbf{NP}$, there exists a PCP system
- by assumption there is a gap-producing reduction f from L to $\text{gap} - \text{qCSP}[\rho, 1]$ for some q and ρ
  - for $x \in L$: $f(x)$ is satisfiable qCSP $\{\psi_i\}_{i=1}^m$
  - for $x \notin L$ at most ρm constraints satisfiable
- on input x the PCP verifier
  - computes $f(x)$
  - expects proof π to be assignment to $f(x)$'s n variables
  - picks $1 \leq j \leq m$ at random (needs $\log m$ bits!)
  - sets $\Phi = \psi_j$
  - asks for value of q variables of ψ_j

$\Leftarrow$

- show that for $L \in$ **NP**, there exists a PCP system
- by assumption there is a gap-producing reduction f from L to $\mathrm{gap}-\mathrm{qCSP}[\rho, 1]$ for some q and ρ
  - for $x \in L$: $f(x)$ is satisfiable qCSP $\{\psi_i\}_{i=1}^{m}$
  - for $x \notin L$ at most ρm constraints satisfiable
- on input x the PCP verifier
  - computes $f(x)$
  - expects proof π to be assignment to $f(x)$'s n variables
  - picks $1 \leq j \leq m$ at random (needs $\log m$ bits!)
  - sets $\Phi = \psi_j$
  - asks for value of q variables of ψ_j
- if $x \in L$ then V accepts with prob. 1
- if $x \notin L$ then V accepts with prob. ρ

$\Leftarrow$

- show that for $L \in$ **NP**, there exists a PCP system
- by assumption there is a gap-producing reduction f from L to $\mathrm{gap} - \mathrm{qCSP}[\rho, 1]$ for some q and ρ
  - for $x \in L$: $f(x)$ is satisfiable qCSP $\{\psi_i\}_{i=1}^m$
  - for $x \notin L$ at most ρm constraints satisfiable
- on input x the PCP verifier
  - computes $f(x)$
  - expects proof π to be assignment to $f(x)$'s n variables
  - picks $1 \leq j \leq m$ at random (needs $\log m$ bits!)
  - sets $\Phi = \psi_j$
  - asks for value of q variables of ψ_j
- if $x \in L$ then V accepts with prob. 1
- if $x \notin L$ then V accepts with prob. ρ
- ρ can be amplified to soundness error at most $1/2$ by constant number of repetitions

Recap: Two views of the PCP theorem

prob. checkable proofs		hardness of approximation
PCP verifier V	$\leftrightarrow$	CSP instance
proof π	$\leftrightarrow$	variable assignment
$\lvert\pi\rvert$	$\leftrightarrow$	number of variables in CSP
number of random bits	$\leftrightarrow$	$\log m$, where m is number of clauses
number of queries	$\leftrightarrow$	arity of constraints

What have we learnt?

- probabilistically checkable proofs are proofs with restrictions on the verifier's number of random bits and the number of proof bits queried
- yields a new, robust characterization of **NP**
- is equivalent to **NP**-hardness of $\text{gap} - \text{qCSP}[\rho, 1]$
- hence to **NP**-hardness of $\text{gap} - \text{3SAT}[\rho, 1]$
- hence to **NP**-hardness of approximation for many problems in **NP** (previous lecture)

Up next: Prove that $\mathbf{NP} \subseteq \mathbf{PCP}[\mathit{poly}(n), 1]$

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

June 4, 2019

Lecture 21

$\mathbf{NP} \subseteq \mathbf{PCP}[poly(n), 1]$

Recap: Two views of the PCP theorem

prob. checkable proofs		hardness of approximation
PCP verifier V	$\leftrightarrow$	CSP instance
proof π	$\leftrightarrow$	variable assignment
$\lvert\pi\rvert$	$\leftrightarrow$	number of variables in CSP
number of random bits	$\leftrightarrow$	$\log m$, where m is number of clauses
number of queries	$\leftrightarrow$	arity of constraints

Goal and plan

Goal

- proof a weaker PCP theorem
- learn interesing encoding/decoding schemes useful in such proofs

Plan

- proof
    - an **NP**-complete language: Quadeq
    - Walsh-Hadamard encodings
    - a **PCP**$[poly, 1]$ system for Quadeq
- summary: PCP and hardness of approximation

Weak PCP

Theorem

$\mathbf{NP} \subseteq \mathbf{PCP}[poly, 1]$

Proof: It suffices to come up with a PCP system for one **NP**-complete language, where the verifier

- uses polynomially many random bits (exponentially long proofs)
- makes a constant number of queries to that proof

Plan:

- an **NP**-complete language: Quadeq
- Walsh-Hadamard encodings
- a $\mathbf{PCP}[poly, 1]$ system for Quadeq

Disclaimer

All arithmetic today will be modulo 2, that is, over the field $\{0, 1\}$!

- $1 + 1 = 0$
- $x^2 = x$
- $x + y = x - y$

Quadeq

- satisfiable quadratic equations over $\{0, 1\}$
- n variables/m equations
- no purely linear terms
- **NP**-complete (exercise!)

Example (Running example)

$$\begin{aligned} xy + xz &= 1 \\ y^2 + yz + z^2 &= 1 \\ x^2 + yx + z^2 &= 0 \end{aligned}$$

Quadeq

- satisfiable quadratic equations over $\{0, 1\}$
- n variables/m equations
- no purely linear terms
- **NP**-complete (exercise!)

Example (Running example)

$$
\begin{aligned}
xy + xz &= 1 \\
y^2 + yz + z^2 &= 1 \\
x^2 + yx + z^2 &= 0
\end{aligned}
$$

Solution: $x = 1,\ y = 0,\ z = 1$
as a vector: $\mathbf{s} = (1\ 0\ 1)$

Be smart, use vector notation

$$\begin{aligned} xy + xz &= 1 \\ y^2 + yz + z^2 &= 1 \\ x^2 + yx + z^2 &= 0 \end{aligned}$$

$$\mathbf{s} = (1\ 0\ 1)$$

Be smart, use vector notation

$$\begin{aligned} xy + xz &= 1 \\ y^2 + yz + z^2 &= 1 \\ x^2 + yx + z^2 &= 0 \end{aligned}$$

$$\mathbf{s} = (1\ 0\ 1)$$

vector notation: for a given $m \times n^2$ matrix A and m vector $\mathbf{b}$ find solution $\mathbf{u} = (x\ y\ z)$ such that

$$A(\mathbf{u} \otimes \mathbf{u}) = \mathbf{b}$$

$\mathbf{u} \otimes \mathbf{u}$	x^2	xy	xz	yx	y^2	yz	zx	zy	z^2	
$\mathbf{s} \otimes \mathbf{s}$	1	0	1	0	0	0	1	0	1	$\mathbf{b}$
A	0	1	1	0	0	0	0	0	0	1
	0	0	0	0	1	1	0	0	1	1
	1	0	0	1	0	0	0	0	1	0

Overview

- Quadeq is the language of satisfiable systems of quadratic equations over $\{0, 1\}$
- natural PCP system expects a solution $\mathbf{u}$ and checks whether it is valid
- but this yields superconstant number of queries!
- how can we encode a solution such that a constant number of queries suffices?

Overview

- Quadeq is the language of satisfiable systems of quadratic equations over $\{0, 1\}$
- natural PCP system expects a solution **u** and checks whether it is valid
- but this yields superconstant number of queries!
- how can we encode a solution such that a constant number of queries suffices?
- use longer proofs!

- an **NP**-complete language: Quadeq ✓
- Walsh-Hadamard encodings
- a **PCP**$[poly, 1]$ system for Quadeq

PCP for Quadeq

Input: $m \times n^2$ matrix A, m vector $\mathbf{b}$

Verifier	Proof π
1. check that f, g are linear functions 2. check that $g = WH(\mathbf{u} \otimes \mathbf{u})$ where $f = WH(\mathbf{u})$ 3. check that g encodes a satisfying assignment	• $\pi \in \{0,1\}^{2^n + 2^{n^2}}$ • π is a pair of linear functions $\langle f, g \rangle$, i.e. strings from $\{0,1\}^{2^n}$ and $\{0,1\}^{2^{n^2}}$, resp. • if $\mathbf{u}$ satisfies $A(\mathbf{u} \otimes \mathbf{u}) = \mathbf{b}$ then $f = WH(\mathbf{u})$ and $g = WH(\mathbf{u} \otimes \mathbf{u})$ are Walsh-Hadamard encodings

Walsh-Hadamard encoding

Definition (WH)

Let $\mathbf{u} \in \{0,1\}^n$ be a vector. The Walsh-Hadamard encoding of $\mathbf{u}$ written $WH(\mathbf{u})$ is the truth table of the linear function $f : \{0,1\}^n \to \{0,1\}$ with $f(\mathbf{x}) = \mathbf{u} \odot \mathbf{x}$ where $(u_1 \dots u_n) \odot (x_1 \dots x_n) = \Sigma_{i=1}^n u_i x_i$.

Walsh-Hadamard encoding

Definition (WH)

Let $\mathbf{u} \in \{0,1\}^n$ be a vector. The Walsh-Hadamard encoding of $\mathbf{u}$ written $WH(\mathbf{u})$ is the truth table of the linear function $f : \{0,1\}^n \to \{0,1\}$ with $f(\mathbf{x}) = \mathbf{u} \odot \mathbf{x}$ where $(u_1 \dots u_n) \odot (x_1 \dots x_n) = \Sigma_{i=1}^n u_i x_i$.

Example

The solution to our running example is $\mathbf{s} = (1\ 0\ 1)$. We have

$$WH(\mathbf{s}) = (0\ 1\ 0\ 1\ 1\ 0\ 1\ 0)$$

Note: $|WH(\mathbf{u})| = 2^{|\mathbf{u}|}$

Properties (without proof)

Random subsum principle

- if $\mathbf{u} \neq \mathbf{v}$ then for 1/2 of the choices of $\mathbf{x}$ we have $\mathbf{u} \odot \mathbf{x} \neq \mathbf{v} \odot \mathbf{x}$
- if $\mathbf{u} \neq \mathbf{v}$ then $WH(\mathbf{u})$ and $WH(\mathbf{v})$ differ on at least half their bits

Properties (without proof)

Random subsum principle

- if $\mathbf{u} \neq \mathbf{v}$ then for 1/2 of the choices of $\mathbf{x}$ we have $\mathbf{u} \odot \mathbf{x} \neq \mathbf{v} \odot \mathbf{x}$
- if $\mathbf{u} \neq \mathbf{v}$ then $WH(\mathbf{u})$ and $WH(\mathbf{v})$ differ on at least half their bits

Local linearity testing

- we say that $f, g : \{0,1\}^n \rightarrow \{0,1\}$ are ρ-close if

$$Pr_{\mathbf{x} \in_R \{0,1\}^n}[f(\mathbf{x}) = g(\mathbf{x})] \geq \rho$$

- if there exists a $\rho > 1/2$ s.t.

$$Pr_{\mathbf{x},\mathbf{y} \in_R \{0,1\}^n}[f(\mathbf{x}+\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})] \geq \rho$$

then f is ρ-close to a linear function

PCP for Quadeq

Input: $m \times n^2$ matrix A, m vector $\mathbf{b}$

Verifier	Proof π

- $\pi \in \{0,1\}^{2^n + 2^{n^2}}$
- π is a pair of linear functions $\langle f, g\rangle$, i.e. strings from $\{0,1\}^{2^n}$ and $\{0,1\}^{2^{n^2}}$, resp.
- if $\mathbf{u}$ satisfies $A(\mathbf{u} \otimes \mathbf{u}) = \mathbf{b}$ then $f = WH(\mathbf{u})$ and $g = WH(\mathbf{u} \otimes \mathbf{u})$ are Walsh-Hadamard encodings

1. check that f, g are linear functions
2. check that $g = WH(\mathbf{u} \otimes \mathbf{u})$ where $f = WH(\mathbf{u})$
3. check that g encodes a satisfying assignment

Local linearity testing

- we test the linearity condition ($f(x+y) = f(x) + f(y)$) independently $1/\delta > 2$ times, and accept if all tests pass
- we accept a linear function with probability 1
- if f is not $1-\delta$-close to a linear function
  - all tests are passed with probability at most $(1-\delta)^{(1/\delta)}$

  $\Rightarrow$ such a function is rejected with probability at least $1 - 1/e > 1/2$
- for instance, we could make a 0.999 linearity test using 1000 trials

Local decoding

- it might happen, that we accept non-linear functions that are very close to linear functions
- in this case we treat them as if they were linear
- if we want to query $f(\mathbf{x})$
  1. we choose $\mathbf{x}' \in \{0, 1\}^n$ at random
  2. set $\mathbf{x}'' = \mathbf{x} + \mathbf{x}'$
  3. let $\mathbf{y}' = f(\mathbf{x}')$ and $\mathbf{y}'' = f(\mathbf{x}'')$
  4. output $\mathbf{y}' + \mathbf{y}''$
- this makes two queries instead of one
- and recovers the value of the closest linear function with high probability

PCP for Quadeq

Input: $m \times n^2$ matrix A, m vector $\mathbf{b}$

Verifier	Proof π
1. check that f, g are linear functions ✓ 2. check that $g = WH(\mathbf{u} \otimes \mathbf{u})$ where $f = WH(\mathbf{u})$ 3. check that g encodes a satisfying assignment	• $\pi \in \{0,1\}^{2^n + 2^{n^2}}$ • π is a pair of linear functions $\langle f, g \rangle$, i.e. strings from $\{0,1\}^{2^n}$ and $\{0,1\}^{2^{n^2}}$, resp. • if $\mathbf{u}$ satisfies $A(\mathbf{u} \otimes \mathbf{u}) = \mathbf{b}$ then $f = WH(\mathbf{u})$ and $g = WH(\mathbf{u} \otimes \mathbf{u})$ are Walsh-Hadamard encodings

Check WH encodings

Test 10 times for random $\mathbf{r}, \mathbf{r}' \in \{0,1\}^n$

$$f(\mathbf{r})f(\mathbf{r}') = g(\mathbf{r} \otimes \mathbf{r}')$$

Check WH encodings

Test 10 times for random $\mathbf{r}, \mathbf{r}' \in \{0,1\}^n$

$$f(\mathbf{r})f(\mathbf{r}') = g(\mathbf{r} \otimes \mathbf{r}')$$

If the proof is correct we always accept:

$$\begin{aligned} f(\mathbf{r})f(\mathbf{r}') &= \left(\sum_{i\in[n]} u_i r_i\right)\left(\sum_{j\in[n]} u_j r'_j\right) \\ &= \sum_{i,j\in[n]} u_i u_j r_i r'_j \\ &= ((\mathbf{u} \otimes \mathbf{u}) \odot (\mathbf{r} \otimes \mathbf{r}')) \\ &= g(\mathbf{r} \otimes \mathbf{r}') \end{aligned}$$

Check WH encodings

Test 10 times for random $\mathbf{r}, \mathbf{r}' \in \{0,1\}^n$

$$f(\mathbf{r})f(\mathbf{r}') = g(\mathbf{r} \otimes \mathbf{r}')$$

If the proof is correct we always accept:

$$\begin{aligned} f(\mathbf{r})f(\mathbf{r}') &= \left(\Sigma_{i\in[n]} u_i r_i\right)\left(\Sigma_{j\in[n]} u_j r'_j\right) \\ &= \Sigma_{i,j\in[n]} u_i u_j r_i r'_j \\ &= ((\mathbf{u} \otimes \mathbf{u}) \odot (\mathbf{r} \otimes \mathbf{r}')) \\ &= g(\mathbf{r} \otimes \mathbf{r}') \end{aligned}$$

If the proof is wrong we reject with probability at least 1/4 by applying the random subsum principle twice, because in esence we compute $\mathbf{r}U\mathbf{r}'$ and $\mathbf{r}W\mathbf{r}'$ for different matrices U and W.

PCP for Quadeq

Input: $m \times n^2$ matrix A, m vector $\mathbf{b}$

Verifier	Proof π

- $\pi \in \{0,1\}^{2^n + 2^{n^2}}$
- π is a pair of linear functions $\langle f, g \rangle$, i.e. strings from $\{0,1\}^{2^n}$ and $\{0,1\}^{2^{n^2}}$, resp.
- if $\mathbf{u}$ satisfies $A(\mathbf{u} \otimes \mathbf{u}) = \mathbf{b}$ then $f = WH(\mathbf{u})$ and $g = WH(\mathbf{u} \otimes \mathbf{u})$ are Walsh-Hadamard encodings

1. check that f, g are linear functions ✓
2. check that $g = WH(\mathbf{u} \otimes \mathbf{u})$ where $f = WH(\mathbf{u})$ ✓
3. check that g encodes a satisfying assignment

Is the assignment satisfying?

- for each of m equations we can check $g(\mathbf{z})$ at some place $\mathbf{z}$ corresponding to the coefficients in matrix A
- but this is not constant queries!

Is the assignment satisfying?

- for each of m equations we can check $g(\mathbf{z})$ at some place $\mathbf{z}$ corresponding to the coefficients in matrix A
- but this is not constant queries!
- instead multiply each equation by a random bit and take the sum of all equations
- if g encodes a solution, we will always have a solution to the sum
- otherwise, we have a solution with probability 1/2 only

Is the system in PCP[$poly(n), 1$]?

1. $\pi \in \{0,1\}^{2^n + 2^{n^2}}$
2. check that f, g are linear functions
   - $2(1-\delta) \cdot n$ random bits, $2(1-\delta)$ queries
3. check that $g = WH(\mathbf{u} \otimes \mathbf{u})$ where $f = WH(\mathbf{u})$
   - $20n$ random bits, 20 queries
4. check that g encodes a satisfying assignment
   - m random bits (one per equation), 1 query

Is the system in PCP$[poly(n), 1]$?

1. $\pi \in \{0, 1\}^{2^n + 2^{n^2}}$
2. check that f, g are linear functions
   - $2(1 - \delta) \cdot n$ random bits, $2(1 - \delta)$ queries
3. check that $g = WH(\mathbf{u} \otimes \mathbf{u})$ where $f = WH(\mathbf{u})$
   - $20n$ random bits, 20 queries
4. check that g encodes a satisfying assignment
   - m random bits (one per equation), 1 query

Yes!

Conclusion

PCP and hardness of approximation

- computing approximate solutions to **NP**-hard problems is important
- the classical Cook-Levin reduction does not rule out efficient approximations
- many nontrivial approximation algorithms exist (2-app for metric TSP, knapsack, 2-app for vertex cover)
- PCP theorem shows hardness of approximating max3SAT to within any constant factor if **P** $\neq$ **NP**
- we showed hardness of approximation for Indset as well
- this is equivalent to having a probabilistically checkable proof system with logarithmic randomness and constant queries
- PCP proofs involve intricate encoding schemes like Walsh-Hadamard

Further Reading *Luca Trevisan*, Inapproximability of Combinatorial Optimization Problems, available from
`http://www.cs.berkeley.edu/~luca/pubs/inapprox.pdf`

Next and final topic: Parallelism

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

June 4, 2019

Lecture 22

Models of Parallel Computation

Goal and plan

Goal

- introduce two models of parallel computation
- understand why they are equivalent

Plan

- PRAM: parallel random access machine
- circuits
- some complexity class definitions

Random access machine

RAM: more realistic model of sequential computation, which can be simulated by standard TMs with polynomial overhead.

- computation unit with user-defined program
- read-only input tape, write-only output tape, unbounded number of local memory cells
- memory cells can hold unbounded integers
- instructions include
  - moving data between memory cells
  - comparisons and branches
  - simple arithmetic operations
- all operations take unit time

Parallel random access machine

PRAM: parallel extension of RAM

- unbounded collection of RAM processors without tapes: $P_0, P_1, P_2, \ldots$
- unbounded collection of shared memory cells: $M[0], M[1], M[2], \ldots$
- each P_i has its own local memory (registers)
- input: n items stored in $M[0], \ldots, M[n-1]$
- output stored on some designated part of memory
- instructions execute in 3-phase cycles
  - read from shared memory
  - local computation
  - write to shared memory
- processors execute cycles synchronously
- P_0 starts and halts execution

Read/write conflicts

It may happen that several processors want to read from or write to the same memory cell in one cycle.

Read/write conflicts

It may happen that several processors want to read from or write to the same memory cell in one cycle.

Three policies:

- **EREW** : exclusive read/exclusive write
- **CREW** : concurrent read/exclusive write allows for simultaneous reads
- **CRCW** : simultaneous read and write allowed

Practical concerns

- idealized: PRAMs are an abstract, idealized formalism
  - unbounded integers
  - communication between any two processors in constant time due to shared memory (in reality: interconnection networks)
  - too many processors
- CRCW and CREW hard to build technically but easier to design algorithms
- still useful as benchmark
  - if there is no good PRAM algorithm, probably the problem is hard to parallelize

Time and space complexity

- time complexity: number of steps of P_0
- space complexity: number of shared memory cells accessed
- one can show that the weakest PRAM (EREW) can simulate the strongest with logarithmic overhead; cf. search-example
- efficient parallel computation
  - polynomially many processors
  - polylogarithmic time, where $polylog(n) = \bigcup_{k \geq 1} \log^k n$
- problems with efficient parallel algorithms are said to be in **NC**
- **NC** is robust wrt different PRAM models (and circuits)

Example: Search

Example

Given n items on the shared memory tape and $p+1 < n$ processors. For some $x \in \mathbb{N}$ P_0 wants to know, whether there exists an $0 \leq i < n$ such that $M[i] = x$.

Example: Search

Example

Given n items on the shared memory tape and $p + 1 < n$ processors. For some $x \in \mathbb{N}$ P_0 wants to know, whether there exists an $0 \leq i < n$ such that $M[i] = x$.

Solution (high level):

1. P_0 publishes x
2. for $1 \leq i \leq p$ each P_i searches through $M[\lceil \frac{n}{p} \rceil (i-1)], \ldots, M[\lceil \frac{n}{p} \rceil i - 1]$
3. each P_i announces its search result

Analysis

Step 2 need n/p parallel time independently of PRAM model.

Analysis

Step 2 need n/p parallel time independently of PRAM model.

Step 1

- needs $\mathcal{O}(1)$ time in CRCW and CREW since P_0 can simply write x on the shared tape which everybody can read simultaneously
- needs $\log p$ steps in EREW by binary broadcast tree

Analysis

Step 2 need n/p parallel time independently of PRAM model.

Step 1

- needs $\mathcal{O}(1)$ time in CRCW and CREW since P_0 can simply write x on the shared tape which everybody can read simultaneously
- needs $\log p$ steps in EREW by binary broadcast tree

Step 3

- needs $\mathcal{O}(1)$ time in CRCW only, where all successful processors indicate success in the same memory cell
- otherwise, we need $\log p$ time to perform a parallel reduction

Other problems in NC

Many practical problems are known to be in **NC**, for details, take some class on parallel algorithms.

- sorting
- matrix multiplication
- expression evaluation
- connected components of graphs
- string matching

Signpost

Just seen:

- RAMs and PRAMs
- CRCW, CREW, EREW
- simulations between models have at most logarithmic overhead
- efficient parallel $\sim$ polylogarithmic (stable under different PRAM models)

Next:

- Boolean circuits as parallel model of computation
- equivalence with respect to efficient parallel algorithms of PRAM and circuits

Boolean Circuits

Definition

A Boolean circuit, C, is a directed acyclic graph with labeled nodes.

- the input nodes are labeled with a variable x_i or with a constant 0 or 1
- the gate nodes have fan-in $k > 0$ are labeled with one of the Boolean functions
  - $\wedge$ (fan-in k)
  - $\vee$ (fan-in k)
  - $\neg$ (fan-in 1)
- the output nodes are labeled *output* and have fan-out 0

Boolean Circuits

Definition

A Boolean circuit, C, is a directed acyclic graph with labeled nodes.

- the input nodes are labeled with a variable x_i or with a constant 0 or 1
- the gate nodes have fan-in $k > 0$ are labeled with one of the Boolean functions
  - $\wedge$ (fan-in k)
  - $\vee$ (fan-in k)
  - $\neg$ (fan-in 1)
- the output nodes are labeled *output* and have fan-out 0

Given an assignment $\sigma : \{0, 1\}^m \rightarrow \{0, 1\}$ to the m variables, $C(\sigma)$ denotes the value of the o output nodes. We denote by $size(C)$ the number of gates and by $depth(C)$ the maximum distance from an input to an output.

Boolean Circuits

Definition

A Boolean circuit, C, is a directed acyclic graph with labeled nodes.

- the input nodes are labeled with a variable x_i or with a constant 0 or 1
- the gate nodes have fan-in $k > 0$ are labeled with one of the Boolean functions
  - $\wedge$ (fan-in k)
  - $\vee$ (fan-in k)
  - $\neg$ (fan-in 1)
- the output nodes are labeled *output* and have fan-out 0

Given an assignment $\sigma : \{0, 1\}^m \to \{0, 1\}$ to the m variables, $C(\sigma)$ denotes the value of the o output nodes. We denote by $size(C)$ the number of gates and by $depth(C)$ the maximum distance from an input to an output.
We distinguish circuits with and without a-priori bounds on fan-in.

Boolean Circuits

Definition

A Boolean circuit, C, is a directed acyclic graph with labeled nodes.

- the input nodes are labeled with a variable x_i or with a constant 0 or 1
- the gate nodes have fan-in $k > 0$ are labeled with one of the Boolean functions
  - $\wedge$ (fan-in k)
  - $\vee$ (fan-in k)
  - $\neg$ (fan-in 1)
- the output nodes are labeled *output* and have fan-out 0

Given an assignment $\sigma : \{0,1\}^m \to \{0,1\}$ to the m variables, $C(\sigma)$ denotes the value of the o output nodes. We denote by $size(C)$ the number of gates and by $depth(C)$ the maximum distance from an input to an output.

We distinguish circuits with and without a-priori bounds on fan-in.

Wlog we assume that all negations appear in the input layer only.

Example: addition

Assume we want to add two n-bit integers, that is, we want circuits to compute $+ : \{0,1\}^{2n} \to \{0,1\}^{n+1}$

Example: addition

Assume we want to add two n-bit integers, that is, we want circuits to compute $+ : \{0,1\}^{2n} \to \{0,1\}^{n+1}$

Ripple carry adder

- n sequential full adder
- depth: $\mathcal{O}(n)$
- size: $\mathcal{O}(n)$

Example: addition

Assume we want to add two n-bit integers, that is, we want circuits to compute $+ : \{0,1\}^{2n} \to \{0,1\}^{n+1}$

Ripple carry adder

- n sequential full adder
- depth: $\mathcal{O}(n)$
- size: $\mathcal{O}(n)$

Conditional sum adder

- depth: $\mathcal{O}(\log n)$
- size: $\mathcal{O}(n \log n)$

Example: addition

Assume we want to add two n-bit integers, that is, we want circuits to compute $+ : \{0,1\}^{2n} \to \{0,1\}^{n+1}$

Ripple carry adder

- n sequential full adder
- depth: $\mathcal{O}(n)$
- size: $\mathcal{O}(n)$

Conditional sum adder

- depth: $\mathcal{O}(\log n)$
- size: $\mathcal{O}(n \log n)$

Carry lookahead adder

- depth: $\mathcal{O}(\log n)$
- size: $\mathcal{O}(n)$

Deciding languages with circuits

Definition

A language $L \subseteq \{0,1\}^*$ is said to be decided by a family of circuits $\{C_n\}$, where C_i takes i input variables, iff for all i holds: $C_i(x) = 1$ iff $x \in L$.

Deciding languages with circuits

Definition

A language $L \subseteq \{0,1\}^*$ is said to be decided by a family of circuits $\{C_n\}$, where C_i takes i input variables, iff for all i holds: $C_i(x) = 1$ iff $x \in L$.

Definition

Let $d, s : \mathbb{N} \to \mathbb{N}$ be functions. We say that a family $\{C_n\}$ has depth d and size s if for all n

- $depth(C_n) \leq d(n)$
- $size(C_n) \leq s(n)$

Examples

Example (Parity)

Parity $= \{x \in \{0,1\}^* \mid x$ has an odd number of 1s$\}$

Examples

Example (Parity)

Parity $= \{x \in \{0,1\}^* \mid x$ has an odd number of 1s$\}$

- circuits are binary trees of *xor* gates
- each xor-gate has depth 3

$\Rightarrow$ logarithmic depth

Examples

Example (Parity)

Parity $= \{x \in \{0,1\}^* \mid x$ has an odd number of 1s$\}$

- circuits are binary trees of *xor* gates
- each xor-gate has depth 3

$\Rightarrow$ logarithmic depth

Example (UHalt)

UHalt $= \{1^n \mid$
n's binary expansion encodes a pair $\langle M, x\rangle$ such that M halts on $x\}$

Examples

Example (Parity)

Parity $= \{x \in \{0,1\}^* \mid x$ has an odd number of 1s$\}$

- circuits are binary trees of *xor* gates
- each xor-gate has depth 3

$\Rightarrow$ logarithmic depth

Example (UHalt)

UHalt $= \{1^n \mid$
n's binary expansion encodes a pair $\langle M, x\rangle$ such that M halts on $x\}$

- circuit family of linear size decides UHalt even though it is undecidable
- for each n with $1^n \in$ UHalt is a tree of and-gates
- otherwise, constant 0 circuit

On Uniformity

Problem on previous slide: the description of the circuit family is not computable.

On Uniformity

Problem on previous slide: the description of the circuit family is not computable.

Solution: uniformity

On Uniformity

Problem on previous slide: the description of the circuit family is not computable.

Solution: uniformity

Definition (logspace uniform)

A family of polynomially-sized circuits, $\{C_n\}$ is logspace-uniform if there exists a logspace TM M such that for every n, $M(1^n) = desc(C_n)$, where $desc(C_n)$ is the description of C_n.

On Uniformity

Problem on previous slide: the description of the circuit family is not computable.

Solution: uniformity

Definition (logspace uniform)

A family of polynomially-sized circuits, $\{C_n\}$ is logspace-uniform if there exists a logspace TM M such that for every n, $M(1^n) = desc(C_n)$, where $desc(C_n)$ is the description of C_n.

Remarks

- a description could be a list of gates along with type and predecessors
- the circuit family for Parity is logspace-uniform

Signpost

Just seen:

- circuit definition
- families of circuits decide languages
- there exist families of polynomial size deciding undecidable languages

$\Rightarrow$ require logspace-uniformity

Next:

- circuits vs PRAMs

Circuits vs PRAMs

For efficient parallel computations only:

parallel time on PRAM	$\sim$	circuit depth
number of processors	$\sim$	circuit size

circuits $\rightarrow$ PRAM

- suppose L decided by family $\{C_n\}$ of polynomial size N and depth $\mathcal{O}(\log^d n)$
- a PRAM with N processors decides L:
- compute a description of C_n
- each circuit node $\rightarrow$ one processor
- each processor computes its output and sends it to all other processors that need it (might require logarithmic overhead for non-CR models)
- parallel time $\sim$ circuit depth
- circuit size $\sim$ number of processors

Circuits vs PRAMs

For efficient parallel computations only:

parallel time on PRAM	$\sim$	circuit depth
number of processors	$\sim$	circuit size

PRAM $\rightarrow$ circuits

- circuit with $N \cdot D$ nodes in D layers
- the i-th node in the t-th layer performs computation of processor i at time t

NC and AC

Obviously, variations of PRAMs and circuits are robust wrt. polynomial size/number of processors and polylogarithmic depth/parallel run time motivating the following definition.

Definition (NC and AC)

Let $k \geq 0$. $L \in \mathbf{AC^k}$ iff L is decided by a logspace-uniform family of circuits with polynomial size and depth $\mathcal{O}(\log^k n)$. If the family of circuits is of bounded fan-in, then $L \in \mathbf{NC^k}$.

- $\mathbf{NC} = \bigcup_{k \geq 0} \mathbf{NC^k}$
- $\mathbf{AC} = \bigcup_{k \geq 0} \mathbf{AC^k}$

NC and AC

Obviously, variations of PRAMs and circuits are robust wrt. polynomial size/number of processors and polylogarithmic depth/parallel run time motivating the following definition.

Definition (NC and AC)

Let $k \geq 0$. $L \in \mathbf{AC^k}$ iff L is decided by a logspace-uniform family of circuits with polynomial size and depth $\mathcal{O}(\log^k n)$. If the family of circuits is of bounded fan-in, then $L \in \mathbf{NC^k}$.

- $\mathbf{NC} = \bigcup_{k \geq 0} \mathbf{NC^k}$
- $\mathbf{AC} = \bigcup_{k \geq 0} \mathbf{AC^k}$

- **NC** is the class of problems with efficient parallel solutions
- **AC** circuits cannot be build easily in hardware
- it is an open problem whether $\mathbf{P} = \mathbf{NC}$, that is, whether all problems in **P** are efficiently parallelizable (conjecture: no)
- Parity $\in \mathbf{NC^1}$ (but not in $\mathbf{AC^0}$)

Summary

- three variations of a PRAM
- uniform and non-uniform circuit families can decide languages
- efficiently parallelizable: **NC**
- circuits and PRAM are equivalent wrt **NC** problems

Up next: small depth circuits (**AC** and **NC**)

- their relation to well-known (space) complexity classes
- some lower bounds

Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

June 4, 2019

Lecture 23

NC and AC scrutinized

Recap

Efficient parallel computation

- computable by some PRAM with
- polynomially many processors in
- polylogarithmic time
- robust wrt to underlying PRAM model

Recap

Efficient parallel computation

- computable by some PRAM with
- polynomially many processors in
- polylogarithmic time
- robust wrt to underlying PRAM model

corresponds to

small depth circuits

- of polynomial size
- polylogarithmic depth
- logspace uniform

Recap – NC and AC

If $L \subseteq \{0,1\}^*$ is decided by a logspace-uniform family $\{C_n\}$ of polynomially sized circuits with bounded fan-in

- and depth $\log^k n$ then $L \in \mathbf{NC^k}$ for $k \geq 0$
- $\mathbf{NC} = \bigcup_{k \geq 0} \mathbf{NC^k}$

Recap – NC and AC

If $L \subseteq \{0,1\}^*$ is decided by a logspace-uniform family $\{C_n\}$ of polynomially sized circuits with bounded fan-in

- and depth $\log^k n$ then $L \in \mathbf{NC^k}$ for $k \geq 0$
- $\mathbf{NC} = \bigcup_{k \geq 0} \mathbf{NC^k}$

If the fan-in is unbounded we obtain the corresponding **AC** hierarchy.

Goal

Find the places of **NC** and **AC** among other complexity classes!

Agenda

- **NC** versus **AC**
- **NC** versus **P**
- **NC^1** versus **L**
- **NC^2** versus **NL**

Unbounded → bounded fan-in

Theorem

For all $k \geq 0$

$$\mathsf{NC}^k \subseteq \mathsf{AC}^k \subseteq \mathsf{NC}^{k+1}$$

Unbounded → bounded fan-in

Theorem

For all $k \geq 0$

$$\mathbf{NC^k} \subseteq \mathbf{AC^k} \subseteq \mathbf{NC^{k+1}}$$

Proof

- first inclusion trivial
- for the second, assume $L \in \mathbf{AC^k}$ by family $\{C_n\}$
- there exists a polynomial $p(n)$ such that
  - C_n has $p(n)$ gates with
  - maximal fan-in of at most $p(n)$
- each such gate can be simulated by a binary tree of gates of the same kind with depth $\log(p(n)) = \mathcal{O}(\log n)$

$\Rightarrow$ the resulting circuit has size at most size $p(n)^2$, depth at most $\log^{k+1} n$ and maximal fan-in 2

Corollary

Theorem

AC $=$ NC

Corollary

Theorem

AC = NC

Remarks

- the inclusions in the theorem on the previous slide are strict for $k = 0$
- one strict inclusion is trivial, the other one is subject of the next lecture
- for practical relevance, we focus on bounded fan-in, ie. **NC**

Agenda

- NC versus AC ✓
- NC versus P
- NC^1 versus L
- NC^2 versus NL

NC versus P

Theorem

NC $\subseteq$ **P**

Proof

- let $L \in$ **NC** by circuit family $\{C_n\}$

$\Rightarrow$ there exists a logspace TM M that computes $M(1^n) = desc(C_n)$

- the following **P** machine decides L
  - on input $x \in \{0,1\}^n$ simulate M to obtain $desc(C_n)$
  - C_n has input variables $z_1, \ldots, z_n$
  - evaluate C_n under the assignment σ that maps z_i to the $i - th$ bit of x
  - output $C_n(\sigma)$
- all steps take polynomial time (evaluation takes time proportional to circuit size)

Remarks

- P equals the set of languages with logspace-uniform circuits of polynomial size and polynomial depth (exercise)
- it is an open problem whether the previous inclusion is strict
- in fact it is open whether $\mathbf{NC}^1 \subset \mathbf{PH}$
- problem is important, since it answers whether all problems in P have fast parallel algorithms
- conjecture: strict

Agenda

- **NC** versus **AC** ✓
- **NC** versus **P** ✓
- $\mathbf{NC^1}$ versus **L**
- $\mathbf{NC^2}$ versus **NL**

Proof Steps

1. logspace reductions are transitive
2. if $L \in \mathbf{NC}^1$ then there exists a logspace uniform family of circuits $\{C_n\}$ of depth $\log n$
3. circuit evaluation of a circuit of depth d and bounded fan-in can be done in space $\mathcal{O}(d)$

What is the theorem?

What is the theorem?

Theorem

$\mathbf{NC}^1 \subseteq \mathbf{L}$.

Proof

- for a language $L \in \mathbf{NC}^1$, we can compute its circuits (step 2) in logspace
- we can evaluate circuits in logspace (step 3)
- the composition of these two algorithms is still logspace (step 1)
- steps 1 and 2 already proven

Proof of Step 3

- evaluate the circuit recursively
- identify gates with paths from output to input node
  - output node: ϵ
  - left predecessor of gate π: $\pi.0$
  - right predecessor of gate π: $\pi.1$

Proof of Step 3

- evaluate the circuit recursively
- identify gates with paths from output to input node
  - output node: ϵ
  - left predecessor of gate π: $\pi.0$
  - right predecessor of gate π: $\pi.1$
- 1. if π is an input return value
  2. if π denotes an *op* gate, compute value of $\pi.0$, value of $\pi.1$ and combine
- recursive depth $\log n$, only one global variable holding current path: total $\log n$ space
- note that the naive recursion takes $\log^2 n$ space!

Agenda

- **NC** versus **AC** ✓
- **NC** versus **P** ✓
- **NC**^{1} versus **L** ✓
- **NC**^{2} versus **NL**

The theorem

Theorem

NL $\subseteq$ **NC^2**

Proof outline

- show that Path $\in$ **NC^2**
- let $L \in$ **NL** and **NL** machine M deciding it; for a given input $x \in \{0, 1\}^*$
- build a circuit C_1 computing the adjacency matrix of M's configuration graph on input x
- build a second circuit C_2 that takes this output and decides whether there is an accepting run
- the composition of C_1 and C_2 decides L
- observe: the composition can be computed in logspace

Path $\in$ **NC^2**

- let A be the $n \times n$ adjacency matrix of a graph
- let $B = A + I$ (add self loops)
- compute the square product B^2

$$B^2_{i,j} = \bigvee_k B_{i,k} \wedge B_{k,j}$$

- contains 1 iff there is a path of length at most 2
- can be done in **AC^0** $\subseteq$ **NC^1**
- $\log n$ times repeated squaring

$\Rightarrow$ paths can be computed in **NC^2**

Agenda

- **NC** versus **AC** ✓
- **NC** versus **P** ✓
- $\mathbf{NC}^1$ versus **L** ✓
- $\mathbf{NC}^2$ versus **NL** ✓

Criticism of NC

The notion of **NC** as efficient parallel computation may be criticized.

- polynomially many processors
  - in the **NC** hierarchy a $\log n$ algorithm with n^2 processors is favored over one with n processors and time $\log^2 n$
  - expensive
- polylogarithmic depth
  - for many practical inputs, sublinear algorithms might be as good or better
  - e.g. $n^{0.1}$ is at most $\log^2 n$ for values up to 2^{100}

Summary

- **AC** $=$ **NC**
- **NC**$^1 \subseteq$ **L** $\subseteq$ **NL** $\subseteq$ **NC**$^2 \subseteq$ **P**
- up next: **AC**$^0 \subset$ **NC**^{1}

Complexity Theory

Jan Křetínský

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

Summer 2016

July 11, 2016

Lecture 24

$\mathbf{AC^0} \subset \mathbf{NC^1}$

Agenda

- lower bounds for circuits
- **$AC^0 \subset NC^1$**
- tool: random restrictions and switching lemma

Circuit lower bounds

- n is trivial
- $5n - o(n)$ for NP-complete problems
- special cases: bounded depth
- any Boolean formula by circuit of depth 2 and exponential size
- some proven to require exponential size, not valid for depth 3 any more
- do NP-complete problems have polynomial circuits with constant depth, i.e., $\mathbf{AC^0}$?

$\mathbf{AC^0} \subset \mathbf{NC^1}$

No!

$\mathbf{AC^0} \subset \mathbf{NC^1}$

No!

Theorem

$\bigoplus \notin \mathbf{AC^0}$

- $\bigoplus \in \mathbf{NC^1}$ by binary "$\oplus$-tree"
- hence $\mathbf{AC^0} \subset \mathbf{NC^1}$

Agenda

- lower bounds for circuits ✓
- $\mathbf{AC}^0 \subset \mathbf{NC}^1$ ✓
- tool: random restrictions and switching lemma

Main idea: random restrictions

- every function with **AC^0** satisfies:
- if vast majority of inputs fixed (randomly) to 0's and 1's
- then with positive probability the resulting function is constant
- but $\bigoplus$ is not!

Håstad's switching lemma

Function f under a partial assignment ρ is denoted $f|_\rho$.
Expressibility of f in k-CNF (or k-DNF) is denoted by
$f \in k$-CNF (or $f \in k$-DNF).

Theorem (Håstad's lemma, 1986)

Let $f \in k$-DNF and ρ random partial assignment to t random input bits.
Then $Pr_\rho[f|_\rho \notin s\text{-}CNF] \leq \left(\frac{(n-t)}{n} k^{10}\right)^{s/2}$ for every $s \geq 2$.

Håstad's switching lemma

Function f under a partial assignment ρ is denoted $f|_\rho$.
Expressibility of f in k-CNF (or k-DNF) is denoted by
$f \in k$-CNF (or $f \in k$-DNF).

Theorem (Håstad's lemma, 1986)

Let $f \in k$-DNF and ρ random partial assignment to t random input bits.
Then $Pr_\rho[f|_\rho \notin s\text{-}CNF] \leq \left(\frac{(n-t)}{n} k^{10}\right)^{s/2}$ for every $s \geq 2$.

- similarly for CNF
- restriction allows for switching between DNF and CNF without much blowup
- proof idea: 1-to-1 mapping of "bad" partial assignments (non-constant results) to "good" partial completions (constant results)

Proof sketch of $\bigoplus \notin$ AC^0

- start with any **AC^0** circuit (in alternating form)
- in ith round:
- fix $n_i - \sqrt{n_i}$ input bits ($n_0 = n$)
- switch the two bottom layers into the other normal form
- collapse with the layer one above

Proof sketch of $\bigoplus \notin$ **AC^0**

- start with any **AC^0** circuit (in alternating form)
- in ith round:
- fix $n_i - \sqrt{n_i}$ input bits ($n_0 = n$)
- switch the two bottom layers into the other normal form
- collapse with the layer one above
- finally, obtain two-layer DNF
- and make it constant (by fixing $\leq k$ variables in the first clause)

Proof sketch of $\bigoplus \notin \mathbf{AC}^0$

- start with any $\mathbf{AC}^0$ circuit (in alternating form)
- in ith round:
- fix $n_i - \sqrt{n_i}$ input bits ($n_0 = n$)
- switch the two bottom layers into the other normal form
- collapse with the layer one above
- finally, obtain two-layer DNF
- and make it constant (by fixing $\leq k$ variables in the first clause)
- but $\bigoplus$ cannot be made constant for any partial assignment

What have we learnt?

- lower bounds are hard
- in special simple cases possible
- tool: random partial assignments

Complexity Theory

Mikhail Raskin, Jan Křetínský

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

Summer 2019

June 14, 2019

Lecture 24'

$\mathbf{AC^0} \subset \mathbf{NC^1}$: original proof

(Furst-Saxe-Sipser 1984)

Agenda

Tool: still random assignments

Separate arguments for *wide* and *narrow* conjunctions/disjunctions

Agenda

Tool: still random assignments

Separate arguments for *wide* and *narrow* conjunctions/disjunctions

- circuits to trees
- make bottom layer fan-in bounded
- make bottom *two-layer* subtrees bounded
- reduce depth

Circuits to trees

Poly-size fixed-depth circuits (Cn).

Circuits to trees

Poly-size fixed-depth circuits (C_n).
Convert to trees (fan-out 1 except for inputs),

Circuits to trees

Poly-size fixed-depth circuits (Cn).
Convert to trees (fan-out 1 except for inputs),
all negations on the bottom level,

Circuits to trees

Poly-size fixed-depth circuits (Cn).
Convert to trees (fan-out 1 except for inputs),
all negations on the bottom level,
conjunctions and disjunctions in alternating layers.

Circuits to trees

Poly-size fixed-depth circuits (Cn).
Convert to trees (fan-out 1 except for inputs),
all negations on the bottom level,
conjunctions and disjunctions in alternating layers.
(Still poly-size fixed-depth, depth did not increase)

Circuits to trees

Poly-size fixed-depth circuits (Cn).
Convert to trees (fan-out 1 except for inputs),
all negations on the bottom level,
conjunctions and disjunctions in alternating layers.
(Still poly-size fixed-depth, depth did not increase)

Paths from root: just a polynomial number (depth is fixed)
Copy subgraphs as needed until we get trees
Same depth, still polynomial size

Circuits to trees

Poly-size fixed-depth circuits (Cn).

Circuits to trees

Poly-size fixed-depth circuits (Cn).
Convert to trees (fan-out 1 except for inputs),

Circuits to trees

Poly-size fixed-depth circuits (Cn).
Convert to trees (fan-out 1 except for inputs),
all negations on the bottom level,

Circuits to trees

Poly-size fixed-depth circuits (Cn).
Convert to trees (fan-out 1 except for inputs),
all negations on the bottom level,
conjunctions and disjunctions in alternating layers.

Circuits to trees

Poly-size fixed-depth circuits (Cn).
Convert to trees (fan-out 1 except for inputs),
all negations on the bottom level,
conjunctions and disjunctions in alternating layers.
(Still poly-size fixed-depth, depth did not increase)

Circuits to trees

Poly-size fixed-depth circuits (Cn).
Convert to trees (fan-out 1 except for inputs),
all negations on the bottom level,
conjunctions and disjunctions in alternating layers.
(Still poly-size fixed-depth, depth did not increase)

Negations: push down

Circuits to trees

Poly-size fixed-depth circuits (Cn).
Convert to trees (fan-out 1 except for inputs),
all negations on the bottom level,
conjunctions and disjunctions in alternating layers.
(Still poly-size fixed-depth, depth did not increase)

Negations: push down
Adjacent conjunctions, adjacent disjunctions: merge

Circuits to trees

Poly-size fixed-depth circuits (Cn).
Convert to trees (fan-out 1 except for inputs),
all negations on the bottom level,
conjunctions and disjunctions in alternating layers.
(Still poly-size fixed-depth, depth did not increase)

Negations: push down
Adjacent conjunctions, adjacent disjunctions: merge

Depth: number of conjunction/disjunction layers

Assume we have a sequence of minimal depth

Agenda

- circuits to trees ✓
- make bottom layer fan-in bounded
- make bottom *two-layer* subtrees bounded
- reduce depth

Bounded bottom layer fan-in

Assign $*$ ($Pr = n^{-1/2}$), 0 and 1 (equal probability) to input variables.
Circuit size: n^k

Bounded bottom layer fan-in

Assign $*$ ($Pr = n^{-1/2}$), 0 and 1 (equal probability) to input variables.
Circuit size: n^k

Goals:

- $\geqslant \sqrt{n}/2$ variables left free ($*$)
- all bottom operations have fan-in $< c$ (c will be $8k$)

The restricted circuit still calculates parity.

Bounded bottom layer fan-in

Assign $*$ ($Pr = n^{-1/2}$), 0 and 1 (equal probability) to input variables.
Circuit size: n^k

Goals:

- $\geqslant \sqrt{n}/2$ variables left free ($*$)
- all bottom operations have fan-in $< c$ (c will be $8k$)

The restricted circuit still calculates parity.

Estimate for a single operation
Separately wide ($\geqslant c \log n$) and narrow cases

Bottom layer: cases

Wide:
$> 1/3$ probability per assignment to become constant
Avoiding: $(2/3)^{c \log n} = o(n^{-c/4})$

Bottom layer: cases

Wide:
$> 1/3$ probability per assignment to become constant
Avoiding: $(2/3)^{c \log n} = o(n^{-c/4})$

Narrow:
Having $> c$ rare $*$ entries
$Pr \leqslant \binom{c \log n}{c} \sqrt{n}^{-c} \leqslant (c \log n)^c n^{-c/2} = o(n^{-c/4})$

Bottom layer: cases

Wide:
$>1/3$ probability per assignment to become constant
Avoiding: $(2/3)^{c \log n} = o(n^{-c/4})$

Narrow:
Having $> c$ rare $*$ entries
$Pr \leqslant \binom{c \log n}{c} \sqrt{n}^{-c} \leqslant (c \log n)^c n^{-c/2} = o(n^{-c/4})$

$c = 8k$, only n^k sources of problems, union bound

Bottom layer: result

Probability of $< \sqrt{n}/2$ assignments of $*$ is also small
By union bound: we still have optimal depth, worse polynomial size

Agenda

- circuits to trees ✓
- make bottom layer fan-in bounded ✓
- make bottom *two-layer* subtrees bounded
- reduce depth

Bottom two layers

Reassign k

We have minimal-depth n^k-sized tree circuits for parity with fan-in c in the bottom layer

Assign $*$ ($Pr = n^{-1/2}$), 0 and 1 (equal probability) to input variables.

Circuit size: n^k

Bottom two layers

Reassign k
We have minimal-depth n^k-sized tree circuits for parity with fan-in c in the bottom layer
Assign $*$ ($Pr = n^{-1/2}$), 0 and 1 (equal probability) to input variables.
Circuit size: n^k

Goals:

- $\geqslant \sqrt{n}/2$ variables left free ($*$)
- all layer-two operations depend on $b(c)$ variables

The restricted circuit still calculates parity.

Bottom two layers: proof

Induction on c, $c = 1$ is the previous case
$b(c) = k \times 4^c$

Bottom two layers: proof

Induction on c, $c = 1$ is the previous case
$b(c) = k \times 4^c$

Wide: $> b \log n$ disjoint bottom-level argument nodes
Probability of constant: $(1 - 4^{-c})^{b \log n} = n^{b \log 1 - 4^{-c}} \leqslant n^{-b4^{-c}} = o(n^{-k})$

Bottom two layers: proof

Induction on c, $c = 1$ is the previous case
$b(c) = k \times 4^c$

Wide: $> b \log n$ disjoint bottom-level argument nodes
Probability of constant: $(1 - 4^{-c})^{b \log n} = n^{b \log 1 - 4^{-c}} \leqslant n^{-b4^{-c}} = o(n^{-k})$

Narrow: Maximal collection of input-disjoint argument nodes
Set of their inputs: H
H hits each argument node
Fixing values of $*$ in H: by induction, dependency on $b(c-1)$ inputs

Bottom two layers: proof

Induction on c, $c = 1$ is the previous case
$b(c) = k \times 4^c$

Wide: $> b \log n$ disjoint bottom-level argument nodes
Probability of constant: $(1 - 4^{-c})^{b \log n} = n^{b \log 1 - 4^{-c}} \leqslant n^{-b4^{-c}} = o(n^{-k})$

Narrow: Maximal collection of input-disjoint argument nodes
Set of their inputs: H
H hits each argument node
Fixing values of $*$ in H: by induction, dependency on $b(c-1)$ inputs
$|H| \leqslant b(c)c \log n$; Probably $< 4k$ entries of $*$; dependency on
$4k + 2^{4k} b(c-1)$ is OK.

Agenda

- circuits to trees ✓
- make bottom layer fan-in bounded ✓
- make bottom *two-layer* subtrees bounded ✓
- reduce depth

Depth reduction

Second layer elements depend on fixed number of inputs — brute force CNF/DNF, polynomial blowup, lower depth
Contradiction!

Agenda

- circuits to trees ✓
- make bottom layer fan-in bounded ✓
- make bottom *two-layer* subtrees bounded ✓
- reduce depth ✓

Complexity Theory

Jan Křetínský

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

Summer 2016

July 11, 2016

Lecture 25

Counting

Agenda

- examples of counting problems
- definition
- how hard are they?

Examples

Deciding is easy, counting is hard

Example (#CYCLE)

Number of simple cycles

- cycle detection in linear time
- if $\#\mathit{CYCLE}$ has a polynomial algorithm then **P** $=$ **NP**

Examples

Deciding is easy, counting is hard

Example (#CYCLE)

Number of simple cycles

- cycle detection in linear time
- if $\#CYCLE$ has a polynomial algorithm then $\mathbf{P} = \mathbf{NP}$

Example (GraphReliability)

$\frac{1}{2^n}\cdot$ number of subgraphs with a path from s to t

Example (Maximum likelyhood in Bayes nets)

Visible variables are $\vee$'s of ≤ 3 hidden variables.
What is the fraction of satisfying assignments with $x_1 = 1$?

- equivalent to $\#SAT$

Definition

Definition ($\#$P)

A function $f : \{0,1\}^* \to \mathbb{N}$ is in $\#$P if there is a polynomial-time TM M and a polynomial p such that $\forall x \in \{0,1\}^*$

$$f(x) = \left|\left\{y \in \{0,1\}^{p(|x|)} : M(x,y) = 1\right\}\right|$$

- counting certificates
- or accepting paths

Definition (FP)

A function $f : \{0,1\}^* \to \mathbb{N}$ is in FP if there is a deterministic polynomial-time TM computing f.

- efficeintly solvable counting

Decision analog

Theorem

$\mathbf{FP} = \#\mathbf{P}$

Decision analog

Theorem

$\mathbf{FP} = \#\mathbf{P} \iff$

Decision analog

Theorem

$\mathbf{FP} = \#\mathbf{P} \iff \mathbf{P} = \mathbf{PP}$

Completeness

Definition

A function f is $\#\mathbf{P}$-complete if $f \in \#\mathbf{P}$ and for every $g \in \#\mathbf{P}$ we have $g \in \mathbf{FP}^f$

- $\#SAT$ is $\#\mathbf{P}$-complete

Completeness

Definition

A function f is $\#\mathbf{P}$-complete if $f \in \#\mathbf{P}$ and for every $g \in \#\mathbf{P}$ we have $g \in \mathbf{FP}^f$

- $\#SAT$ is $\#\mathbf{P}$-complete

Example (Determinant)

$det(A) = \sum_{\sigma \in S_n} sgn(\sigma) \prod_{i=1}^{n} A_{i,\sigma(i)}$

- computable in polynomial time

Example (Permanent)

$perm(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i,\sigma(i)}$

- $\#\mathbf{P}$-complete (for 0,1 matrices) [Valiant'79]
- hence $perm \in \mathbf{FP} \implies \mathbf{P} = \mathbf{NP}$

Toda's theorem

Theorem (Toda'91)

$\mathbf{PH} \subseteq \mathbf{P}^{\#SAT}$

Proof idea

- randomized reduction from **PH** to $\oplus SAT$
  (odd number of satisfying assignments; $\oplus$**P**-complete problem)
- derandomization

What have we learnt?

- counting seems harder than deciding
- $\#$**P**-complete problems arise from **NP**-complete problems as well as from those in **P**
- more powerful than alternating quantifiers
- classes **PP** and $\oplus$**P**: most and least significant bits of $\#$**P** function