Complexity Theory

Jan Křetínský

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

Summer 2016

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 $\in \Sigma_2^p$.

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.

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

Examples

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

Examples

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

Examples

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

Oracles and PH

Recall that

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

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

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

$\mathbf{\Sigma_3^p} = \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}}$

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:
  - $\mathbf{P}^X \neq \mathbf{NP}^X$
  - $\mathbf{P}^Y = \mathbf{NP}^Y$ (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:
  - $\mathbf{P}^X \neq \mathbf{NP}^X$
  - $\mathbf{P}^Y = \mathbf{NP}^Y$ (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 → 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^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 Σ_iP 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, $\Sigma_i^p = \Sigma_i$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*
$\mathbf{ASPACE}(f(n)) = \mathbf{TIME}(2^{O(f(n))})$.

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
$\mathbf{ASPACE}(f(n)) = \mathbf{TIME}(2^{O(f(n))})$.

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

- $\mathbf{ATIME}(f(n)) \subseteq \mathbf{SPACE}(f(n)$
  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
- $\mathbf{ASPACE}(f(n)) \supseteq \mathbf{TIME}(2^{O(f(n))})$

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**$(2^{O(f(n))})$
  configuration graph + "attractor" construction
- **ASPACE**$(f(n)) \supseteq$ **TIME**$(2^{O(f(n))})$
  guess and check the tableaux of the computation
  (+ halting state on the left)

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)