Complexity Theory

Jan Křetínský

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

Summer 2016

Based on slides by Jörg Kreiker

Lecture 8

PSPACE

Agenda

- Wrap-up Ladner proof and time vs. space
- succinctness
- QBF and GG
- **PSPACE** completeness
- QBF is **PSPACE**-complete
- Savitch's theorem

Comments about previous lecture

Enumeration of languages in **P**:

- enumerate pairs $\langle M_i, p_j \rangle$
- i enumerates all TMs, j all polynomials
- run M_i for p_j steps

Comments about previous lecture

Enumeration of languages in **P**:

- enumerate pairs $\langle M_i, p_j \rangle$
- i enumerates all TMs, j all polynomials
- run M_i for p_j steps

Time vs Space

- based on configuration graphs one can also show **DTIME**$(s(n)) \subseteq$ **NTIME**$(s(n)) \subseteq$ **SPACE**$(s(n))$
- if configurations include a counter over all possible choices

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 $\psi = x \vee y \vee \overline{z}$

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

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 $\psi = x \vee y \vee \overline{z}$

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

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, ie. 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 ψ = Qx.ψ′
        if Q = ∃
            if qbfsolve(ψ′[x ↦ 0]) return true
            if qbfsolve(ψ′[x ↦ 1]) return true
        if Q = ∀
            b1 = qbfsolve(ψ′[x ↦ 0])
            b2 = qbfsolve(ψ′[x ↦ 1])
            return b1 ∧ b2
return false
```

QBF is in PSPACE

Polynomial space algorithm to decide QBF

```
qbfsolve(ψ)
    if ψ is quantifier-free
        return evaluation of ψ
    if ψ = Qx.ψ′
        if Q = ∃
            if qbfsolve(ψ′[x ↦ 0]) return true
            if qbfsolve(ψ′[x ↦ 1]) return true
        if Q = ∀
            b1 = qbfsolve(ψ′[x ↦ 0])
            b2 = qbfsolve(ψ′[x ↦ 1])
            return b1 ∧ b2
    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 $\rightarrow$ 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 $\bigwedge_i \overline{b_i}$

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 $\bigwedge_i \overline{b_i}$

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

Agenda

- Wrap-up Ladner proof and time vs. space ✓
- 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, 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

$$\psi_i(C, C') = \exists C''.\forall D_1.\forall D_2. \; ((D_1 = C \wedge D_2 = C'') \vee (D_1 = C'' \wedge D_2 = C')) \Rightarrow \psi_{i-1}(D_1, D_2)$$

- C'' stands for m variables

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

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

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* $s(n) \geq \log n$
NSPACE$(s(n)) \subseteq$ **SPACE**$(s(n)^2)$.

Proof

Let M be a NDTM accepting L. Let $G(M, x)$ be its configuration graph of size $m\mathcal{O}(2^{s(n)})$ such that 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 $m\mathcal{O}(2^{s(n)})$ such that 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 $m\mathcal{O}(2^{s(n)})$ such that 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 = reach$(u, z, i - 1)$
  - b_2 = 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 $m\mathcal{O}(2^{s(n)})$ such that 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 = reach$(u, z, i-1)$
  - b_2 = reach$(z, v, i-1)$
  - return $b_1 \wedge b_2$

$\Rightarrow$ 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. Proceedings of the 5th Symposium on Theory of Computing, 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

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**