COMP3630 / COMP6363

week 10: **Other Complexity Classes**

This Lecture Covers Chapter 11 of HMU: Other Complexity Classes

slides created by: Dirk Pattinson, based on material by
Peter Hoefner and Rob van Glabbeck; with improvements by Pascal Bercher

convenor & lecturer: Pascal Bercher

The Australian National University

Semester 1, 2023

Content of this Chapter

- **PSPACE**-completeness
- Quantified Boolean Formulae (QBF)
- QBF is **PSPACE**-complete

Additional Reading: Chapter 11 of HMU.

PSPACE-completeness

Definition 11.1.1

A problem L is **PSPACE**-hard if there is a polytime reduction from any **PSPACE** problem to L.

A problem L is **PSPACE**-complete, if it is **PSPACE**-hard and in **PSPACE**.

Q. Why polytime, and not polyspace reductions?

PSPACE-completeness

Definition 11.1.1

A problem L is **PSPACE**-hard if there is a polytime reduction from any **PSPACE** problem to L.

A problem L is **PSPACE**-complete, if it is **PSPACE**-hard and in **PSPACE**.

Q. Why polytime, and not polyspace reductions?
A. As usual: otherwise the translation process could solve the problem.

PSPACE-completeness

Definition 11.1.1

A problem L is **PSPACE**-hard if there is a polytime reduction from any **PSPACE** problem to L.

A problem L is **PSPACE**-complete, if it is **PSPACE**-hard and in **PSPACE**.

Q. Why polytime, and not polyspace reductions?
A. As usual: otherwise the translation process could solve the problem.

Observation.

Let L be a **PSPACE**-complete problem.

1. If $L \in \mathbf{P}$, then $\mathbf{P} = \mathbf{PSPACE}$. (And thus $\mathbf{P} = \mathbf{NP}$)
2. if $L \in \mathbf{NP}$, then $\mathbf{NP} = \mathbf{PSPACE}$.

Quantified Boolean Formulae (QBFs)

Definition 11.2.1

If V is a set of variables, then the set of quantified boolean formulae over V is given by:

- Every variable $v \in V$ is a QBF, and so are $\top$ and $\bot$
- If ϕ, ψ are QBF, then so are $\phi \wedge \psi$ and $\phi \vee \psi$
- If ϕ is a QBF, then so is $\neg\phi$.
- If ϕ is a QBF and $v \in V$ is a variable, then $\exists v \phi$ and $\forall v \psi$ are QBF.

Quantified Boolean Formulae (QBFs)

Definition 11.2.1

If V is a set of variables, then the set of quantified boolean formulae over V is given by:

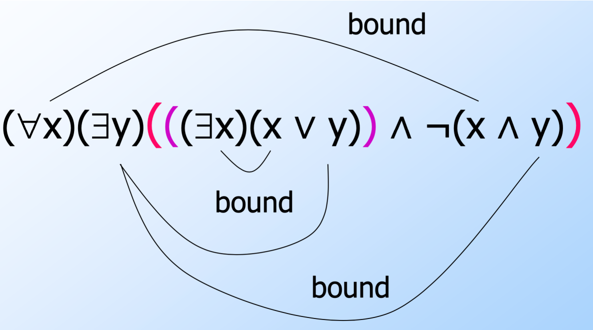
- Every variable $v \in V$ is a QBF, and so are $\top$ and $\bot$
- If ϕ, ψ are QBF, then so are $\phi \wedge \psi$ and $\phi \vee \psi$
- If ϕ is a QBF, then so is $\neg\phi$.
- If ϕ is a QBF and $v \in V$ is a variable, then $\exists v \phi$ and $\forall v \psi$ are QBF.

Definition 11.2.2

In a QBF ϕ, a variable v is bound if it is in the scope of a quantifier $\forall v$ or $\exists v$. The variable v is free otherwise.

If $x \in \{\top, \bot\}$ is a truth value, then $\phi[x/v]$ is the result of replacing all free occurrences of v with x.

Example



- Usually, one writes these formulae without the parentheses pairs around the quantified variables, e.g, $\forall x \phi$ instead of $(\forall x)\phi$.
- Note how inner quantifiers have precedence over outer ones.
- Also, this formula does not have free variables, i.e., all are bound.

Evaluation of QBFs

Observation.

A QBF ϕ without free variables can be evaluated to a truth value:

- $\text{evalQBF}(\forall v \phi) = \phi[\top/v] \wedge \phi[\bot/v]$
- $\text{evalQBF}(\exists v \phi) = \phi[\top/v] \vee \phi[\bot/v]$

and quantifier-free formulae without free variables can be evaluated.

Evaluation of QBFs

Observation.

A QBF ϕ without free variables can be evaluated to a truth value:

- $\text{evalQBF}(\forall v \phi) = \phi[\top/v] \wedge \phi[\bot/v]$
- $\text{evalQBF}(\exists v \phi) = \phi[\top/v] \vee \phi[\bot/v]$

and quantifier-free formulae without free variables can be evaluated.

QBFs versus boolean formulae.

A boolean formula ϕ with variables $v_1, \ldots, v_n$ is:

- satisfiable if $\exists v_1 \exists v_2 \ldots \exists v_n \phi$ evaluates to true.
- a tautology if $\forall v_1 \forall v_2 \ldots \forall v_n \phi$ evaluates to true.

Evaluation of QBFs

Observation.

A QBF ϕ without free variables can be evaluated to a truth value:

- $\text{evalQBF}(\forall v \phi) = \phi[\top/v] \wedge \phi[\bot/v]$
- $\text{evalQBF}(\exists v \phi) = \phi[\top/v] \vee \phi[\bot/v]$

and quantifier-free formulae without free variables can be evaluated.

QBFs versus boolean formulae.

A boolean formula ϕ with variables $v_1, \ldots, v_n$ is:

- satisfiable if $\exists v_1 \exists v_2 \ldots \exists v_n \phi$ evaluates to true.
- a tautology if $\forall v_1 \forall v_2 \ldots \forall v_n \phi$ evaluates to true.

Definition 11.2.3

The QBF problem is the problem of determining whether a given quantified boolean formula without free variables evaluates to true:

$$\text{QBF} = \{\langle \phi \rangle \mid \phi \text{ a true QBF without free variables}\}$$

QBFs vs Boolean Formulae

- Evaluating a boolean formula without free variables (i.e., with variables substituted by $\top$ or $\bot$) is in **P**.

QBFs vs Boolean Formulae

- Evaluating a boolean formula without free variables (i.e., with variables substituted by $\top$ or $\bot$) is in **P**.
- So, an idea is to substitute all bound variables by its truth values:
  - $(\forall v \phi) \rightsquigarrow \phi[\top/x] \wedge \phi[\bot/x]$
  - $(\exists v \phi) \rightsquigarrow \phi[\top/x] \vee \phi[\bot/x]$

QBFs vs Boolean Formulae

- Evaluating a boolean formula without free variables (i.e., with variables substituted by $\top$ or $\bot$) is in **P**.
- So, an idea is to substitute all bound variables by its truth values:
  - $(\forall v \phi) \rightsquigarrow \phi[\top/x] \wedge \phi[\bot/x]$
  - $(\exists v \phi) \rightsquigarrow \phi[\top/x] \vee \phi[\bot/x]$
- But due to doubling the formula with each substitution, the resulting formula may be exponentially large. So we showed that QBF is in **EXPTIME**.

Q. Can we do better?

QBF is in **PSPACE**

Main Idea.

- to evaluate $\forall v \phi$, <u>don't</u> write out $\phi[\top/v] \wedge \phi[\bot/v]$.
- instead, evaluate $\phi[\top/v]$ and $\phi[\bot/v]$ in sequence.
- avoids exponential space blowup

Recursive Algorithm evalQBF(ϕ)

QBF is in **PSPACE**

Main Idea.

- to evaluate $\forall v \phi$, don't write out $\phi[\top/v] \wedge \phi[\bot/v]$.
- instead, evaluate $\phi[\top/v]$ and $\phi[\bot/v]$ in sequence.
- avoids exponential space blowup

Recursive Algorithm evalQBF(ϕ)

- case $\phi = \top$: return $\top$

QBF is in **PSPACE**

Main Idea.

- to evaluate $\forall v \phi$, don't write out $\phi[\top/v] \wedge \phi[\bot/v]$.
- instead, evaluate $\phi[\top/v]$ and $\phi[\bot/v]$ in sequence.
- avoids exponential space blowup

Recursive Algorithm evalQBF(ϕ)

- case $\phi = \top$: return $\top$
- case $\phi = (\psi_1 \wedge \psi_2)$: if evalQBF($\psi_1$) then return evalQBF($\psi_2$) else return $\bot$

QBF is in **PSPACE**

Main Idea.

- to evaluate $\forall v \phi$, don't write out $\phi[\top/v] \wedge \phi[\bot/v]$.
- instead, evaluate $\phi[\top/v]$ and $\phi[\bot/v]$ in sequence.
- avoids exponential space blowup

Recursive Algorithm evalQBF(ϕ)

- case $\phi = \top$: return $\top$
- case $\phi = (\psi_1 \wedge \psi_2)$: if evalQBF($\psi_1$) then return evalQBF($\psi_2$) else return $\bot$
- case $\phi = \forall v \psi$: if evalQBF($\psi[\top/v]$) then return evalQBF($\phi[\bot/v]$) else return $\bot$

QBF is in **PSPACE**

Main Idea.

- to evaluate $\forall v\phi$, don't write out $\phi[\top/v] \wedge \phi[\bot/v]$.
- instead, evaluate $\phi[\top/v]$ and $\phi[\bot/v]$ in sequence.
- avoids exponential space blowup

Recursive Algorithm evalQBF(ϕ)

- case $\phi = \top$: return $\top$
- case $\phi = (\psi_1 \wedge \psi_2)$: if evalQBF($\psi_1$) then return evalQBF($\psi_2$) else return $\bot$
- case $\phi = \forall v\psi$: if evalQBF($\psi[\top/v]$) then return evalQBF($\phi[\bot/v]$) else return $\bot$
- other cases: analogous

QBF is in **PSPACE**

Main Idea.

- to evaluate $\forall v \phi$, don't write out $\phi[\top/v] \wedge \phi[\bot/v]$.
- instead, evaluate $\phi[\top/v]$ and $\phi[\bot/v]$ in sequence.
- avoids exponential space blowup

Recursive Algorithm evalQBF(ϕ)

- case $\phi = \top$: return $\top$
- case $\phi = (\psi_1 \wedge \psi_2)$: if evalQBF($\psi_1$) then return evalQBF($\psi_2$) else return $\bot$
- case $\phi = \forall v \psi$: if evalQBF($\psi[\top/v]$) then return evalQBF($\phi[\bot/v]$) else return $\bot$
- other cases: analogous

Analysis.

Given QBF ϕ of size n:

- at most n recursive calls active
- each call stores a partially evaluated QBF of size n
- total space requirement $\mathcal{O}(n^2)$

QBF is **PSPACE**-hard (and hence -complete)

Proof Idea/Overview.

Reduce any problem in **PSPACE** to QBF:

- Let L be in **PSPACE**.
- Then L is accepted by a polyspace-bounded TM with bound $p(n)$.
- If $w \in L$, then M accepts in $\leq c^{p(n)}$ moves.
- Construct QBF ϕ: "there is a sequence of $c^{p(n)}$ ID's that accepts w".
- Use recursive doubling to perform this reduction in polytime.

(Detailed encoding in next two slides. Shows similarities to Cook's *SAT* encoding.)

The Gory Detail

Variables.

- We use two sets of variables, $x_{j,s}$ and $y_{j,s}$. Need $\mathcal{O}(p(n))$ variables to represent an ID:
- variables $x_{j,s}/y_{j,s} = \top$ iff the j-th symbol of the resp. ID is s, $1 \leq j \leq p(n) + 1$.

The Gory Detail

Variables.

- We use two sets of variables, $x_{j,s}$ and $y_{j,s}$. Need $\mathcal{O}(p(n))$ variables to represent an ID:
- variables $x_{j,s}/y_{j,s} = \top$ iff the j-th symbol of the resp. ID is s, $1 \leq j \leq p(n)+1$.

Structure of the QBF.

$$\phi = (\exists X)(\exists Y)(S \wedge N \wedge F \wedge U)$$

- We use X as the tuple of all x-variables, and Y as the tuple of all y-variables. They will be used to encode the initial and final configuration.

The Gory Detail

Variables.

- We use two sets of variables, $x_{j,s}$ and $y_{j,s}$. Need $\mathcal{O}(p(n))$ variables to represent an ID:
- variables $x_{j,s}/y_{j,s} = \top$ iff the j-th symbol of the resp. ID is s, $1 \leq j \leq p(n) + 1$.

Structure of the QBF.

$$\phi = (\exists X)(\exists Y)(S \wedge N \wedge F \wedge U)$$

- We use X as the tuple of all x-variables, and Y as the tuple of all y-variables. They will be used to encode the initial and final configuration.
  - $(\exists \mathbf{X})$ is short for $\exists x_{0,q_0} \ldots \exists x_{0,q_{|Q|}} \ldots \exists x_{p(n),q_0} \ldots \exists x_{p(n),q_{|Q|}}$, i.e., we quantify all x variables.
  - $(\exists \mathbf{Y})$ is the very same as X, but works on all the y variables instead.

The Gory Detail

Variables.

- We use two sets of variables, $x_{j,s}$ and $y_{j,s}$. Need $\mathcal{O}(p(n))$ variables to represent an ID:
- variables $x_{j,s}/y_{j,s} = \top$ iff the j-th symbol of the resp. ID is s, $1 \leq j \leq p(n)+1$.

Structure of the QBF.

$$\phi = (\exists X)(\exists Y)(S \wedge N \wedge F \wedge U)$$

- We use X as the tuple of all x-variables, and Y as the tuple of all y-variables. They will be used to encode the initial and final configuration.
  - $(\exists \mathbf{X})$ is short for $\exists x_{0,q_0} \ldots \exists x_{0,q_{|Q|}} \ldots \exists x_{p(n),q_0} \ldots \exists x_{p(n),q_{|Q|}}$, i.e., we quantify all x variables.
  - $(\exists \mathbf{Y})$ is the very same as X, but works on all the y variables instead.
- **S**: says that X initially represents $ID_0 = q_0 w$, just as in Cook's theorem.
  $x_{0,q_0} \wedge x_{1,w_1} \cdots \wedge x_{k,w_{|w|}} \wedge y_{|w|+1,B} \wedge \cdots \wedge y_{p(n),B}$

The Gory Detail

Variables.

- We use two sets of variables, $x_{j,s}$ and $y_{j,s}$. Need $\mathcal{O}(p(n))$ variables to represent an ID:
- variables $x_{j,s}/y_{j,s} = \top$ iff the j-th symbol of the resp. ID is s, $1 \leq j \leq p(n)+1$.

Structure of the QBF.

$$\phi = (\exists X)(\exists Y)(S \wedge N \wedge F \wedge U)$$

- We use X as the tuple of all x-variables, and Y as the tuple of all y-variables. They will be used to encode the initial and final configuration.
  - $(\exists \mathbf{X})$ is short for $\exists x_{0,q_0} \ldots \exists x_{0,q_{|Q|}} \ldots \exists x_{p(n),q_0} \ldots \exists x_{p(n),q_{|Q|}}$, i.e., we quantify all x variables.
  - $(\exists \mathbf{Y})$ is the very same as X, but works on all the y variables instead.
- **S**: says that X initially represents $ID_0 = q_0 w$, just as in Cook's theorem.
  $x_{0,q_0} \wedge x_{1,w_1} \cdots \wedge x_{k,w_{|w|}} \wedge y_{|w|+1,B} \wedge \cdots \wedge y_{p(n),B}$
- **F**: says that Y represents an accepting ID ID_f, just as in Cook's theorem.
  $\bigvee_{\substack{0 \leq i \leq p(n) \\ q \text{ accepting}}} y_{i,q}$

The Gory Detail

Variables.

- We use two sets of variables, $x_{j,s}$ and $y_{j,s}$. Need $\mathcal{O}(p(n))$ variables to represent an ID:
- variables $x_{j,s}/y_{j,s} = \top$ iff the j-th symbol of the resp. ID is s, $1 \leq j \leq p(n)+1$.

Structure of the QBF.

$$\phi = (\exists X)(\exists Y)(S \wedge N \wedge F \wedge U)$$

- We use X as the tuple of all x-variables, and Y as the tuple of all y-variables. They will be used to encode the initial and final configuration.
  - $(\exists \mathbf{X})$ is short for $\exists x_{0,q_0} \ldots \exists x_{0,q_{|Q|}} \ldots \exists x_{p(n),q_0} \ldots \exists x_{p(n),q_{|Q|}}$, i.e., we quantify all x variables.
  - $(\exists \mathbf{Y})$ is the very same as X, but works on all the y variables instead.
- **S**: says that X initially represents $ID_0 = q_0 w$, just as in Cook's theorem.
  $x_{0,q_0} \wedge x_{1,w_1} \cdots \wedge x_{k,w_{|w|}} \wedge y_{|w|+1,B} \wedge \cdots \wedge y_{p(n),B}$
- **F**: says that Y represents an accepting ID ID_f, just as in Cook's theorem.
  $\bigvee_{\substack{0 \leq i \leq p(n) \\ q \text{ accepting}}} y_{i,q}$
- **U**: says that every ID has at most one symbol per position, just as in Cook's theorem.

The Gory Detail

Variables.

- We use two sets of variables, $x_{j,s}$ and $y_{j,s}$. Need $\mathcal{O}(p(n))$ variables to represent an ID:
- variables $x_{j,s}/y_{j,s} = \top$ iff the j-th symbol of the resp. ID is s, $1 \leq j \leq p(n) + 1$.

Structure of the QBF.

$$\phi = (\exists X)(\exists Y)(S \land N \land F \land U)$$

- We use X as the tuple of all x-variables, and Y as the tuple of all y-variables. They will be used to encode the initial and final configuration.
  - $(\exists \mathbf{X})$ is short for $\exists x_{0,q_0} \dots \exists x_{0,q_{|Q|}} \dots \exists x_{p(n),q_0} \dots \exists x_{p(n),q_{|Q|}}$, i.e., we quantify all x variables.
  - $(\exists \mathbf{Y})$ is the very same as X, but works on all the y variables instead.
- **S**: says that X initially represents $ID_0 = q_0 w$, just as in Cook's theorem.
  $x_{0,q_0} \land x_{1,w_1} \cdots \land x_{k,w_{|w|}} \land y_{|w|+1,B} \land \cdots \land y_{p(n),B}$
- **F**: says that Y represents an accepting ID ID_f, just as in Cook's theorem.
  $\bigvee_{\substack{0 \leq i \leq p(n) \\ q \text{ accepting}}} y_{i,q}$
- **U**: says that every ID has at most one symbol per position, just as in Cook's theorem.
- **N**: transition from $X \approx ID_0$ to some $Y \approx ID_f$ in $\leq c^{p(n)}$ steps (see next slide).

Recursive Doubling

- $N = N(ID_0, ID_f)$: have sequence of length $\leq c^{p(n)}$ from ID_0 to ID_f.
  Again, ID_0 and ID_f are just our variables X and Y, but they are, by S and F, constrained to represent the initial ID and any accepting ID.

Recursive Doubling

- $N = N(ID_0, ID_f)$: have sequence of length $\leq c^{p(n)}$ from ID_0 to ID_f.
  Again, ID_0 and ID_f are just our variables X and Y, but they are, by S and F, constrained to represent the initial ID and any accepting ID.
- Detour: $N_0(X, Y) = X \vdash^* Y$ in ≤ 1 steps: as for Cook's theorem

Recursive Doubling

- $N = N(ID_0, ID_f)$: have sequence of length $\leq c^{p(n)}$ from ID_0 to ID_f.
  Again, ID_0 and ID_f are just our variables X and Y, but they are, by S and F, constrained to represent the initial ID and any accepting ID.
- Detour: $N_0(X, Y) = X \vdash^* Y$ in ≤ 1 steps: as for Cook's theorem
- Detour: $N_i(X, Y) = X \vdash^* Y$ in $\leq 2^i$ steps:

  - Could also say $(\exists K)(N_{i-1}(X, K) \wedge N_{i-1}(K, Y))$
  - this would write out N_{i-1} twice, doubling formula size at each step
  - above trick is key step in proof to keep formula size small (prevent doubling)

Recursive Doubling

- $N = N(ID_0, ID_f)$: have sequence of length $\leq c^{p(n)}$ from ID_0 to ID_f.
  Again, ID_0 and ID_f are just our variables X and Y, but they are, by S and F, constrained to represent the initial ID and any accepting ID.
- Detour: $N_0(X, Y) = X \vdash^* Y$ in ≤ 1 steps: as for Cook's theorem
- Detour: $N_i(X, Y) = X \vdash^* Y$ in $\leq 2^i$ steps:

$$
\begin{aligned}
N_i(X, Y) = (\exists K)(\forall P)(\forall Q)[& \\
& ((P, Q) = (X, K) \vee (P, Q) = (K, Y)) \\
& \rightarrow N_{i-1}(P, Q)]
\end{aligned}
$$

  - Could also say $(\exists K)(N_{i-1}(X, K) \wedge N_{i-1}(K, Y))$
  - this would write out N_{i-1} twice, doubling formula size at each step
  - above trick is key step in proof to keep formula size small (prevent doubling)

Recursive Doubling

- $N = N(ID_0, ID_f)$: have sequence of length $\leq c^{p(n)}$ from ID_0 to ID_f.
  Again, ID_0 and ID_f are just our variables X and Y, but they are, by S and F, constrained to represent the initial ID and any accepting ID.
- Detour: $N_0(X, Y) = X \vdash^* Y$ in ≤ 1 steps: as for Cook's theorem
- Detour: $N_i(X, Y) = X \vdash^* Y$ in $\leq 2^i$ steps:

$$\begin{aligned} N_i(X, Y) = (\exists K)(\forall P)(\forall Q)[& \\ & ((P, Q) = (X, K) \vee (P, Q) = (K, Y)) \\ & \rightarrow N_{i-1}(P, Q)] \end{aligned}$$

  - Could also say $(\exists K)(N_{i-1}(X, K) \wedge N_{i-1}(K, Y))$
  - this would write out N_{i-1} twice, doubling formula size at each step
  - above trick is key step in proof to keep formula size small (prevent doubling)
- Let $N(X, Y) = N_k(X, Y)$ where $2^k \geq c^{p(n)}$ (note $k \in \mathcal{O}(p(n))$)
- each N_i can be written in $\mathcal{O}(p(n))$ many steps, plus the time to write N_{i-1}
- so $\mathcal{O}(p(n)^2)$ overall

Recursive Doubling

- $N = N(ID_0, ID_f)$: have sequence of length $\leq c^{p(n)}$ from ID_0 to ID_f.
  Again, ID_0 and ID_f are just our variables X and Y, but they are, by S and F, constrained to represent the initial ID and any accepting ID.
- Detour: $N_0(X, Y) = X \vdash^* Y$ in ≤ 1 steps: as for Cook's theorem
- Detour: $N_i(X, Y) = X \vdash^* Y$ in $\leq 2^i$ steps:

$$\begin{aligned} N_i(X, Y) = (\exists K)(\forall P)(\forall Q)[& \\ & ((P, Q) = (X, K) \vee (P, Q) = (K, Y)) \\ & \rightarrow N_{i-1}(P, Q)] \end{aligned}$$

  - Could also say $(\exists K)(N_{i-1}(X, K) \wedge N_{i-1}(K, Y))$
  - this would write out N_{i-1} twice, doubling formula size at each step
  - above trick is key step in proof to keep formula size small (prevent doubling)
- Let $N(X, Y) = N_k(X, Y)$ where $2^k \geq c^{p(n)}$ (note $k \in \mathcal{O}(p(n))$)
- each N_i can be written in $\mathcal{O}(p(n))$ many steps, plus the time to write N_{i-1}
- so $\mathcal{O}(p(n)^2)$ overall

By construction, $\phi = \top$ iff M accepts w.