אוניברסיטת
אריאל
בשומרון

Computational Complexity

Lecturer: Edward A. Hirsch

https://edwardahirsch.github.io/edwardahirsch

edward.a.hirsch@gmail.com

December 3, 2025

Lecture plan

- The Karp–Lipton Collapse Theorem: a proof.
- Circuit lower bounds for classes of the polynomial hierarchy:
  - Kannan's theorem: $\Sigma_4^p \not\subseteq \mathbf{Size}[n^k]$.
  - Consequences for Σ_2^p.
- Discussion on lower bounds and KL-style theorems.

- Space-bounded computation.
  - **PSPACE**, examples, the `QBF` language.

Polynomial-size circuits and the Karp–Lipton theorem

Definition (P/poly)

$L \in \mathbf{P/poly} \iff$ there is a family $\{C_n\}_{n\in\mathbb{N}}$ of Boolean circuits (n is the number of inputs), each C_n being of polynomial size in n, such that $\forall x \in \{0,1\}^*$

$$x \in L \iff C_{|x|}(x) = 1.$$

This is called (non-uniform) polynomial-size circuit family.

Theorem (Σ_k^p-complete language — proved last time)

$$\mathtt{QBF}_k = \{ \text{ true formulas } \exists w_1 \forall w_2 \exists \ldots Q w_k \ \Phi \ \},$$

where Φ is a CNF (if k is odd) or DNF (if k is even) in variables $w_1, \ldots, w_k$
(each of these strings contains many Boolean variables! $\exists w_1 \sim \exists w_{1,1} \exists w_{1,2} \ldots \exists w_{1,t}$).

Also known as Σ_k-QBF.

The Karp–Lipton theorem: The idea

Theorem (R. Karp, R. Lipton, M. Sipser)

If **NP** $\subseteq$ **P/poly**, then **PH** $\subseteq \Sigma_2^p$.

- It suffices to prove $\Sigma_3^p = \Sigma_2^p$ (even $\Pi_2^p = \Sigma_2^p$).
- $\exists x \forall y \exists z\ F(x, y, z)$ — how to remove a quantifier?
- If **NP** $\subseteq$ **P/poly**, then SAT has small circuits $\{S_n\}_{n \in \mathbb{N}}$.
- **It remains to find them (and use)!.. we will do it using just two quantifiers:** $\exists$ to choose a circuit and $\forall$ to verify its correctness.

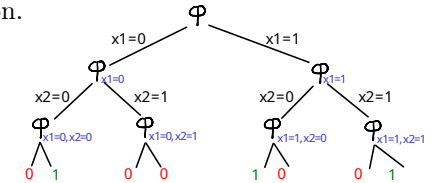
The Karp–Lipton theorem: The idea

Theorem (R. Karp, R. Lipton, M. Sipser)

If **NP** $\subseteq$ **P/poly**, then **PH** $\subseteq \Sigma_2^p$.

- It suffices to prove $\Sigma_3^p = \Sigma_2^p$ (even $\Pi_2^p = \Sigma_2^p$).
- $\exists x \forall y \exists z\ F(x, y, z)$ — how to remove a quantifier?
- If **NP** $\subseteq$ **P/poly**, then SAT has small circuits $\{S_n\}_{n \in \mathbb{N}}$.
- **It remains to find them (and use)!.. we will do it using just two quantifiers:** $\exists$ to choose a circuit and $\forall$ to verify its correctness.
- S_n is a correct SAT circuit $\iff \forall \Phi$ of size $\leqslant n$, $S_n(\Phi) = \mathtt{SAT}(\Phi)$.
- We can compute $S_n(\Phi)$ fast (S_n is small), but we cannot compute SAT...
- SAT is downward-self-reducible (i.e., reducible to its smaller instances, of course, not free). We already used it in the search-to-decision reduction.

$$F \in \mathtt{SAT} \iff F|_{x_1=0} \in \mathtt{SAT} \text{ or } F|_{x_1=1} \in \mathtt{SAT}$$



The Karp–Lipton Theorem: A formal proof

Current goal: verify circuits for SAT

- One could do the search-to-decision reduction to get not just "yes", but a sat. assignment. It would make sure that $S_n(\Phi) = 1$ is correct. But how to check the case $S_n(\Phi) = 0$?
- W.l.o.g. we encode formulas so that at length n there is also encoding of all smaller formulas (otherwise use a different circuit S_i where needed).
- Check $\forall\Phi$ of size $\leqslant n$ (here x_1 is the first variable in Φ):

$$S_n(\Phi) = S_n(\Phi|_{x_1:=0}) \vee S_n(\Phi|_{x_1:=1})$$

The Karp–Lipton Theorem: A formal proof

Current goal: verify circuits for SAT

- One could do the search-to-decision reduction to get not just "yes", but a sat. assignment. It would make sure that $S_n(\Phi) = 1$ is correct. But how to check the case $S_n(\Phi) = 0$?
- W.l.o.g. we encode formulas so that at length n there is also encoding of all smaller formulas (otherwise use a different circuit S_i where needed).
- Check $\forall \Phi$ of size $\leqslant n$ (here x_1 is the first variable in Φ):

$$S_n(\Phi) = S_n(\Phi|_{x_1:=0}) \vee S_n(\Phi|_{x_1:=1})$$

- Add $S_n(\texttt{True}) = 1$, $S_n(\texttt{False}) = 0$ for formulas without variables as the end of this induction.
- Summary: S_n solves SAT correctly for size-n formulas iff

$$\forall \Phi \text{ of size } \leqslant n \ \underbrace{S_n(\Phi) = (S_n(\Phi|_{x_1:=0}) \vee S_n(\Phi|_{x_1:=1})) \wedge S_n(\texttt{True}) = 1 \wedge S_n(\texttt{False}) = 0}.$$

- Let us call it $\mathrm{VERIFY}(S_n, \Phi)$ and use it to show the collapse.

The Karp–Lipton Theorem: A formal proof

Current goal: verify circuits for SAT

- One could do the search-to-decision reduction to get not just "yes", but a sat. assignment. It would make sure that $S_n(\Phi) = 1$ is correct. But how to check the case $S_n(\Phi) = 0$?
- W.l.o.g. we encode formulas so that at length n there is also encoding of all smaller formulas (otherwise use a different circuit S_i where needed).
- Check $\forall\Phi$ of size $\leqslant n$ (here x_1 is the first variable in Φ):

$$S_n(\Phi) = S_n(\Phi|_{x_1:=0}) \vee S_n(\Phi|_{x_1:=1})$$

- Add $S_n(\texttt{True}) = 1$, $S_n(\texttt{False}) = 0$ for formulas without variables as the end of this induction.
- Summary: S_n solves SAT correctly for size-n formulas iff

$$\forall\Phi \text{ of size } \leqslant n \ \underbrace{S_n(\Phi) = (S_n(\Phi|_{x_1:=0}) \vee S_n(\Phi|_{x_1:=1})) \wedge S_n(\texttt{True}) = 1 \wedge S_n(\texttt{False}) = 0}.$$

- Let us call it $\mathrm{VERIFY}(S_n, \Phi)$ and use it to show the collapse.

Current goal: use circuits for SAT to get rid of one quantifier

$\exists u \forall v \exists w \ F(u, v, w)$ is equivalent to

$$\exists S_n \exists u \qquad \forall v \forall \Phi \qquad \mathrm{VERIFY}(S_n, \Phi) \ \wedge \ S_n(\ \underbrace{F|_{u,v}}_{\text{variables } w}\ ) = 1.$$

The Karp–Lipton Theorem: summary

We have just proved

Theorem (R. Karp, R. Lipton, M. Sipser)

If $\mathbf{NP} \subseteq \mathbf{P/poly}$, then $\mathbf{PH} \subseteq \Sigma_2^p$.

The Karp–Lipton Theorem: summary

We have just proved

Theorem (R. Karp, R. Lipton, M. Sipser)

If $\mathbf{NP} \subseteq \mathbf{P/poly}$, then $\mathbf{PH} \subseteq \Sigma_2^p \cap \boldsymbol{\Pi_2^p}$.

($\cap\, \Pi_2^p$ is due to the symmetry of $\mathbf{PH}$: $\mathbf{PH} = \mathbf{co}\text{-}\mathbf{PH}$)

There are also much deeper collapses!

Circuit lower bounds

- Circuit lower bounds are a "holy grail" of complexity and TCS.
  (Think about **NP** $\not\subseteq$ **P** or **NP** $\not\subseteq$ **P/poly**.)
- There are no functions without circuits, but there exist functions without small circuits!

Theorem

There is $f : \{0,1\}^* \to \{0,1\}$ such that $\nexists \{C_n\}_{n\in\mathbb{N}}$ of polynomial size in n computing f.

Proof.

- Split f into "slices" $f_n : \{0,1\}^n \to \{0,1\}$.
- For a specific n, there are 2^{2^n} different functions $g : \{0,1\}^n \to \{0,1\}$.
- For a specific polynomial p, there are at most $2^{O(p(n)\log n)}$ circuits of size $\leqslant p(n)$.
- Asymptotically $2^{O(p(n)\log n)} < 2^{2^n}$, so for most n there is g without such circuits, let $f_n := g$.

Remark

A $p(n)$-size circuit with binary operations can be described by $O(p(n)\log n)$ bits. — Exercise.

However, even if you describe it using $O(p(n)^2)$ bits (inefficiently: using the adjacency matrix of the directed graph), the numbers in the proof of the next theorem can be changed accordingly and it will work.

Circuit lower bounds

- Circuit lower bounds are a "holy grail" of complexity and TCS.
  (Think about **NP** $\not\subseteq$ **P** or **NP** $\not\subseteq$ **P/poly**.)
- There are no functions without circuits, but there exist functions without small circuits!

Theorem

There is $f : \{0,1\}^* \to \{0,1\}$ such that $\nexists \{C_n\}_{n\in\mathbb{N}}$ of polynomial size in n computing f.

Proof.

- Split f into "slices" $f_n : \{0,1\}^n \to \{0,1\}$.
- For a specific n, there are 2^{2^n} different functions $g : \{0,1\}^n \to \{0,1\}$.
- For a specific polynomial p, there are at most $2^{O(p(n)\log n)}$ circuits of size $\leqslant p(n)$.
- Asymptotically $2^{O(p(n)\log n)} < 2^{2^n}$, so for most n there is g without such circuits, let $f_n := g$.

□

- Good, then give me this function!
- We are interested in f computed in **NP**.
- Currently no one can prove it exists, but we can prove something close.

Fixed-polynomial circuit lower bounds

1. We will not prove an exponential lower bound:
   just arbitrarily large polynomial.

2. We will not prove it for **NP** (nobody can):
   only for a somewhat larger class
   (Σ_4^p, then Σ_2^p, further improvements are possible).

Fixed-polynomial circuit lower bounds: Kannan's theorem

Theorem (Kannan)

$\forall k \quad \Sigma_4^p \not\subseteq \mathbf{Size}[n^k]$.

Proof.

- For a specific n, there are 2^{2^n} different functions $g : \{0,1\}^n \to \{0,1\}$.
- Look at the first $(k+1)\log n$ variables: there are $2^{2^{(k+\mathbf{1})\log n}} = 2^{n^{k+\mathbf{1}}}$ functions depending on them and just $2^{O(n^k \log n)}$ circuits of size $O(n^k)$.
- We need to find (and compute) just one! With 4 quantifiers...

Fixed-polynomial circuit lower bounds: Kannan's theorem

Theorem (Kannan)

$\forall k \quad \Sigma_4^p \not\subseteq \mathbf{Size}[n^k]$.

Proof.

- For a specific n, there are 2^{2^n} different functions $g : \{0,1\}^n \to \{0,1\}$.
- Look at the first $(k+1)\log n$ variables: there are $2^{2^{(k+1)\log n}} = 2^{n^{k+1}}$ functions depending on them and just $2^{O(n^k \log n)}$ circuits of size $O(n^k)$.
- We need to find (and compute) just one! With 4 quantifiers...
- Here is "one"... but there are many...

$\exists g$ depending on $x_1, \ldots, x_{(k+1)\log n}$ //Truth table of poly size

$\forall C_n$ of size $\leqslant n^k$
$\exists x \in \{0,1\}^n$

$(g(x_1, \ldots, x_n) \neq C_n(x_1, \ldots, x_n)$

□

Fixed-polynomial circuit lower bounds: Kannan's theorem

Theorem (Kannan)

$\forall k \quad \Sigma_4^p \not\subseteq \mathbf{Size}[n^k]$.

Proof.

- For a specific n, there are 2^{2^n} different functions $g : \{0,1\}^n \to \{0,1\}$.
- Look at the first $(k+1)\log n$ variables: there are $2^{2^{(k+1)\log n}} = 2^{n^{k+1}}$ functions depending on them and just $2^{O(n^k \log n)}$ circuits of size $O(n^k)$.
- We need to find (and compute) just one! With 4 quantifiers...
- Here is the "smallest" one, now what is its value?

$$
\begin{aligned}
&\exists g \text{ depending on } x_1, \ldots, x_{(k+1)\log n} && \text{//Truth table of poly size}\\
&\forall g' < g && \text{//lexicographically}\\
&\forall C_n \text{ of size} \leqslant n^k\\
&\exists x \in \{0,1\}^n\\
&\exists C'_n \text{ of size} \leqslant n^k\\
&\forall x' \in \{0,1\}^n && (g(x_1, \ldots, x_n) \neq C_n(x_1, \ldots, x_n)\\
& && \wedge\ g'(x'_1, \ldots, x'_n) = C'_n(x'_1, \ldots, x'_n)
\end{aligned}
$$

□

Fixed-polynomial circuit lower bounds: Kannan's theorem

Theorem (Kannan)

$\forall k \quad \Sigma_4^p \not\subseteq \mathbf{Size}[n^k]$.

Proof.

- For a specific n, there are 2^{2^n} different functions $g : \{0,1\}^n \to \{0,1\}$.
- Look at the first $(k+1)\log n$ variables: there are $2^{2^{(k+1)\log n}} = 2^{n^{k+1}}$ functions depending on them and just $2^{O(n^k \log n)}$ circuits of size $O(n^k)$.
- We need to find (and compute) just one! With 4 quantifiers...
- On input z of size n, compute $h(z)$ as...

$$
\begin{aligned}
&\exists g \text{ depending on } x_1, \ldots, x_{(k+1)\log n} && \text{//Truth table of poly size}\\
&\forall g' < g && \text{//lexicographically}\\
&\forall C_n \text{ of size} \leqslant n^k\\
&\exists x \in \{0,1\}^n\\
&\exists C'_n \text{ of size} \leqslant n^k\\
&\forall x' \in \{0,1\}^n && (g(x_1, \ldots, x_n) \neq C_n(x_1, \ldots, x_n)\\
& && \quad \wedge g'(x'_1, \ldots, x'_n) = C'_n(x'_1, \ldots, x'_n) \wedge g(z) = 1)
\end{aligned}
$$

□

Fixed-polynomial circuit lower bounds: Kannan's theorem

Theorem (Kannan)

$\forall k \quad \Sigma_4^p \not\subseteq \mathbf{Size}[n^k]$.

NB!!! It does not mean $\Sigma_4^p \not\subseteq \mathbf{P/poly}$.

Proof.

- For a specific n, there are 2^{2^n} different functions $g : \{0,1\}^n \to \{0,1\}$.
- Look at the first $(k+1)\log n$ variables: there are $2^{2^{(k+1)\log n}} = 2^{n^{k+1}}$ functions depending on them and just $2^{O(n^k \log n)}$ circuits of size $O(n^k)$.
- We need to find (and compute) just one! With 4 quantifiers...
- On input z of size n, compute $h(z)$ as...

$$\begin{aligned}
&\exists g \text{ depending on } x_1, \ldots, x_{(k+1)\log n} && \text{//Truth table of poly size}\\
&\forall g' < g && \text{//lexicographically}\\
&\forall C_n \text{ of size} \leqslant n^k\\
&\exists x \in \{0,1\}^n\\
&\exists C'_n \text{ of size} \leqslant n^k\\
&\forall x' \in \{0,1\}^n \qquad (g(x_1, \ldots, x_n) \neq C_n(x_1, \ldots, x_n)\\
&\qquad\qquad \wedge\, g'(x'_1, \ldots, x'_n) = C'_n(x'_1, \ldots, x'_n) \wedge g(z) = 1)
\end{aligned}$$

□

Now with just two quantifiers

Corollary

$\forall k \in \mathbb{N} \quad \Sigma_2^p \not\subseteq \mathbf{Size}[n^k]$.

Proof.

- Assume the contrary: ($\mathbf{NP} \subseteq$) $\Sigma_2^p \subseteq \mathbf{Size}[n^k]$ ($\subseteq \mathbf{P/poly}$).
- The Karp–Lipton Theorem $\Longrightarrow$ collapse $\mathbf{PH} = \Sigma_2^p$.
- But then $\Sigma_2^p = \Sigma_4^p \not\subseteq \mathbf{Size}[n^k]$ by Kannan's Theorem.

□

Now with just two quantifiers

Corollary

$\forall k \in \mathbb{N} \quad \Sigma_2^p \not\subseteq \mathbf{Size}[n^k]$.

Proof.

- Assume the contrary: ($\mathbf{NP} \subseteq$) $\Sigma_2^p \subseteq \mathbf{Size}[n^k]$ ($\subseteq \mathbf{P/poly}$).
- The Karp–Lipton Theorem $\Longrightarrow$ collapse $\mathbf{PH} = \Sigma_2^p$.
- But then $\Sigma_2^p = \Sigma_4^p \not\subseteq \mathbf{Size}[n^k]$ by Kannan's Theorem.

□

- Thus a better KL theorem would give a better corollary.
- But only if the collapse is to a class containing **NP**.
- There are other techniques for smaller classes even not containing **NP**.

Now with just two quantifiers

Corollary

$\forall k \in \mathbb{N} \quad \Sigma_2^p \not\subseteq \mathbf{Size}[n^k]$.

Proof.

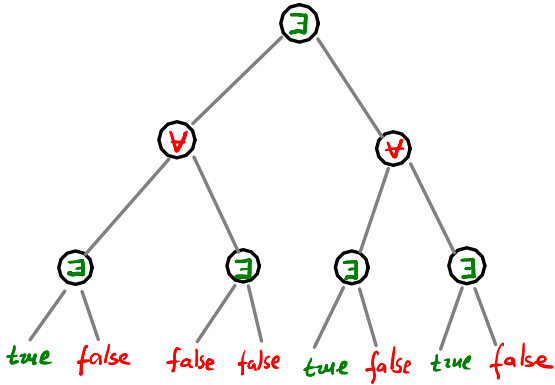
- Assume the contrary: $(\mathbf{NP} \subseteq)\ \ \Sigma_2^p \subseteq \mathbf{Size}[n^k]\ \ (\subseteq \mathbf{P/poly})$.
- The Karp–Lipton Theorem $\Longrightarrow$ collapse $\mathbf{PH} = \Sigma_2^p$.
- But then $\Sigma_2^p = \Sigma_4^p \not\subseteq \mathbf{Size}[n^k]$ by Kannan's Theorem.

□

- Thus a better KL theorem would give a better corollary.
- But only if the collapse is to a class containing **NP**.
- There are other techniques for smaller classes even not containing **NP**.

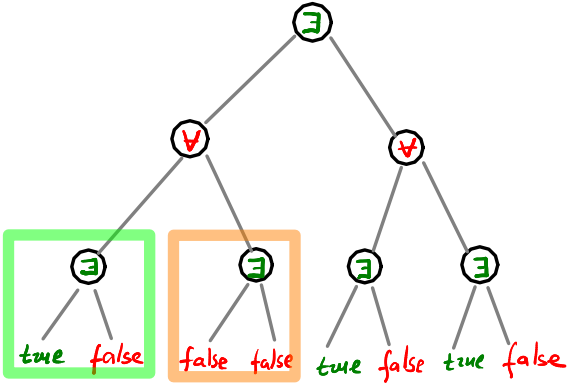
Space Complexity

$\exists\forall\exists\forall\ldots$ — where all this goes?



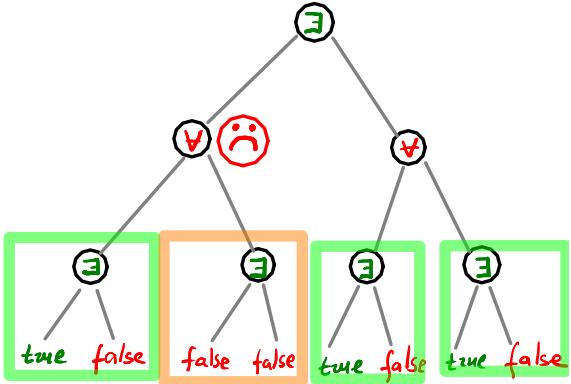
Space Complexity

$\exists\forall\exists\forall\ldots$ — where all this goes?



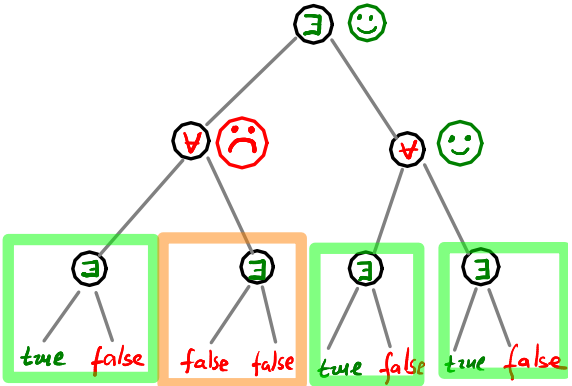
Space Complexity

$\exists\forall\exists\forall\ldots$ — where all this goes?

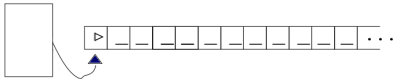


Space Complexity

$\exists\forall\exists\forall\ldots$ — where all this goes?



Space complexity classes



- Time is the number of steps before a Turing machine stops.
- Space is the number of cells it uses, that is, the sum of the rightmost positions of its heads.
- $$\mathbf{DSpace}[f(n)] = \{L \mid L \text{ is decided by a DTM using space } O(f(n))\}.$$

  The input tape must be separated and made readonly, but for polynomial space it does not matter.

$$\mathbf{PSPACE} = \bigcup_{k \geq 0} \mathbf{DSpace}[n^k].$$

Soon we'll see it corresponds to two-player games like chess.

- For a NTM, it is the maximum space used on all nondeterministic paths.

$$\mathbf{NSpace}[f(n)] = \{L \mid L \text{ is decided by a NTM using space } O(f(n))\}.$$

Examples (Games!)

Example (SPACE-BH, SPACE TMSAT in the book)

SPACE-BH $= \{\langle M, z, 1^t \rangle : \text{DTM } M \text{ accepts } z \text{ in space } t\}$.

Why **PSPACE**? If $M(x)$ uses space $s(|x|)$, then UTM$(\langle M, x \rangle)$ can use space $O(s(|x|))$

SPACE-BH is **PSPACE**-complete (Exercise!).

Examples (Games!)

Example (SPACE-BH, SPACE TMSAT in the book)

SPACE-BH $= \{\langle M, z, 1^t\rangle : \text{DTM } M \text{ accepts } z \text{ in space } t\}$.

Why **PSPACE**? If $M(x)$ uses space $s(|x|)$, then UTM$(\langle M, x\rangle)$ can use space $O(s(|x|))$

SPACE-BH is **PSPACE**-complete (Exercise!).

Example

A directed **graph** G, a distinguished **node** $v \in G$, **two players** A and B building a **path**.

A selects an edge $(v, \boldsymbol{u_1})$, then
B selects an edge $(\boldsymbol{u_1}, u_2)$, then
A selects an edge $(u_2, u_3)\ldots$ etc

It is prohibited to visit the same node twice: the player who has to do it, loses.

NB! It is not about Eulerian or Hamiltonian paths!

Problem: given G, v, tell us whether A wins.

Why **PSPACE**?

- At each step, the range of moves is limited.
- The length of this game is limited.
- Both "valid moves" and "winning position" are poly-time computable.

Exercise: complete the proof (after you see next slide).

Exercise (more difficult, but doable): prove it is **PSPACE**-complete.

QBF (aka QBF-SAT, aka TQBF): a **PSPACE**-complete problem

$$\texttt{QBF} = \{ \text{ true quantified CNF formulas } Q_1 x_1 \; Q_2 x_2 \ldots \; \Phi \; \},$$

where Φ in CNF (Exercise: it does not matter), $Q_i \in \{\forall, \exists\}$. All x_i's are single bits, no free variables!

QBF (aka QBF-SAT, aka TQBF): a **PSPACE**-complete problem

$$\text{QBF} = \{ \text{ true quantified CNF formulas } Q_1 x_1 \ Q_2 x_2 \ldots \ \Phi \ \},$$

where Φ in CNF (Exercise: it does not matter), $Q_i \in \{\forall, \exists\}$. All x_i's are single bits, no free variables!

Theorem

QBF is **PSPACE**-complete under polynomial-time many-one reductions.
(Exercise: **PSPACE** is closed under them.)

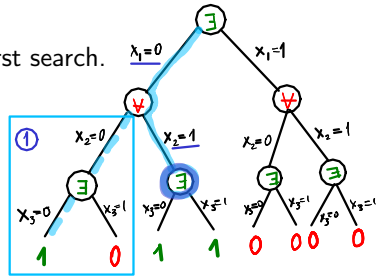
QBF $\in$ **PSPACE**: — in this lecture we prove only this part

Conside Φ's brute-force search tree (a leaf $0, 1, 0$ gets the value $\Phi(0.1.0)$).

Compute $Q_1 x_1 \ Q_2 x_2 \ \ldots Q_n x_n \ \Phi(x_1, x_2, \ldots, x_n)$, using depth-first search.

Space: the current path + two last values.

Corollary: $\Sigma_k, \Pi_k \subseteq$ **PH** $\subseteq$ **PSPACE**.



QBF (aka QBF-SAT, aka TQBF): a **PSPACE**-complete problem

$$\texttt{QBF} = \{ \text{ true quantified CNF formulas } Q_1 x_1 \; Q_2 x_2 \ldots \; \Phi \; \},$$

where Φ in CNF (Exercise: it does not matter), $Q_i \in \{\forall, \exists\}$. All x_i's are single bits, no free variables!

Theorem

QBF is **PSPACE**-complete under polynomial-time many-one reductions.
(Exercise: **PSPACE** is closed under them.)

QBF is **PSPACE**-hard:
to prove it, we consider an arbitrary $L \in$ **PSPACE** and
build a reduction from $L \in$ **PSPACE** to QBF. $\longrightarrow$ Next lecture