אוניברסיטת
אריאל
בשומרון

Computational Complexity

Lecturer: Edward A. Hirsch

https://edwardahirsch.github.io/edwardahirsch

edward.a.hirsch@gmail.com

December 10, 2025

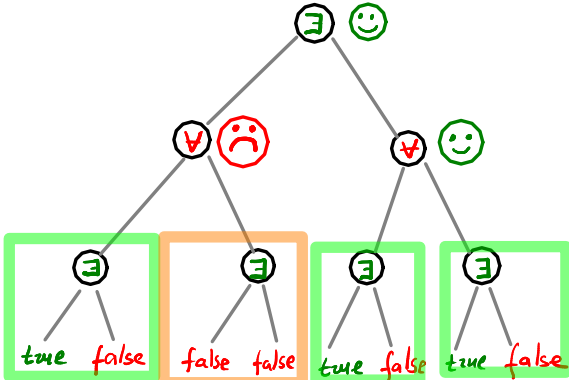
Lecture plan

- Space-bounded computation.
  - `QBF`: a **PSPACE**-complete problem.
  - Logspace-bounded computation: **L** and **NL**.
  - Logspace reductions, **P**-completeness.
  - `CIRCUIT_EVAL`: a **P**-complete problem.

Space Complexity

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

(**polynomial space** complexity — continued)



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

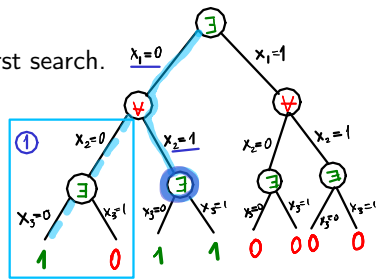
QBF is **PSPACE**-complete under polynomial-time many-one reductions.
(Exercise: **PSPACE** is closed under them.)

QBF $\in$ **PSPACE**:
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

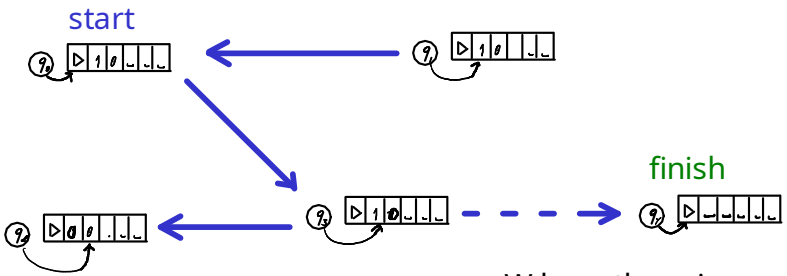
$$\texttt{QBF} = \{ \text{ true quantified CNF formulas } Q_1 x_1 \; Q_2 x_2 \ldots \Phi \; \},$$

Theorem

QBF is **PSPACE**-complete under polynomial-time many-one reductions.

Reduce $L \in$ **PSPACE** to QBF.

$p(n)$ is not the space bound, but almost



W.l.o.g. there is a single accepting configuration

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

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

Theorem

QBF is **PSPACE**-complete under polynomial-time many-one reductions.

Reduce $L \in$ **PSPACE** to QBF.

$p(n)$ is not the space bound, but almost

Reachability in the configuration graph ($2^{p(n)}$ configurations) of a Turing machine M accepting L.

Build $\Phi_i(c_1, c_2) =$ «there is a path $c_1 \rightsquigarrow c_2$ of length $\leqslant 2^i$».

$$\Phi_i(c_1, c_2) \;=\; \exists \boldsymbol{d} \;\; \Phi_{i-1}(c_1, \boldsymbol{d}) \wedge \Phi_{i-1}(\boldsymbol{d}, c_2).$$

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

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

Theorem

QBF is **PSPACE**-complete under polynomial-time many-one reductions.

Reduce $L \in$ **PSPACE** to QBF.

$p(n)$ is not the space bound, but almost

Reachability in the configuration graph ($2^{p(n)}$ configurations) of a Turing machine M accepting L.

Build $\Phi_i(c_1, c_2) =$ «there is a path $c_1 \rightsquigarrow c_2$ of length $\leqslant 2^i$».

~~$\Phi_i(c_1, c_2) = \exists d \; \Phi_{i-1}(c_1, d) \wedge \Phi_{i-1}(d, c_2).$~~

$$\boldsymbol{\Phi_i}(c_1, c_2) = \exists \boldsymbol{d} \forall x \forall y (((x = c_1 \wedge y = \boldsymbol{d}) \vee (x = \boldsymbol{d} \wedge y = c_2)) \Rightarrow \boldsymbol{\Phi_{i-1}}(x, y)).$$

($\Phi_0(c_1, c_2)$ is like in the Cook–Levin theorem: one step of TM M)

? $\Phi_{p(n)}(c_{start}, c_{accept})$ — built by induction

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

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

Theorem

QBF is **PSPACE**-complete under polynomial-time many-one reductions.

Reduce $L \in$ **PSPACE** to QBF.

$p(n)$ is not the space bound, but almost

Reachability in the configuration graph ($2^{p(n)}$ configurations) of a Turing machine M accepting L.

Build $\Phi_i(c_1, c_2) =$ «there is a path $c_1 \rightsquigarrow c_2$ of length $\leqslant 2^i$».

~~$\Phi_i(c_1, c_2) = \exists d \; \Phi_{i-1}(c_1, d) \wedge \Phi_{i-1}(d, c_2).$~~

$$\boldsymbol{\Phi_i(c_1, c_2) = \exists d \forall x \forall y (((x = c_1 \wedge y = d) \vee (x = d \wedge y = c_2)) \Rightarrow \Phi_{i-1}(x, y)).}$$

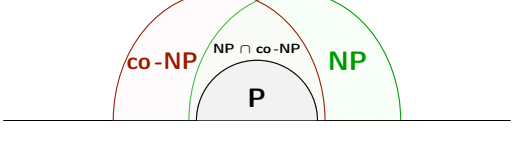
($\Phi_0(c_1, c_2)$ is like in the Cook–Levin theorem: one step of TM M)

? $\Phi_{p(n)}(c_{start}, c_{accept})$ — built by induction

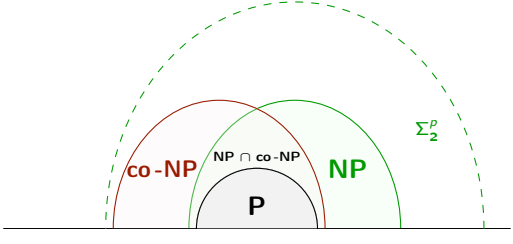
We did not use that the machine is deterministic!

Corollary: **PSPACE** = **NPSPACE**, and unlike **NP**, the class **NPSPACE** = **co-NPSPACE**.

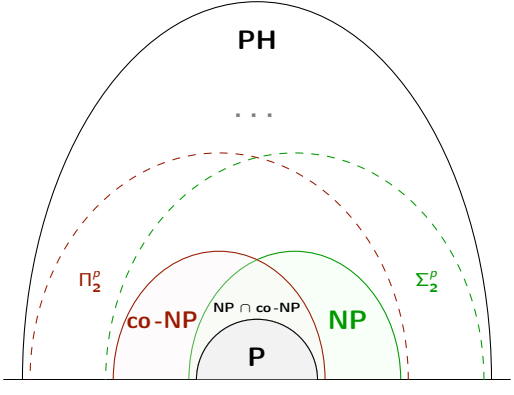
Back to the general picture



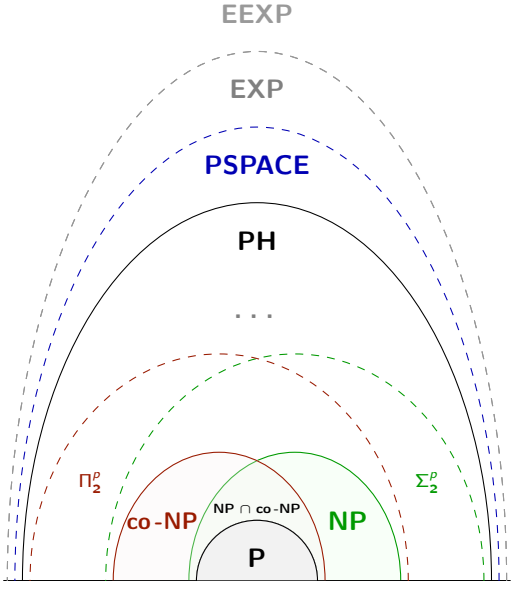
Back to the general picture



Back to the general picture

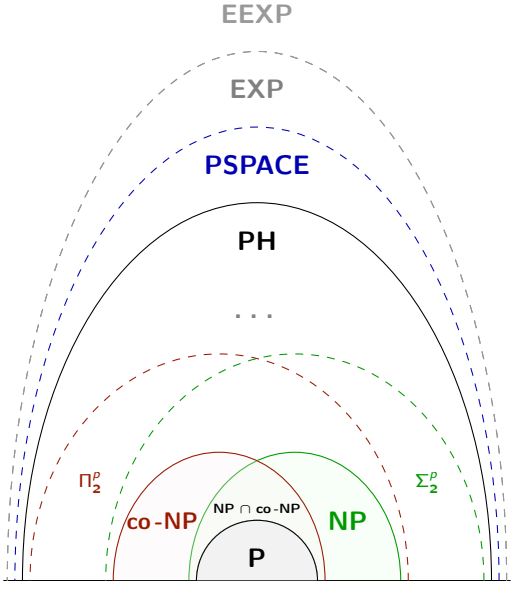


Back to the general picture



Can **PH** = **PSPACE**?

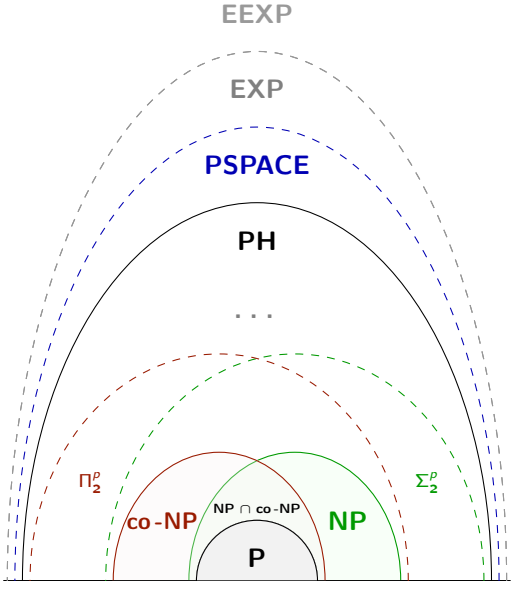
Back to the general picture



Can **PH** = **PSPACE**?

Then **PH** would collapse,
because **PSPACE** has complete problems.

Back to the general picture

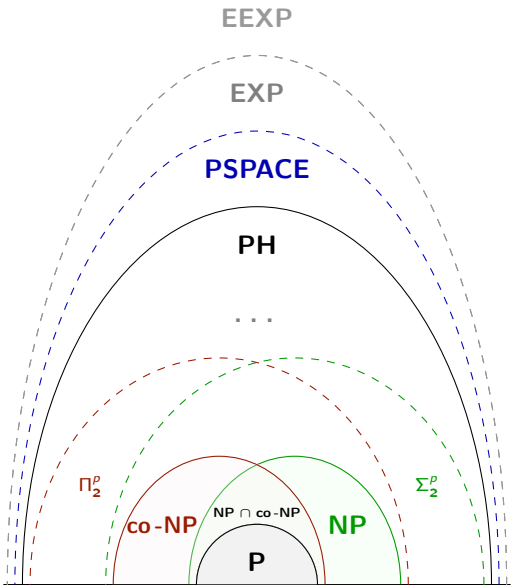


Can **PH** = **PSPACE**?

Then **PH** would collapse,
because **PSPACE** has complete problems.

Why **PSPACE** ⊆ **EXP**?

Back to the general picture



Can **PH** = **PSPACE**?

Then **PH** would collapse,
because **PSPACE** has complete problems.

Why **PSPACE** ⊆ **EXP**?

Because there are just
exponentially many configurations.

(We can stop the machine when it comes back.)

How time is related to space?

More accurately than "up to a polynomial"

- $\mathbf{DSpace}[s(n)] \subseteq \mathbf{DTime}[const^{s(n)}]$ (due to the number of configurations) for $s(n) \geqslant \log n$.

- $\mathbf{DTime}[t(n)] \subseteq \mathbf{DSpace}[t(n)]$ (trivial).

How time is related to space?

More accurately than "up to a polynomial"

- $\mathbf{DSpace}[s(n)] \subseteq \mathbf{DTime}[const^{s(n)}]$ (due to the number of configurations) for $s(n) \geqslant \log n$.

- $\mathbf{DTime}[t(n)] \subseteq \mathbf{DSpace}[t(n)]$ (trivial).
- $\mathbf{DTime}[t(n)] \subseteq \mathbf{DSpace}[\sqrt{t(n) \log n}]$ (Ryan Williams, 2025) for $t(n) \geqslant n$

More to learn about Space Complexity

Small-space classes
Logarithmic space, nondeterministic log. space

Parallel computation
How is it related to logarithmic space?

The space hierarchy theorem
More space $\Rightarrow$ more languages

Small space classes: deterministic, nondeterministic, and. . . parallel!

$$\mathbf{L} = \mathbf{DL} = \mathbf{DSpace}[\log n]$$

$$\mathbf{NL} = \mathbf{NSpace}[\log n]$$

$$\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P}$$

$$\mathbf{L} \stackrel{?}{=} \mathbf{NL} \stackrel{?}{=} \mathbf{P}$$

Small space classes: deterministic, nondeterministic, and. . . parallel!

$$\mathbf{L} = \mathbf{DL} = \mathbf{DSpace}[\log n]$$

$$\mathbf{NL} = \mathbf{NSpace}[\log n]$$

$\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P}$

$\mathbf{L} \stackrel{?}{=} \mathbf{NL} \stackrel{?}{=} \mathbf{P}$

Stop!!! How come?! Does it read the input?

Definition

DSpace$[f(n)] = \{L \mid L$ is decided by a DTM using space $O(f(n))\}$,
where the DTM has a dedicated read-only input tape.

NSpace$[f(n)] = \{L \mid L$ is decided by a NTM using space $O(f(n))\}$,
where the NTM has a dedicated read-only input tape.

Important! If you think about **NSpace** computations as a DTM with a witness, the witness tape must be read in a readonly unidirectional mode (otherwise the head position counts).

Small space classes: deterministic, nondeterministic, and. . . parallel!

$$\mathbf{L} = \mathbf{DL} = \mathbf{DSpace}[\log n] \qquad \mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P}$$

$$\mathbf{NL} = \mathbf{NSpace}[\log n] \qquad \mathbf{L} \stackrel{?}{=} \mathbf{NL} \stackrel{?}{=} \mathbf{P}$$

Stop!!! How come?! Does it read the input?

Definition

$\mathbf{DSpace}[f(n)] = \{L \mid L \text{ is decided by a DTM using space } O(f(n))\}$,
where the DTM has a dedicated read-only input tape.

$\mathbf{NSpace}[f(n)] = \{L \mid L \text{ is decided by a NTM using space } O(f(n))\}$,
where the NTM has a dedicated read-only input tape.

Important! If you think about **NSpace** computations as a DTM with a witness, the witness tape must be read in a readonly unidirectional mode (otherwise the head position counts).

Definition

S is **P**-complete if every $W \in \mathbf{P}$ many-one reduces to S using a logspace reduction.

A logspace many-one reduction is computed by a polynomial-time DTM that uses log space on its working tapes, its input tape is read-only and its output tape is write-only (once) and unidirectional.

Configurations in the context of low space complexity

Consider a NTM using space $s(n) \geqslant \log n$. Let us count configurations ($n =$ input size)!

A configuration is something that determines the future of the machine, thus

- The state (from the state of the finite states).
- The contents of the working tapes.
- The current position of all the heads on the working tapes.
- The position of the head on the input tape.

Configurations in the context of low space complexity

Consider a NTM using space $s(n) \geqslant \log n$. Let us count configurations ($n =$ input size)!

A configuration is something that determines the future of the machine, thus

- The state (from the state of the finite states). $O(1)$ bits
- The contents of the working tapes.
- The current position of all the heads on the working tapes.
- The position of the head on the input tape.

Configurations in the context of low space complexity

Consider a NTM using space $s(n) \geqslant \log n$. Let us count configurations ($n =$ input size)!

A configuration is something that determines the future of the machine, thus

- The state (from the state of the finite states). $O(1)$ bits
- The contents of the working tapes. $O(s(n))$ bits
- The current position of all the heads on the working tapes.
- The position of the head on the input tape.

Configurations in the context of low space complexity

Consider a NTM using space $s(n) \geqslant \log n$. Let us count configurations ($n =$ input size)!

A configuration is something that determines the future of the machine, thus

- The state (from the state of the finite states). $O(1)$ bits
- The contents of the working tapes. $O(s(n))$ bits
- The current position of all the heads on the working tapes. $O(\log s(n))$ bits
- The position of the head on the input tape.

Configurations in the context of low space complexity

Consider a NTM using space $s(n) \geqslant \log n$. Let us count configurations ($n =$ input size)!

A configuration is something that determines the future of the machine, thus

- The state (from the state of the finite states). $O(1)$ bits
- The contents of the working tapes. $O(s(n))$ bits
- The current position of all the heads on the working tapes. $O(\log s(n))$ bits
- The position of the head on the input tape. $O(\log n)$ bits

Total:

$$2^{O(s(n)+\log s(n)+\log n)} \text{ i.e., } \mathrm{const}^{s(n)}$$

configurations.

Remark

Machines using less than $\log n$ space are a bizarre phenomenon. In fact,

$$\mathbf{DSpace}[\log\log n] \neq \mathbf{DSpace}[(\log\log n)^{1-\varepsilon}] = \mathbf{DSpace}[O(1)] = \mathbf{REG}.$$

We won't prove it.

Exercises on space complexity

Exercise

Show that a logspace reduction can always be made polynomial-time.

Exercise

Arora-Barak's book defines implicit logspace reductions instead:

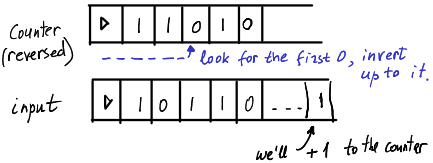
Definition 4.16 A function $f\colon \{0,1\} \to \{0,1\}$ is implicitly logspace computable, if

- f is polynomially bounded (i.e., $|f(x)| \leqslant \mathrm{poly}(|x|)$), and
- the languages $L_f = \{\langle x, i\rangle : f(x)_i = 1\}$ and $L'_f = \{\langle x, i\rangle \colon i \leqslant |f(x)|\}$ are in L.

Prove that there is a logspace reduction between two languages W and S $\iff$ there is a reduction between them that is implicitly logspace computable.

Examples of logspace-computable problems

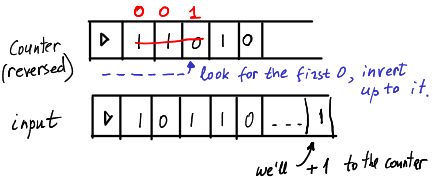
Counting the input length...



...and any symmetric function (depending on the number of 1's only).

Examples of logspace-computable problems

Counting the input length...

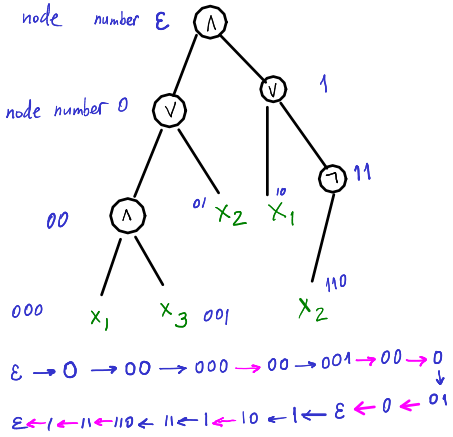


...and any symmetric function (depending on the number of 1's only).

Examples of logspace-computable problems

$$\texttt{BALANCED\_FORMULA\_EVAL} = \{\langle \text{balanced formula } \Phi, \text{input } x\rangle : \Phi(x) = 1\}.$$

- Balanced = log depth.
- A formula is a tree, it can be evaluated by DFS in space proportional to its depth.



In fact, one can do it for unbalanced formulas as well (Sam Buss, ca 1987).

More on **P**-completeness

Theorem

1. Logspace reductions are transitive.
2. **L** and **NL** are closed under logspace reductions.
3. In particular, if C is **P**-complete, then $C \in \mathbf{L} \iff \mathbf{P} = \mathbf{L}$.

A standard **P**-complete problem:

$$\texttt{CIRCUIT\_EVAL} = \{\langle \text{circuit } C, \text{input } x\rangle : C(x) = 1\}.$$

More on **P**-completeness

Theorem

1. Logspace reductions are transitive.
2. **L** and **NL** are closed under logspace reductions.
3. In particular, if C is **P**-complete, then $C \in \mathbf{L} \iff \mathbf{P} = \mathbf{L}$.

A standard **P**-complete problem:

$$\texttt{CIRCUIT\_EVAL} = \{\langle \text{circuit } C, \text{input } x\rangle : C(x) = 1\}.$$

`CIRCUIT_EVAL` $\in$ **P** is obvious.

A logspace reduction of a DTM simulation by a circuit: see the `Circuit-SAT`-completess proof. (Verify that it is logspace! Essentially, two "for" cycles with counters.)

More on **P**-completeness

Theorem

1. Logspace reductions are transitive.
2. **L** and **NL** are closed under logspace reductions.
3. In particular, if C is **P**-complete, then $C \in \mathbf{L} \iff \mathbf{P} = \mathbf{L}$.

A standard **P**-complete problem:

$$\texttt{CIRCUIT\_EVAL} = \{\langle \text{circuit } C, \text{input } x\rangle : C(x) = 1\}.$$

`CIRCUIT_EVAL` $\in$ **P** is obvious.

A logspace reduction of a DTM simulation by a circuit: see the `Circuit-SAT`-completess proof.
(Verify that it is logspace! Essentially, two "for" cycles with counters.)

It remains to prove 1 (transitivity), 2 is analogous: Compose logspace reductions f, g into $f \circ g$.

Logspace Algorithm for computing $f(g(x))$:
— Start computing f,
— For every bit of f's input simulate $g(x)$ from scratch!
(We don't care about the time.)