## Boolean Logic[a]

**Boolean variables:** $x_1, x_2, \ldots$.

**Literals:** $x_i$, $\neg x_i$.

**Boolean connectives:** $\vee, \wedge, \neg$.

**Boolean expressions:** Boolean variables, $\neg\phi$ (**negation**), $\phi_1 \vee \phi_2$ (**disjunction**), $\phi_1 \wedge \phi_2$ (**conjunction**).

- $\bigvee_{i=1}^{n} \phi_i$ stands for $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_n$.
- $\bigwedge_{i=1}^{n} \phi_i$ stands for $\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$.

**Implications:** $\phi_1 \Rightarrow \phi_2$ is a shorthand for $\neg\phi_1 \vee \phi_2$.

**Biconditionals:** $\phi_1 \Leftrightarrow \phi_2$ is a shorthand for $(\phi_1 \Rightarrow \phi_2) \wedge (\phi_2 \Rightarrow \phi_1)$.

[a] Boole (1815–1864) in 1847.



## Truth Assignments

- A **truth assignment** $T$ is a mapping from boolean variables to **truth values** `true` and `false`.
- A truth assignment is **appropriate** to boolean expression $\phi$ if it defines the truth value for every variable in $\phi$.
  - $\{x_1 = \texttt{true}, x_2 = \texttt{false}\}$ is appropriate to $x_1 \vee x_2$.



## Satisfaction

- $T \models \phi$ means boolean expression $\phi$ is true under $T$; in other words, $T$ **satisfies** $\phi$.
- $\phi_1$ and $\phi_2$ are **equivalent**, written

$$\phi_1 \equiv \phi_2,$$

  if for any truth assignment $T$ appropriate to both of them, $T \models \phi_1$ if and only if $T \models \phi_2$.
  - Equivalently, $T \models (\phi_1 \Leftrightarrow \phi_2)$.



## Truth Tables

- Suppose $\phi$ has $n$ boolean variables.
- A **truth table** contains $2^n$ rows, one for each possible truth assignment of the $n$ variables together with the truth value of $\phi$ under that truth assignment.
- A truth table can be used to prove if two boolean expressions are equivalent.
  - Check if they give identical truth values under all $2^n$ truth assignments.

## A Truth Table

| $p$ | $q$ | $p \land q$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |




## De Morgan's[a] Laws

- **De Morgan's laws** say that

$$\neg(\phi_1 \land \phi_2) = \neg\phi_1 \lor \neg\phi_2,$$
$$\neg(\phi_1 \lor \phi_2) = \neg\phi_1 \land \neg\phi_2.$$

- Here is a proof for the first law:

| $\phi_1$ | $\phi_2$ | $\neg(\phi_1 \land \phi_2)$ | $\neg\phi_1 \lor \neg\phi_2$ |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

[a] Augustus DeMorgan (1806–1871).




## Conjunctive Normal Forms

- A boolean expression $\phi$ is in **conjunctive normal form** (**CNF**) if

$$\phi = \bigwedge_{i=1}^{n} C_i,$$

where each **clause** $C_i$ is the disjunction of one or more literals.

- For example,

$$(x_1 \lor x_2) \land (x_1 \lor \neg x_2) \land (x_2 \lor x_3).$$

is in CNF.




## Disjunctive Normal Forms

- A boolean expression $\phi$ is in **disjunctive normal form** (**DNF**) if

$$\phi = \bigvee_{i=1}^{n} D_i,$$

where each **implicant** $D_i$ is the conjunction of one or more literals.

- For example,

$$(x_1 \land x_2) \lor (x_1 \land \neg x_2) \lor (x_2 \land x_3).$$

is in DNF.

## Any Expression $\phi$ Can Be Converted into CNFs and DNFs

$\phi = x_j$**:** This is trivially true.

$\phi = \neg\phi_1$ **and a CNF is sought:** Turn $\phi_1$ into a DNF and apply de Morgan's laws to make a CNF for $\phi$.

$\phi = \neg\phi_1$ **and a DNF is sought:** Turn $\phi_1$ into a CNF and apply de Morgan's laws to make a DNF for $\phi$.

$\phi = \phi_1 \vee \phi_2$ **and a DNF is sought:** Make $\phi_1$ and $\phi_2$ DNFs.

$\phi = \phi_1 \vee \phi_2$ **and a CNF is sought:** Let $\phi_1 = \bigwedge_{i=1}^{n_1} A_i$ and $\phi_2 = \bigwedge_{i=1}^{n_2} B_i$ be CNFs. Set $\phi = \bigwedge_{i=1}^{n_1} \bigwedge_{j=1}^{n_2} (A_i \vee B_j)$.

$\phi = \phi_1 \wedge \phi_2$**:** Similar to above.




## Satisfiability

- A boolean expression $\phi$ is **satisfiable** if there is a truth assignment $T$ appropriate to it such that $T \models \phi$.
- $\phi$ is **valid** or a **tautology**,[a] written $\models \phi$, if $T \models \phi$ for all $T$ appropriate to $\phi$.
- $\phi$ is **unsatisfiable** if and only if $\phi$ is false under all appropriate truth assignments if and only if $\neg\phi$ is valid.

[a] Wittgenstein (1889–1951) in 1922. Wittgenstein is one of the most important philosophers of all time. "God has arrived," Keynes said of him on January 18, 1928. "I met him on the 5:15 train."




## SATISFIABILITY (SAT)

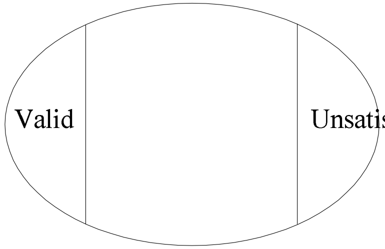
- The **length** of a boolean expression is the length of the string encoding it.
- SATISFIABILITY (SAT): Given a CNF $\phi$, is it satisfiable?
- Solvable in time $O(n^2 2^n)$ on a TM by the truth table method.
- Solvable in polynomial time on an NTM, hence in NP (p. 80).
- A most important problem in answering the P = NP problem (p. 242).




## UNSATISFIABILITY (UNSAT or SAT COMPLEMENT) and VALIDITY

- UNSAT (SAT COMPLEMENT): Given a boolean expression $\phi$, is it unsatisfiable?
- VALIDITY: Given a boolean expression $\phi$, is it valid?
  - $\phi$ is valid if and only if $\neg\phi$ is unsatisfiable.
  - So UNSAT and VALIDITY have the same complexity.
- Both are solvable in time $O(n^2 2^n)$ on a TM by the truth table method.

## Relations among SAT, UNSAT, and VALIDITY




- The negation of an unsatisfiable expression is a valid expression.
- None of the three problems—satisfiability, unsatisfiability, validity—are known to be in P.



## Horn Clauses

- A **Horn clause** is a clause with at most one *positive* literal.
  - $\neg x_2 \lor x_3$, $\neg x_1 \lor \neg x_2 \lor \neg x_3$.
- Let $\phi$ be a conjunction of Horn clauses.
  - So $\phi$ is a CNF.
- Satisfiability of $\phi$ is in P (see text).

## Boolean Functions

- An $n$-ary boolean function is a function
$$f : \{\texttt{true}, \texttt{false}\}^n \to \{\texttt{true}, \texttt{false}\}.$$
- It can be represented by a truth table.
- There are $2^{2^n}$ such boolean functions.
  - Each of the $2^n$ truth assignments can make $f$ true or false.



## Boolean Functions (concluded)

- A boolean expression expresses a boolean function.
  - Think of its truth value under all truth assignments.
- A boolean function expresses a boolean expression.
  - $\bigvee_{T \models \phi, \text{ literal } y_i \text{ is true under } T} (y_1 \wedge \cdots \wedge y_n)$.
    - $y_1 \wedge \cdots \wedge y_n$ is the **minterm** over $\{x_1, \ldots, x_n\}$ for $T$.
  - The boolean function on p. 134 produces $p \wedge q$.
  - The length[a] is $\leq n2^n \leq 2^{2n}$.
  - In general, the exponential length in $n$ cannot be avoided (p. 157)!

[a]We count the logical connectives here.



## Boolean Circuits

- A **boolean circuit** is a graph $C$ whose nodes are the **gates**.
- There are no cycles in $C$.
- All nodes have indegree (number of incoming edges) equal to 0, 1, or 2.
- Each gate has a **sort** from
$$\{\texttt{true}, \texttt{false}, \vee, \wedge, \neg, x_1, x_2, \ldots\}.$$

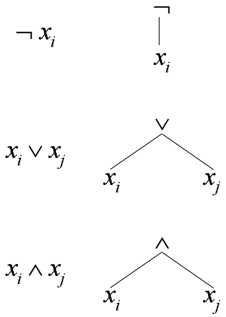


## Boolean Circuits (concluded)

- Gates of sort from $\{\texttt{true}, \texttt{false}, x_1, x_2, \ldots\}$ are the **inputs** of $C$ and have an indegree of zero.
- The **output gate**(s) has no outgoing edges.
- A boolean circuit computes a boolean function.
- The same boolean function can be computed by infinitely many boolean circuits.

## Boolean Circuits and Expressions

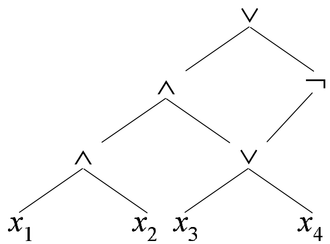
- They are equivalent representations.
- One can construct one from the other:






## An Example

$$((x_1 \wedge x_2) \wedge (x_3 \vee x_4)) \vee (\neg (x_3 \vee x_4))$$




- Circuits are more economical because of the possibility of sharing.



## CIRCUIT SAT and CIRCUIT VALUE

**CIRCUIT SAT:** Given a circuit, is there a truth assignment such that the circuit outputs true?

**CIRCUIT VALUE:** The same as CIRCUIT SAT except that the circuit has no variable gates.

- CIRCUIT SAT $\in$ NP: Guess a truth assignment and then evaluate the circuit.
- CIRCUIT VALUE $\in$ P: Evaluate the circuit from the input gates gradually towards the output gate.

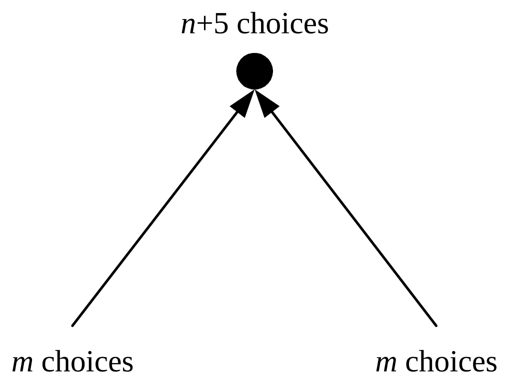


## Some Boolean Functions Need Exponential Circuits

**Theorem 16 (Shannon (1949))** *For any $n \geq 2$, there is an $n$-ary boolean function $f$ such that no boolean circuits with $2^n/(2n)$ or fewer gates can compute it.*[a]

- There are $2^{2^n}$ different $n$-ary boolean functions.
- There are at most $((n+5) \times m^2)^m$ boolean circuits with $m$ or fewer gates.
- But $((n+5) \times m^2)^m < 2^{2^n}$ when $m = 2^n/(2n)$.
  - $m \log_2((n+5) \times m^2) = 2^n (1 - \frac{\log_2 \frac{4n^2}{n+5}}{2n}) < 2^n$ for $n \geq 2$.

[a] Can be strengthened to "almost all boolean functions ..."



## Comments

- The lower bound is rather tight because an upper bound is $n2^n$ (p. 151).
- In the proof, we are counting the number of "physical" circuits.
- Some circuits may not be valid at all.
- Others may compute the same boolean functions.
- Both are fine because we only need an upper bound.
- We do not need to consider the outdoing edges because they have been counted in the incoming edges.

## Proper (Complexity) Functions

- We say that $f : \mathbb{N} \to \mathbb{N}$ is a **proper (complexity) function** if the following hold:
  - $f$ is nondecreasing.
  - There is a $k$-string TM $M_f$ such that $M_f(x) = \sqcap^{f(|x|)}$ for any $x$.[a]
  - $M_f$ halts after $O(|x| + f(|x|))$ steps.
  - $M_f$ uses $O(f(|x|))$ space besides its input $x$.
- $M_f$'s behavior depends only on $|x|$ not $x$'s contents.
- $M_f$'s running time is basically bounded by $f(n)$.

[a]This point will become clear in Proposition 17 on p. 164.

## Examples of Proper Functions

- Most "reasonable" functions are proper: $c$, $\lceil \log n \rceil$, polynomials of $n$, $2^n$, $\sqrt{n}$, $n!$, etc.
- If $f$ and $g$ are proper, then so are $f + g$, $fg$, and $2^g$.
- Nonproper functions when serving as the time bounds for complexity classes spoil "the theory building."
  - For example, $\text{TIME}(f(n)) = \text{TIME}(2^{f(n)})$ for some recursive function $f$ (the **gap theorem**).
- We shall henceforth use only proper functions in relation to complexity classes $\text{TIME}(f(n))$, $\text{SPACE}(f(n))$, $\text{NTIME}(f(n))$, and $\text{NSPACE}(f(n))$.

## Space-Bounded Computation and Proper Functions

- In the definition of space-bounded computations, the TMs are not required to halt at all.
- When the space is bounded by a proper function $f$, computations can be assumed to halt:
  - Run the TM associated with $f$ to produce an output of length $f(n)$ first.
  - The space-bound computation must repeat a configuration if it runs for more than $c^{n+f(n)}$ steps for some $c$ (p. 182).
  - So we can count steps to prevent infinite loops.



## Precise Turing Machines

- A TM $M$ is **precise** if there are functions $f$ and $g$ such that for every $n \in \mathbb{N}$, for every $x$ of length $n$, and for every computation path of $M$,
  - $M$ halts after precise $f(n)$ steps, and
  - All of its strings are of length precisely $g(n)$ at halting.
    * If $M$ is a TM with input and output, we exclude the first and the last strings.
- $M$ can be deterministic or nondeterministic.



## Precise TMs Are General

**Proposition 17** *Suppose a TM*[a] *$M$ decides $L$ within time (space) $f(n)$, where $f$ is proper. Then there is a precise TM $M'$ which decides $L$ in time $O(n + f(n))$ (space $O(f(n))$, respectively).*

- $M'$ on input $x$ first simulates the TM $M_f$ associated with the proper function $f$ on $x$.
- $M_f$'s output of length $f(|x|)$ will serve as a "yardstick" or an "alarm clock."

[a]It can be deterministic or nondeterministic.



## The Proof (continued)

- If $f$ is a space bound:
  - $M'$ simulates *on* $M_f$'s output string.
  - The total space, not counting the input string, is $O(f(n))$.

## The Proof (concluded)

- If $f$ is a time bound:
  - The simulation of each step of $M$ on $x$ is matched by advancing the cursor on the "clock" string.
  - The simulation stops at the moment the "clock" string is exhausted.
  - The time bound is therefore $O(|x| + f(|x|))$.



## The Most Important Complexity Classes

We write expressions like $n^k$ to denote the union of all complexity classes, one for each value of $k$.

- For example, $\text{NTIME}(n^k) = \bigcup_{j>0} \text{NTIME}(n^j)$.

$$\begin{aligned}
\text{P} &= \text{TIME}(n^k),\\
\text{NP} &= \text{NTIME}(n^k),\\
\text{PSPACE} &= \text{SPACE}(n^k),\\
\text{NPSPACE} &= \text{NSPACE}(n^k),\\
\text{EXP} &= \text{TIME}(2^{n^k}),\\
\text{L} &= \text{SPACE}(\log n),\\
\text{NL} &= \text{NSPACE}(\log n).
\end{aligned}$$



## Complements of Nondeterministic Classes

- From p. 122, we know R, RE, and coRE are distinct.
  - coRE contains the complements of languages in RE, *not* the languages not in RE.
- Recall that the **complement** of $L$, denoted by $\bar{L}$, is the language $\Sigma^* - L$.
  - SAT COMPLEMENT is the set of unsatisfiable boolean expressions.
  - HAMILTONIAN PATH COMPLEMENT is the set of graphs without a Hamiltonian path.



## The Co-Classes

- For any complexity class $\mathcal{C}$, co$\mathcal{C}$ denotes the class

$$\{\bar{L} : L \in \mathcal{C}\}.$$

- Clearly, if $\mathcal{C}$ is a *deterministic* time or space *complexity class*, then $\mathcal{C} = \text{co}\mathcal{C}$.
  - They are said to be **closed under complement**.
  - A deterministic TM deciding $L$ can be converted to one that decides $\bar{L}$ within the same time or space bound by reversing the "yes" and "no" states.
- Whether nondeterministic classes for time are closed under complement is not known (p. 79).

## Comments

- Then co$\mathcal{C}$ is the class

$$\{\bar{L} : L \in \mathcal{C}\}.$$

  - So $L \in \mathcal{C}$ if and only if $\bar{L} \in$ co$\mathcal{C}$.
- But it is *not* true that $L \in \mathcal{C}$ if and only if $L \not\in$ co$\mathcal{C}$.
  - co$\mathcal{C}$ is not defined as $\bar{\mathcal{C}}$.
- For example, suppose $\mathcal{C} = \{\{2, 4, 6, 8, 10, \ldots\}\}$.
- Then co$\mathcal{C} = \{\{1, 3, 5, 7, 9, \ldots\}\}$.
- But $\bar{\mathcal{C}} = 2^{\{1,2,3,\ldots\}^*} - \{\{2, 4, 6, 8, 10, \ldots\}\}$.



## The Quantified Halting Problem

- Let $f(n) \geq n$ be proper.
- Define

$$H_f = \{M; x : M \textit{ accepts} \text{ input } x \text{ after at most } f(|\,x\,|) \text{ steps}\},$$

  where $M$ is deterministic.
- Assume the input is binary.



## $H_f \in \mathsf{TIME}(f(n)^3)$

- For each input $M; x$, we simulate $M$ on $x$ with an alarm clock of length $f(|\,x\,|)$.
  - Use the single-string simulator (p. 57), the universal TM (p. 109), and the linear speedup theorem (p. 62).
- From p. 61, the total running time is $O(\ell k^2 f(n)^2)$, where $\ell$ is the length to encode each symbol or state of $M$ and $k$ is $M$'s number of strings.
- As $\ell = O(\log n)$, the running time is $O(f(n)^3)$, where the constant is independent of $M$.



## $H_f \not\in \mathsf{TIME}(f(\lfloor n/2 \rfloor))$

- Suppose TM $M_{H_f}$ decides $H_f$ in time $f(\lfloor n/2 \rfloor)$.
- Consider machine $D_f(M)$:

  **if** $M_{H_f}(M; M) =$ "yes" **then** "no" **else** "yes"

- $D_f$ on input $M$ runs in the same time as $M_{H_f}$ on input $M; M$, i.e., in time $f(\lfloor \frac{2n+1}{2} \rfloor) = f(n)$, where $n = |\,M\,|$.

## The Proof (concluded)

- First,

$$\begin{aligned} & D_f(D_f) = \text{“yes”} \\ \Rightarrow\ & D_f; D_f \notin H_f \\ \Rightarrow\ & D_f \text{ does not accept } D_f \text{ within time } f(|\, D_f \,|) \\ \Rightarrow\ & D_f(D_f) = \text{“no”} \end{aligned}$$

  a contradiction

- Similarly, $D_f(D_f) =$ “no” $\Rightarrow D_f(D_f) =$ “yes.”



## The Time Hierarchy Theorem

**Theorem 18** *If $f(n) \geq n$ is proper, then*

$$\text{TIME}(f(n)) \subsetneq \text{TIME}(f(2n+1)^3).$$

- The quantified halting problem makes it so.

**Corollary 19** $\text{P} \subsetneq \text{EXP}$.

- $\text{P} \subseteq \text{TIME}(2^n)$ because $\text{poly}(n) \leq 2^n$ for $n$ large enough.
- But by Theorem 18,

$$\text{TIME}(2^n) \subsetneq \text{TIME}((2^{2n+1})^3) \subseteq \text{TIME}(2^{n^2}) \subseteq \text{EXP}.$$



## The Space Hierarchy Theorem

**Theorem 20** *If $f(n)$ is proper, then*

$$\text{SPACE}(f(n)) \subsetneq \text{SPACE}(f(n) \log f(n)).$$

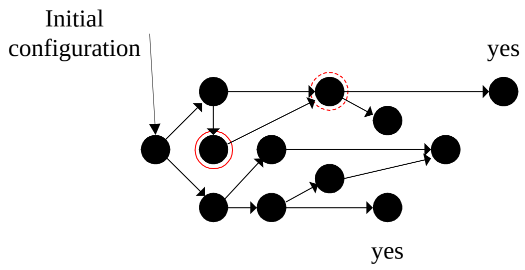
**Corollary 21** $\text{L} \subsetneq \text{PSPACE}$.



## The Reachability Method

- A computation of a TM can be represented by directional transitions between configurations.
- The reachability method constructs a directed graph with all the TM configurations as its nodes and edges connecting two nodes if one yields the other.
- The start node representing the initial configuration has zero in degree.
- When the TM is nondeterministic, a node may have an out degree greater than one.

## Illustration of the Reachability Method




The reachability method may give the edges on the fly without explicitly storing the whole configuration graph.



## Relations between Complexity Classes

**Theorem 22** *Suppose $f(n)$ is proper. Then*

1. SPACE($f(n)$) $\subseteq$ NSPACE($f(n)$), TIME($f(n)$) $\subseteq$ NTIME($f(n)$).
2. NTIME($f(n)$) $\subseteq$ SPACE($f(n)$).
3. NSPACE($f(n)$) $\subseteq$ TIME($k^{\log n + f(n)}$).

- Proof of 2:
  - Explore the computation *tree* of the NTM for "yes."
  - Use the *depth-first* search as $f$ is proper.



## Proof of Theorem 22(2)

- (continued)
  - Specifically, generate a $f(n)$-bit sequence denoting the nondeterministic choices over $f(n)$ steps.
  - Simulate the NTM based on the choices.
  - Recycle the space and then repeat the above steps until a "yes" is encountered or the tree is exhausted.
  - Each path simulation consumes at most $O(f(n))$ space because it takes $O(f(n))$ time.
  - The total space is $O(f(n))$ as space is recycled.



## Proof of Theorem 22(3)

- Let $k$-string NTM

$$M = (K, \Sigma, \Delta, s)$$

  with input and output decide $L \in$ NSPACE($f(n)$).
- Use the reachability method on the configuration graph of $M$ on input $x$ of length $n$.
- A configuration is a $(2k+1)$-tuple

$$(q, w_1, u_1, w_2, u_2, \ldots, w_k, u_k).$$

## Proof of Theorem 22(3) (continued)

- We only care about

$$(q, i, w_2, u_2, \ldots, w_{k-1}, u_{k-1}),$$

where $i$ is an integer between 0 and $n$ for the position of the first cursor.

- The number of configurations is therefore at most

$$|K| \times (n+1) \times |\Sigma|^{(2k-4)f(n)} = O(c_1^{\log n + f(n)}) \qquad (3)$$

for some $c_1$, which depends on $M$.

- Add edges to the configuration graph based on the transition function.



## Proof of Theorem 22(3) (concluded)

- $x \in L \Leftrightarrow$ there is a path in the configuration graph from the initial configuration to a configuration of the form ("yes", $i, \ldots$) [there may be many of them].
- The problem is therefore that of REACHABILITY on a graph with $O(c_1^{\log n + f(n)})$ nodes.
- It is in TIME($c^{\log n + f(n)}$) for some $c$ because REACHABILITY is in TIME($n^k$) for some $k$ and

$$\left[ c_1^{\log n + f(n)} \right]^k = (c_1^k)^{\log n + f(n)}.$$





## The Grand Chain of Inclusions

$$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXP}.$$

- It is known that PSPACE $\subsetneq$ EXP.
- By Corollary 21 (p. 176), we know L $\subsetneq$ PSPACE.
- The chain must break somewhere between L and PSPACE.
- It is suspected that all four inclusions are proper.
- But there are no proofs yet.

## Nondeterministic Space and Deterministic Space

- By Theorem 5 (p. 88),

$$\text{NTIME}(f(n)) \subseteq \text{TIME}(c^{f(n)}),$$

  an exponential gap.

- There is no proof that the exponential gap is inherent, however.
- How about NSPACE vs. SPACE?
- Surprisingly, the relation is only quadratic, a polynomial, by Savitch's theorem.