Introduction to Complexity Theory / Time and Space Complexity

Time for Deciding a Language

- Let us consider $A = \{a^n b^n : n \geq 0\}$.
- How much time does a single-tape TM need to decide A?
- Consider
  $M_1 =$ "On input string w:
  1. Scan the tape and reject if an a appears after a b.
  2. Repeat if a or b appear on the tape:
     1. Scan across the tape, cross an a and a b.
  3. If a's or b's still remain, reject. Otherwise, accept."
- How much "time" does M_1 need for an input w?

Time Complexity

Definition 1

Let M be a TM that halts on all inputs. The running time (or time complexity) of M is the function $f : \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the running time of M on any input of length n.

- If $f(n)$ is the running time of M, we say M runs in time $f(n)$ and M is an $f(n)$ time TM.
- In worst-case analysis, the longest running time of all inputs of a particular length is considered.
- In average-case analysis, the average of all running time of inputs of a particular length is considered instead.
- We only consider worst-case analysis in the course.

Big-O and Small-O

Definition 2

Let $f, g : \mathbb{N} \to \mathbb{R}^+$. $\underline{f(n) = O(g(n))}$ if there are $c, n_0 \in \mathbb{Z}^+$ such that for all $n \geq n_0$,

$$f(n) \leq c(g(n)).$$

- $g(n)$ is an upper bound (or an asymptotic upper bound) for $f(n)$.
- $n^c (c \in \mathbb{R}^+)$ is a polynomial bound.
- $2^{n^d} (d \in \mathbb{R}^+)$ is an exponential bound.

Definition 3

Let $f, g : \mathbb{N} \to \mathbb{R}^+$. f (n) = o (g (n)) if

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0.$$

That is, for any $c \in \mathbb{R}$, there is an n_0 that $f(n) < c(g)$ for all $n \geq n_0$.

Time Complexity of M_1

- Recall
  M_1 = "On input string w:
  1. Scan the tape and reject if an a appears after a b.
  2. Repeat if a or b appear on the tape:
     1. Scan across the tape, cross an a and a b.
  3. If a's or b's still remain, reject. Otherwise, accept."
- Let $|w| = n$.
  - Step 1 takes $O(n)$ (precisely, $\leq n$).
  - Step 2 has $O(n)$ iterations (precisely, $\leq n/2$).
    - ★ An iteration takes $O(n)$ (precisely, $\leq n$).
  - Step 3 takes $O(n)$ (precisely, $\leq n$).
- The TM M_1 decides $A = \{a^n b^n : n \geq 0\}$ in time $O(n^2)$.
  - $O(n^2) = O(n) + O(n) \times O(n) + O(n)$.

Time Complexity Class

Definition 4

Let $t : \mathbb{N} \to \mathbb{R}^+$. The time complexity class $TIME(t(n))$ is the collection of all languages that are decided by a 1-tape $O(t(n))$ time deterministic TM.

- $A = \{a^n b^n : n \geq 0\}$ is decided by M_1 in time $O(n^2)$. $A \in TIME(n^2)$.
- Time complexity classes characterizes languages, not TM's.
  - We don't say $M_1 \in TIME(n^2)$.
- A language may be decided by several TM's.
- Can A be decided more quickly asymptotically?

Deciding $\{a^n b^n : n \geq 0\}$ Faster

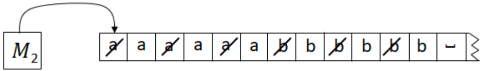
- Consider the following TM:
  M_2 = "On input string w:
  1. Scan the tape and reject if an a appears after a b.
  2. Repeat if a or b appear on the tape:
     1. Scan the tape, cross every other a and b
        Reject if even/odd parities disagree
  3. Accept if all crossed off.
- Analysis of M_2.
  - Step 1 takes $O(n)$.
  - Step 2 has $O(\log n)(= \log_2(n))$ iterations (why?). Each iteration takes $O(n)$.
  - Step 3 takes $O(n)$.
- M_2 decides A in time $O(n \log n)$.

It can be shown (not trivial) that

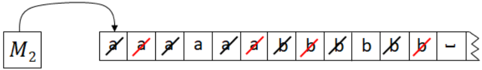
Theorem 5

A 1-tape TM cannot decide A by using fewer than $n \log n$ steps.

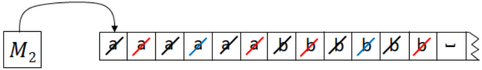
How M_2 works



	Parities
a's	3 (odd)
b's	3 (odd)



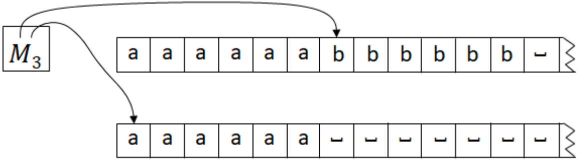
	Parities	
a's	3 (odd)	2 (even)
b's	3 (odd)	2 (even)



	Parities		
a's	3 (odd)	2 (even)	1 (odd)
b's	3 (odd)	2 (even)	1 (odd)

Deciding $\{a^n b^n : n \geq 0\}$ Using a Two-tape TM

- Consider the following two-tape TM:
  M_3 = "On input string w:
  1. Scan tape 1 and reject if an a appears after a b.
  2. Scan tape 1 and copy the a's onto tape 2.
  3. Scan tape 1 and cross an a on tape 2 for a b on tape 1.
  4. If all a's are crossed off before reading all b's, reject. If some a's are left after reading all b's, reject. Otherwise, accept."
- Analysis of M_3.
  - Each step takes $O(n)$.



Model Dependence

- **Computability theory**: model independence
  All reasonable variants of TM's decide the same language (Church-Turing thesis). Therefore model choice doesn't matter.

- **Complexity theory**: model dependence
  Different variants of TM's may decide the same in different time.
  - For the same language $A = \{a^n b^n : n \geq 0\}$.
    - ★ TM M_1 decides A in time $O(n^2)$,
    - ★ TM M_2 decides A in time $O(n \log n)$,
    - ★ Two-tape M_3 decides A in time $O(n)$.

Complexity Relationship with Multitape TM's

Theorem 6

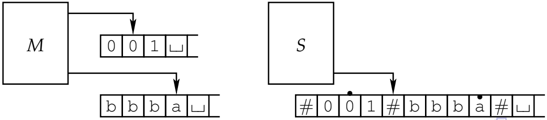
Let $t(n)$ be a function with $t(n) \geq n$. Every $t(n)$ time multitape Turing machine has an equivalent $O(t^2(n))$ time single-tape TM.

Proof.

We analyze the simulation of a k-tape TM M is by the TM S. Observe that each tape of M has length at most $t(n)$ (why?).
For each step of M, S has two passes:

- The first pass gathers information ($O(kt(n))$).
- The second pass updates information with at most k shifts ($O(k^2t(n))$).

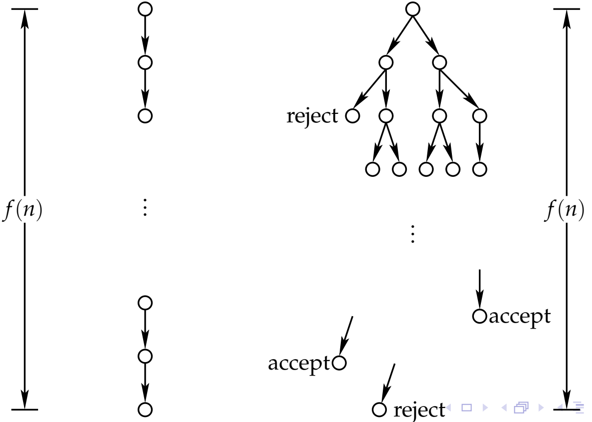
Hence S takes $O(n) + O(k^2t^2(n))$ ($= O(n) + O(t(n)) \times O(k^2t(n))$). Since $t(n) \geq n$, we have S runs in time $O(t^2(n))$ (k is independent of the input). □



Time Complexity of Nondterministic TM's

Definition 7

Let N be a nondeterministic TM that is a decider. The running time of N is a function $f : \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the maximum number of steps among any branch of N's computation on input of length n.



Complexity Relationship with NTM's

Theorem 8

Let $t(n)$ be a function with $t(n) \geq n$. Every $t(n)$ time single-tape NTM has an equivalent $2^{O(t(n))}$ time single-tape TM.

Proof.

Let N be an NTM running in time $t(n)$. Recall the simulation of N by a 3-tape TM D with the address tape alphabet $\Sigma_b = \{1, 2, \ldots, b\}$ (b is the maximal number of choices allowed in N).

Since N runs in time $t(n)$, the computation tree of N has $O(b^{t(n)})$ nodes. For each node, D simulates it from the start configuration and thus takes time $O(t(n))$. Hence the simulation of N on the 3-tape D takes $2^{O(t(n))} (= O(t(n)) \times O(b^{t(n)}))$ time.

By Theorem 6, D can be simulated by a single-tape TM in time $(2^{O(t(n))})^2 = 2^{O(t(n))}$. □

The Class P

- It turns out that reasonable deterministic variants of TM's can be simulated by a TM with a polynomial time overhead.
  - multitape TM's, TM's with random access memory, etc.
- The polynomial time complexity class is rather robust.
  - That is, it remains the same with different computational models.

Definition 9

P is the class of languages decidable in polynomial time on a determinsitic single-tape TM. That is,

$$P = \bigcup_k TIME(n^k).$$

- We are interested in intrinsic characters of computation and hence ignore the difference among variants of TM's in this course.
- Solving a problem in time $O(n)$ and $O(n^{100})$ certainly makes lots of difference in practice.

The Nondeterministic Time Complexity Class

Definition 10

NTIME (t (n)) = { L : L is a language decided by a O (t (n)) time NTM }.

Definition 11

$$NP = \bigcup_k NTIME(n^k).$$

- Recall that class $TIME(t(n))$ and

$$P = \bigcup_k TIME(n^k).$$

Another View of the Class NP

Definition 12

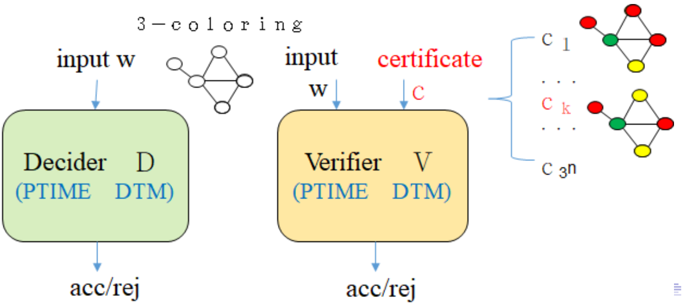
A verifier for a language A is an algorithm V where

$$A = \{w : V \text{ accepts } \langle w, c \rangle \text{ for some } c\}.$$

c is a certificate or proof of membership in A. A polynomial time verifier runs in polynomial time in $|w|$ (not $\langle w, c \rangle$). A language A is polynomially verifiable if it has a polynomial time verifier.

- Note that a certificate has a length polynomial in $|w|$.
  - Otherwise, V cannot run in polynomial time in $|w|$.
- Compare the verifier version of NP with the following: *Language C is Turing-recognizable* $\Leftrightarrow$ *there is a decidable language D such that* $C = \{x \mid \exists y, \langle x, y \rangle \in D, x, y \in \Sigma^*\}$
  - Recognizable lang. $\leftrightarrow$ NP; Decidable lang. $\leftrightarrow$ P
  - $x \in C$ if $\exists y, \langle x, y \rangle \in D$ $\leftrightarrow$ $w \in A$ if $\exists c, \langle w, c \rangle$ accepted by Ptime DTM V.

- **3-colorability problem:** Decide whether vertices of a graph G can be 3-colored with adjacent vertices colored differently.
- There are 3^n possible colorings for a graph with n vertices. Checking all of them by a decider requires exponential time.
- G is 3-colorable $\Leftrightarrow \exists$ a valid 3-color assignment, which serves as a proof.
- Verifier V's work is to, given a certificate, checking whether it is indeed a "proof".



NP and Ptime Verifiers

Theorem 13

A language is in NP if and only if has a polynomial time verifier.

Proof.

Let V be a verifier for a language A running in time n^k. Consider
$N =$ "On input w of length n:

1. Nondeterministically select string c of length $\leq n^k$.
2. Run V on $\langle w, c \rangle$.
3. If V accepts, accept; otherwise, reject."

Conversely, let the NTM N decide A and c the address of an accepting configuration in the computation tree of N. Consider
$V =$ "On input $\langle w, c \rangle$:

1. Simulate N on w from the start configuration by c.
2. If the configuration with address c is accepting, accept; otherwise, reject." □

NP and Ptime Verifiers

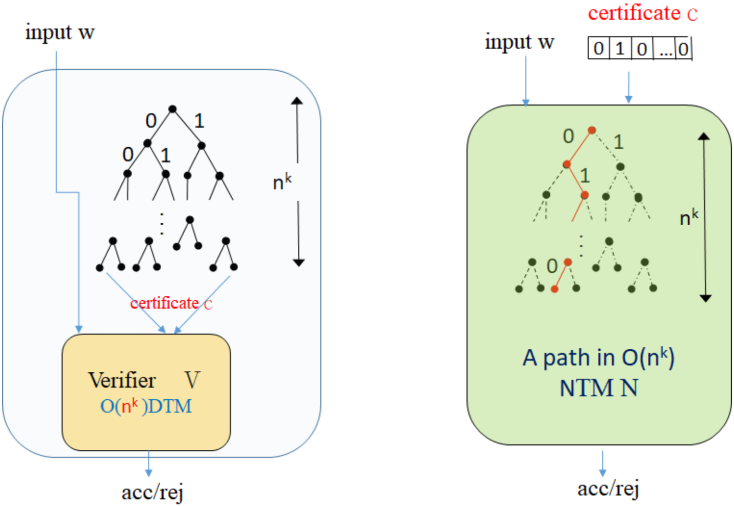


Figure: (Left) Verifier $V \Rightarrow$ NTM N. (Right) NTM $N \Rightarrow$ Verifier V.

Hamiltonian Paths

- A Hamiltonian path in a directed graph G is a path that goes through every node exactly once.

Theorem 14

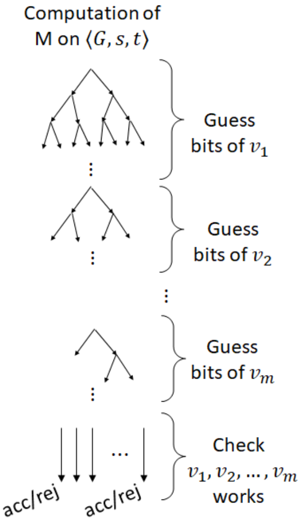
HAMPATH $\in$ *NP*.

Proof.

"On input $\langle G, s, t \rangle$ (assume G has m nodes)

1. Nondeterministically write a sequence $v_1, v_2, \ldots, v_m$ of m nodes.
2. Accept if $v_1 = s$, $v_m = t$, each (v_i, v_{i+1}) is an edge and no v_i repeats.
3. Reject if any condition fails"

(Fig. from M. Sipser's class notes)



The Class *coNP*

Definition 15

$coNP = \{L : \overline{L} \in NP\}$.

- $\overline{HAMPATH} \in coNP$ since $\overline{\overline{HAMPATH}} = HAMPATH \in NP$.
  - $\overline{HAMPATH}$ does not appear to be polynomial time verifiable.
  - What is a certificate showing there is no Hamiltonian path?
- We do not know if *coNP* is different from *NP*.
- Recall
  - *P* is the class of languages which membership can be decided quickly.
  - *NP* is the class of languages which membership can be verified quickly.
- $$L \in P \text{ implies } L \in NP \text{ for every language } L.$$

P vs NP

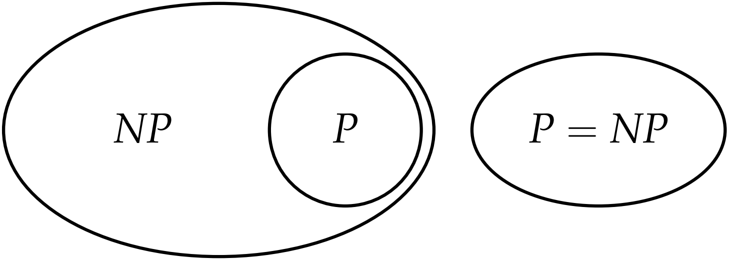


Figure: Possible Relation between P and NP

- To the best of our knowledge, we only know

$$NP \subseteq EXPTIME = \bigcup_k TIME(2^{n^k}). \qquad \text{(Theorem 8)}$$

- Particularly, we do no know if $P \stackrel{?}{=} NP$.

Satisfiability

- Let $\mathbb{B} = \{0, 1\}$ be the truth values.
- A Boolean variable takes values from $\mathbb{B}$.
- Recall the Boolean operations

$$\begin{array}{rclcrclcrcl} 0 \wedge 0 & = & 0 & & 0 \vee 0 & = & 0 & & & & \\ 0 \wedge 1 & = & 0 & & 0 \vee 1 & = & 1 & & \overline{0} & = & 1 \\ 1 \wedge 0 & = & 0 & & 1 \vee 0 & = & 1 & & \overline{1} & = & 0 \\ 1 \wedge 1 & = & 1 & & 1 \vee 1 & = & 1 & & & & \end{array}$$

- A Boolean formula is an expression constructed from Boolean variables and opearations.
  - $\phi = (\overline{x} \wedge y) \vee (x \wedge \overline{z})$ is a Boolean formula.
- A Boolean formula is satisfiable if an assignments of 0′s and 1′s to Boolean variables makes the formula evaluate to 1.
  - ϕ is satisfiable by taking $\{x \mapsto 0, y \mapsto 1, z \mapsto 0\}$.

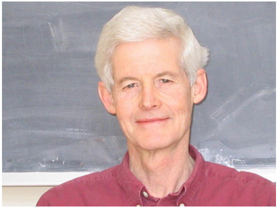
The Satisfiability Problem

- The satisfiability problem is to test whether a Boolean formula is satisfiable.
- Consider

$$SAT = \{\langle\phi\rangle : \phi \text{ is a satisfiable Boolean formula}\}.$$

Theorem 16 (Cook-Levin)

$SAT \in P$ if and only if $P = NP$.



Polynomial Time Reducibility

Definition 17

$f : \Sigma^* \to \Sigma^*$ is a polynomial time computable function if a polynomial time TM M halts with only $f(w)$ on its tape upon any input w.

Definition 18

A language A is polynomial time mapping reducible (polynomial time reducible, or polynomial time many-one reducible) to a language B (written $A \leq_P B$) if there is a polynomial time computable function $f : \Sigma^* \to \Sigma^*$ that

$$w \in A \text{ if and only if } f(w) \in B \text{ for every } w.$$

f is called the polynomial time reduction of A to B.

- Recall the definitions of computable functions and mapping reducibility.

Properties about Polynomial Time Reducibility

Theorem 19

If $A \leq_P B$ and $B \in P$, $A \in P$.

Proof.

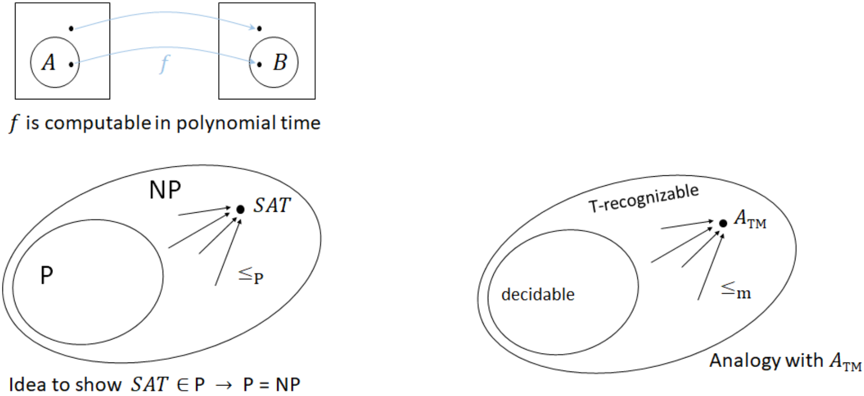
Let the TM M decide B and f a polynomial time reduction of A to B. Consider

$N =$ "On input w:

1. Compute $f(w)$.
2. Run M on $f(w)$."

Since the composition of two polynomials is again a polynomial, N runs in polynomial time. □

Polynomial Time Reducibility



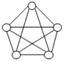
(Fig. from M. Sipser's class notes)

The $3SAT$ Problem

- A literal is a Boolean variable or its negation.
- A clause is a disjunction ($\vee$) of literals.
  - $x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4$ is a clause.
- A Boolean formula is in conjunctive normal form (or a CNF-formula) if it is a conjunction ($\wedge$) of clauses.
  - $(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_2 \vee x_2 \vee \overline{x_5}) \wedge (x_4 \vee x_6)$ is a CNF-formula.
- In a satisfiable CNF-formula, each clause must contain at least one literal assigned to 1.
- A Boolean formula is a 3CNF-formula if it is a CNF-formula whose clauses have three literals.
  - $(x_1 \vee \overline{x_3} \vee x_4) \wedge (x_2 \vee x_2 \vee \overline{x_5}) \wedge (x_4 \vee x_5 \vee \overline{x_6})$ is a 3CNF-formula.
- Consider

$$3SAT = \{\langle \phi \rangle : \phi \text{ is a satisfiable 3CNF-formula}\}.$$

$3SAT \leq_P CLIQUE$



5-clique

- A <u>k−clique</u> of graph G is a k-vertex complete subgraph of G.
- $CLIQUE = \{\langle G, k\rangle \mid \text{graph } G \text{ contains a } k-clique\}$

Theorem 20

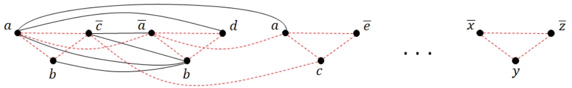
$3SAT \leq_P CLIQUE.$

Proof.

Given a 3CNF-formula $\phi = (a_1 \vee b_1 \vee c_1) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$, find graph G and a number k s.t. $\langle\phi\rangle \in 3SAT$ iff $\langle G, k\rangle \in CLIQUE$. E.g., □

$$\phi = (a \vee b \vee \overline{c}) \wedge (\overline{a} \vee b \vee d) \wedge (a \vee c \vee \overline{e}) \wedge \cdots \wedge (\overline{x} \vee y \vee \overline{z})$$

G =



k = # clauses

Forbidden edges:
1) within a clause
2) inconsistent labels (a and $\overline{a}$)

G has <u>all</u> non-forbidden edges

$3SAT \leq_P CLIQUE$

Proof.

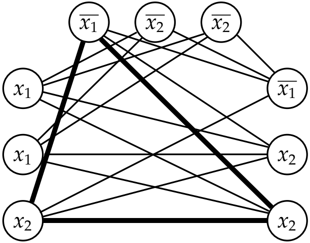
We need gadgets to simulate Boolean variables and clauses in ϕ.

- For each clause $a_i \vee b_i \vee c_i$, add three corresponding nodes to G.
  - G has $3k$ nodes.
- For each pair of nodes in G, add an edge except when
  - the pair of nodes correspond to literals in a clause.
  - the pair of nodes correspond to complementary literals (such as a and $\bar{a}$)

Claim: ϕ is satisfiable if and only if G has a k-clique.

- ($\Rightarrow$) Take any satisfying assignment to ϕ. Pick 1 true literal in each clause. The corresponding nodes in G are a k-clique because they don't have forbidden edges.
- ($\Leftarrow$) Take any k-clique in G. It must have 1 node in each clause. Set each corresponding literal True. That gives a satisfying assignment to ϕ.

$3SAT \leq_P CLIQUE$



$$(x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

NP-Completeness

Definition 21

A language B is NP-complete if

- B is in NP; and
- every A in NP is polynomial time reducible to B.

Theorem 22

If B is NP-complete and $B \in P$, then $P = NP$.

Theorem 23

If $C \in NP$, B is NP-complete, and $B \leq_P C$, then C is NP-complete.

Proof.

Since B is NP-complete, there is a polynomial time reduction f of A to B for any $A \in NP$. Since $B \leq_P C$, there is a polynomial time reduction g of B to C. $g \circ f$ is a polynomial time reduction of A to C. □

Cook-Levin Theorem

Theorem 24

SAT is NP-complete.

Proof.

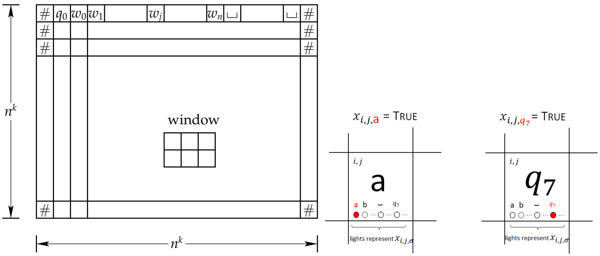
(**In NP**) For any Boolean formula ϕ, an NTM nondeterministically choose a truth assignment. It checks whether the assignment satisfies ϕ. If so, accept; otherwise, reject. Hence $SAT \in NP$.
(**NP-hard**) To show SAT to be NP-hard,we need to show $\forall A \in NP, A \leq_p SAT$.

- **Question:** as there are infinitely many languages A in NP, how to check $A \in NP$?
- **Answer:** each such language A is parameterized by an NTM N and a time bound n^k. I.e., for input w, N operates in $|w|^k$ time, and $L(N) = A$.
- We establish a polynomial-time reduction $f : A \to SAT$ such that

$$f : \Sigma^* \to \text{formulas}$$
$$f(w) = \langle \phi_{N,w} \rangle$$
$$w \in A \text{ iff } \phi_{N,w} \text{ is satisfiable.}$$

□

Cook-Levin Theorem



Proof (cont'd).

Let $A \in NP$ and the NTM N decide A in n^k time. For any input w, a tableau for N on w is an $n^k \times n^k$ table whose rows are the configurations along a branch of the computation of N on w. A tableau of size $n^k \times n^k$ has $n^k \times n^k$ cells. We assume each configuration starts and ends with a # symbol. A tableau is accepting if any of its rows is an accepting configuration.

Each accepting tableau for N on w corresponds to an accepting computation of N on w. We therefore construct a Boolean formula ϕ such that ϕ is satisfiable if and only if there is an accepting tableau for N on w.

Cook-Levin Theorem

Proof (cont'd).

Let $C = Q \cup \Gamma \cup \{\#\}$ where Q and Γ are the states and the tape alphabet of N.

- The variables for $\phi_{N,w}$ are $x_{i,j,s}$, for $1 \leq i, j \leq n^k$ and $s \in C$.
- The Boolean variable denotes the content of the cell $cell[i, j]$. That is, $x_{i,j,s}$ is 1 if and only if $cell[i, j] = s$.
- A satisfiable truth assignment to $\phi_{N,w}$ captures a accepting computation of N on w.

$$\phi_{N,w} = \phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{move}} \wedge \phi_{\text{accept}}$$

To force each cell to contain exactly one symbol from C, consider

$$\phi_{\text{cell}} = \bigwedge_{1 \leq i,j \leq n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}}) \right) \right].$$

Cook-Levin Theorem

Proof (cont'd).

To force the tableau to begin with the start configuration, consider

$$\begin{aligned}\phi_{\text{start}} &= x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ &\quad x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \cdots \wedge x_{1,n+2,w_n} \wedge \\ &\quad x_{1,n+3,\sqcup} \wedge \cdots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}.\end{aligned}$$

To force an accepting configuration to appear in the tableau, consider

$$\phi_{\text{accept}} = \bigvee_{1 \leq i,j \leq n^k} x_{i,j,q_{\text{accept}}}.$$

To force the configuration at row i yields the configuration at row $i+1$, consider a window of 2×3 cells. For example, assume $\delta(q_1, \mathtt{a}) = \{(q_1, \mathtt{b}, R)\}$ and $\delta(q_1, \mathtt{b}) = \{(q_2, \mathtt{c}, L), (q_2, \mathtt{a}, R)\}$. The following windows are valid:

a	q_1	b
q_2	a	c

a	q_1	b
a	a	q_2

a	a	q_1
a	a	b

#	b	a
#	b	a

a	b	a
a	b	q_2

b	b	b
c	b	b

Cook-Levin Theorem

Proof.

Since C is finite, there are only a finite number of valid windows. For any window W

c_1	c_2	c_3
c_4	c_5	c_6

, consider

$$\psi_W = x_{i,j-1,c_1} \wedge x_{i,j,c_2} \wedge x_{i,j+1,c_3} \wedge x_{i+1,j-1,c_4} \wedge x_{i+1,j,c_5} \wedge x_{i+1,j+1,c_6}$$

To force every window in the tableau to be valid, consider

$$\phi_{\text{move}} = \bigwedge_{1 \le i \le n^k, 1 \le j < n^k} \left(\bigvee_{W \text{ is a valid}} \psi_W \right).$$

Finally, consider the following Boolean formula:

$$\phi = \phi_{\text{cell}} \wedge \phi_{\text{start}} \wedge \phi_{\text{accept}} \wedge \phi_{\text{move}}.$$

$|\phi_{\text{cell}}| = O(n^{2k})$, $|\phi_{\text{start}}| = O(n^k)$, $|\phi_{\text{accept}}| = O(n^{2k})$, and $|\phi_{\text{move}}| = O(n^{2k})$. Hence $|\phi| = O(n^{2k})$. Moreover, ϕ can be constructed from N in time polynomial in n. □

3SAT is NP-Complete

Corollary 25

3SAT is NP-complete.

Proof.

We convert the Boolean formula ϕ in the proof of Theorem 24 into a 3CNF-formula.
We begin by converting ϕ into a CNF-formula.
Observe that the conjunction of CNF-formulae is again a CNF-formula. Note that ϕ_{cell}, ϕ_{start}, and ϕ_{accept} are already in CNF (why?). ϕ_{move} is of the following form:

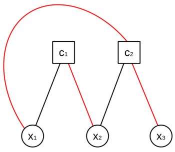
$$\bigwedge_{1 \leq i \leq n^k, 1 \leq j < n^k} \left(\bigvee_{W \text{ is valid}} (l_1 \wedge l_2 \wedge l_3 \wedge l_4 \wedge l_5 \wedge l_6) \right)$$

By the law of distribution, ϕ_{move} can be converted into a CNF-formula. Note that the conversion may increase the size of ϕ_{move}. Yet the size is independent of $|w|$. Hence the size of the CNF-formula ϕ still polynomial in $|w|$.
To a clause of k literals into clauses of 3 literals, consider $l_1 \mapsto (l_1 \vee l_1 \vee l_1)$,
$l_1 \vee l_2 \mapsto (l_1 \vee l_2 \vee l_2)$, and
$l_1 \vee l_2 \vee \cdots l_p \mapsto (l_1 \vee l_2 \vee z_1) \wedge (\overline{z_1} \vee l_3 \vee z_2) \wedge \cdots \wedge (\overline{z_{p-3}} \vee l_{p-1} \vee l_p)$. □

Variants of SAT

- A Boolean formula is in disjunctive normal form (or a DNF-formula) if it is a disjunction ($\vee$) of clauses.
  $(x_1 \wedge \overline{x_2} \wedge \overline{x_3} \wedge x_4) \vee (x_2 \wedge x_2 \wedge \overline{x_5}) \vee (x_4 \wedge x_6)$ is a DNF-formula.
- Consider $\textit{DNF-SAT} = \{\langle \phi \rangle : \phi$ is a satisfiable DNF-formula$\}$.
  - Is is well known that any CNF formula ϕ can be converted into an equivalent DNF formula ϕ', and vice versa.
  - So ..., is *DNF-SAT* NP-complete? If not, why?
- **Planar-SAT:** Planar-SAT = $\{\langle \phi \rangle : \phi,$ whose induced graph is planar, is satisfiable$\}$.
  - **Fact:** Planar-SAT is NP-complete.

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



$c_1 = x_1 \vee \neg x_2$
$c_2 = \neg x_1 \vee x_2 \vee \neg x_3$

More *NP*-Complete Problems

- To find more *NP*-complete problems, we apply Theorem 23.
- Concretely, to show C is *NP*-complete, do
  - prove C is in *NP*; and
  - find a polynomial time reduction of an *NP*-complete problem (say, $3SAT$) to C.
- In Theorem 20, we have shown $3SAT \leq_P CLIQUE$. Therefore

Corollary 26

CLIQUE is NP-complete.

Space Complexity

Definition 27

Let M be a TM that halts on all inputs. The space complexity of M is $f : \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the maximum number of tape cells that M scans on any input of length n.
If the space complexity of M is $f(n)$, we say M runs in space $f(n)$.

Definition 28

If N is an NTM wherein all branches of its computation halts on all inputs. The space complexity of N is $f : \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the maximum number of tape cells that N scans on any branch of its computation for any input of length n.
If the space complexity of N is $f(n)$, we say N runs in space $f(n)$.

Space Complexity Classes

Definition 29

Let $f : \mathbb{N} \to \mathbb{R}^+$. The space complexity classes, $SPACE(f(n))$ and $NSPACE(f(n))$, are

$$\begin{aligned} SPACE(f(n)) &= \{L : L \text{ is decided by an } O(f(n)) \text{ space TM}\} \\ NSPACE(f(n)) &= \{L : L \text{ is decided by an } O(f(n)) \text{ space NTM}\} \end{aligned}$$

$SAT \in SPACE(n)$

Example 30

Give a TM that decides SAT in space $O(n)$.

Proof.

Consider

M_1 = "On input $\langle \phi \rangle$ where ϕ is a Boolean formula:

1. For each truth assignment to $x_1, x_2, \ldots, x_m$ of ϕ, do
   1. Evaluate ϕ on the truth assignment.
2. If ϕ ever eavluates to 1, accept; otherwise, reject."

M_1 runs in space $O(n)$ since it only needs to store the current truth assignment for m variables and $m \in O(n)$. □

Savitch's Theorem

Theorem 31 (Savitch)

For $f : \mathbb{N} \to \mathbb{R}^+$ *with* $f(n) \geq n$, $NSPACE(f(n)) \subseteq SPACE(f^2(n))$.

Proof.

Let N be an NTM deciding A in space $f(n)$. Assume N has a unique accepting configuration c_{accept} (how?). We construct a TM M deciding A in space $O(f^2(n))$. Let w be an input to N, c_1, c_2 configurations of N on w, and $t \in \mathbb{N}$. Consider

$CANYIELD$ = "On input $c_1, c_2,$ and t [The goal is to check $c_1 \xrightarrow{t} c_2$]:

1. If $t = 1$, test $c_1 = c_2$, or $c_1 \vdash c_2$ in N. If either succeeds, accept; otherwise, reject.
2. If $t > 1$, repeat for all configurations c_m that uses $f(n)$ space
   1. Recursively test $CANYIELD(c_1, c_m, \frac{t}{2}) \wedge CANYIELD(c_m, c_2, \frac{t}{2})$.
      (i.e., $c_1 \xrightarrow{\frac{t}{2}} c_m \wedge c_m \xrightarrow{\frac{t}{2}} c_2$)
   2. If both accept, accept.
3. Reject."

Savitch's Theorem

Proof (cont'd).

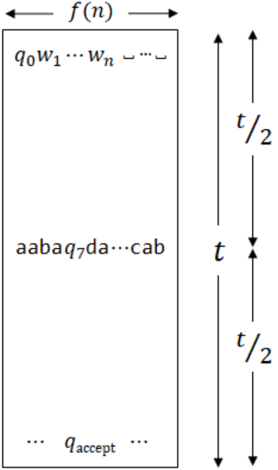
The number of configurations is bounded by $|Q| \times f(n) \times m^{f(n)} = 2^{df(n)}$ for some d, where $m = |\Gamma|$ and $n = |w|$.

M = "On input w:

1. Run $CANYIELD(c_{\text{start}}, c_{\text{accept}}, 2^{df(n)})$.

   (i.e., test $c_{start} \overset{2^{df(n)}}{\rightarrow} c_{accept}$)"

Since $t = 2^{df(n)}$, the depth of recusion is $O(\lg 2^{df(n)}) = O(f(n))$. Moreover, $CANYIELD$ can store its step number, c_1, c_2, t in space $O(f(n))$. Thus M runs in space $O(f(n) \times f(n)) = O(f^2(n))$. □



(Fig. from M. Sipser's class notes)

Definition 32

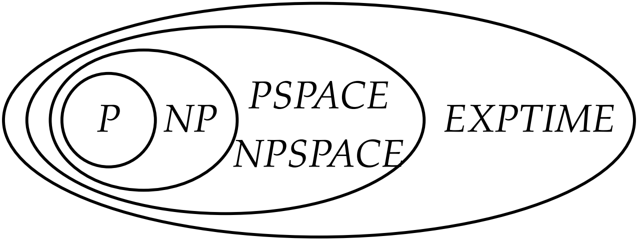
PSPACE is the class of languages decidable by TM's in polynomial space. That is,

$$PSPACE = \bigcup_k SPACE(n^k).$$

- Consider the class of langauges decidable by NTM's in polynomial space $NPSPACE = \bigcup_k NSPACE(n^k)$.
- By Savitch's Theorem, $NSPACE(n^k) \subseteq SPACE(n^{2k})$. Clearly, $SPACE(n^k) \subseteq NSPACE(n^k)$. Hence $NPSPACE = PSPACE$.
- Consider $ALL_{\text{NFA}} = \{M \mid M \text{ is an NFA}, L(M) = \Sigma^*\}$.
  - $ALL_{\text{NFA}} \in coNSPACE(n)$.
    (Why? Can you show $\overline{ALL_{\text{NFA}}} \in NSPACE(n)$? Hint: if $L(M) \neq \emptyset$, then $\exists w \in L(M), |w| \leq 2^{|Q|}$ (Why?))
  - By Savitch's Theorem, $\overline{ALL_{\text{NFA}}} \in NSPACE(n) \subseteq SPACE(n^2)$. Hence $ALL_{\text{NFA}} \in PSPACE$.

P, NP, $PSPACE$, and $EXPTIME$

- $P \subseteq PSPACE$
  - A TM running in time $t(n)$ uses space $t(n)$ (provided $t(n) \geq n$).
- Similarly, $NP \subseteq NPSPACE$ and thus $NP \subseteq PSPACE$.
- $PSPACE \subseteq EXPTIME = \cup_k TIME(2^{n^k})$
  - A TM running in space $f(n)$ has at most $f(n)2^{O(f(n))}$ different configurations (provided $f(n) \geq n$).
    - ★ A configuration contains the current state, the location of tape head, and the tape contents.
- In summary, $P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$.
  - We will show $P \neq EXPTIME$.



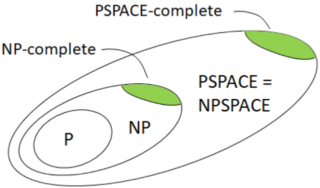
Definition 33

A language B is <u>*PSPACE*-complete</u> if it satisfies

- $B \in PSPACE$; and
- $A \leq_P B$ for every $A \in PSPACE$.

If B only satisfies the second condition, we say it is <u>*PSPACE*-hard</u>.

- We do not define "polynomial space reduction" nor use it. Why?
- Intuitively, a complete problem is most difficult in the class. If we can solve a complete problem, we can solve all problems in the same class easily.



TQBF

- Recall the universal quantifier $\forall$ and the existential quantifier $\exists$.
- When we use quantifiers, we should specify a universe.
  - $\forall x \exists y[x < y \land y < x + 1]$ is false if $\mathbb{Z}$ is the universe.
  - $\forall x \exists y[x < y \land y < x + 1]$ is true if $\mathbb{Q}$ is the universe.
- A quantified Boolean formula is a quantified Boolean formula over the universe $\mathbb{B}$.
- Any formula with quantifiers can be converted to a formula begins with quantifiers.
  - $\forall x[x \geq 0 \implies \exists y[y^2 = x]]$ is equivalent to $\forall x \exists y[x \geq 0 \implies y^2 = x]$.
  - This is called prenex normal form.
- We always consider formulae in prenex normal form.
- If all variables are quantified in a formula, we say the formula is fully quantified (or a sentence).
- Consider

$$TQBF = \{\langle \phi \rangle : \phi \text{ is a true fully quantified Boolean formula}\}.$$

TQBF is *PSPACE*-Complete

Theorem 34

TQBF is PSPACE-complete.

Proof.

We first show $TQBF \in PSPACE$. Consider
$T =$ "On input $\langle \phi \rangle$ where ϕ is a fully quantified Boolean formula:

1. If ϕ has no quantifiers, it has no variables. If $\phi = \text{TRUE}$, accept; or $\phi = \text{FALSE}$, reject.
2. If ϕ is $\exists x \psi$, call T recursively on $\psi[x \mapsto 0]$ and $\psi[x \mapsto 1]$. If either accepts, accept; otherwise, reject.
3. If ϕ is $\forall x \psi$, call T recursively on $\psi[x \mapsto 0]$ and $\psi[x \mapsto 1]$. If both accepts, accept; otherwise, reject.

The depth of recursion is the number of variables. At each level, T needs to store the value of one variable. Hence T runs in space $O(n)$.

Proof (cont'd).

Let M be a TM deciding A in space n^k. We give a polynomial-time reduction f mapping A to TQBF.

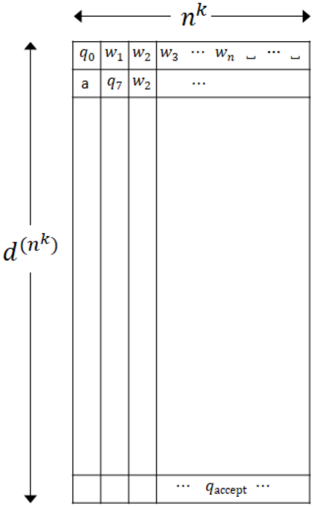
- $f : \Sigma^* \to QBF$ formulas
- $f(w) = \langle \phi_{M,w} \rangle$
- $w \in A$ iff $\phi_{M,w}$ is true

□

• (First attempt): Try the Tableau method, which involves:

- n^k columns and d^{n^k} rows

A nave $\phi_{M,w}$ is of length $n^k \times d^{n^k}$, which is exponential – × Too long!
Notice that such $\phi_{M,w}$ does not use $\exists, \forall$ quantifiers – room for improvement.



(Fig. from M. Sipser's class notes)

$TQBF$ is $PSPACE$-Complete

- (Second attempt): Given configurations c_i and c_j, construct $\phi_{c_i,c_j,t}$ certifying $c_i \xrightarrow{t} c_j$ recursively.

$$\phi_{c_i,c_j,t} = \exists c_{mid}[\overbrace{\phi_{c_i,c_{mid},\frac{t}{2}}}^{(1)} \wedge \overbrace{\phi_{c_{mid},c_j,\frac{t}{2}}}^{(2)}]$$

1. $\exists c_{mid_1}[\phi_{c_i,c_{mid_1},\frac{t}{4}} \wedge \phi_{c_{mid_1},c_{mid},\frac{t}{4}}]$
2. $\exists c_{mid_2}[\phi_{c_{mid},c_{mid_2},\frac{t}{4}} \wedge \phi_{c_{mid_2},c_j,\frac{t}{4}}]$
3. ...
4. $\phi_{\ldots,1}$ is expressed using a 2×3 window, like in Cook-Levin's proof.

- Unfortunately, $\phi_{c_{start},c_{accept},d^{n^k}}$ is exponential in $|w|$, as each recursion doubles the size – × Too long!
- For improvement, notice that the above ϕ does not use $\forall$ quantifiers.

TQBF is *PSPACE*-Complete

Proof (cont'd).

(3rd Attempt) For $t > 1$, let $\phi_{c_i,c_j,t} =$

$$\exists c_{mid} \forall c_g \forall c_h \left[((c_g = c_i \wedge c_h = c_{mid}) \vee (c_g = c_{mid} \wedge c_h = c_j)) \implies \overbrace{\phi_{c_g,c_h,\frac{t}{2}}}^{(1)} \right]$$

1. (1): $\phi_{c_g,c_h,\frac{t}{2}} =$
$\exists c_{m_1} \forall c_{g_1} \forall c_{h_1} [((c_{g_1} = c_g \wedge c_{h_1} = c_{m_1}) \vee (c_{g_1} = c_{m_1} \wedge c_{h_1} = c_h))$
$\implies \phi_{c_{g_1},c_{h_1},\frac{t}{4}}$
2. ...
3. $\phi_{\ldots,1}$ is expressed using a 2×3 window, like in Cook-Levin's proof.

Each level increases the size of $\phi_{c_i,c_j,t}$ by$O(n^k)$. Hence $|\phi_{c_{\text{start}},c_{\text{accept}},2^{dn^k}}| \in O(n^{2k})$.

TQBF is *PSPACE*-Complete

- The 3rd (correct) attempt uses formula of the form

$$\overbrace{\exists\ldots\forall\ldots\exists\ldots\forall\ldots}^{\#\text{ of alternations}=O(n^k)} \quad \psi,$$

where ψ is an unquantified Boolean formula which can be checked in polynomial time.

- Quantifiers allow us to "reuse" subformulas, to make $|\phi_{c_{\text{start}},c_{\text{accept}},2^{dn^k}}|$ short, i.e., $\in O(n^{2k})$!
- Recall that an *NP* language L can be expressed as $x \in L \Leftrightarrow \exists c R(x,c)$, where $R()$ is a polynomial time predicate and c is the certificate.
- How about $\overbrace{\exists\ldots\forall\ldots\exists\ldots\forall\ldots}^{\#\text{ of alt}=O(k)}\psi$?
  - The k-level of the polynomial-time hierarchy.

TM's with Sublinear Space

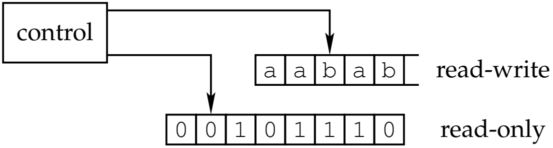


Figure: Schematics for TM's using Sublinear Space

- For sublinear space, we consider TM's with two tapes.
  - a read-only input tape containing the input string; and
  - a read-write work tape.
- The input head cannot move outside the portion of the tape containing the input.
- The cells scanned on the work tape contribute to the space complexity.

Space Complexity Classes L and NL

Definition 35

$\underline{L}$ $(= SPACE(\log n))$ is the class of languages decidable by a TM in logarithmic space.
$\underline{NL}$ $(= NSPACE(\log n))$ is the class of languages decidable by an NTM in logarithmic space.

Example 36

$A = \{\mathtt{0}^k\mathtt{1}^k : k \geq 0\} \in L$.

Proof.

Consider
$M =$ "On input w:

1. Check if w is of the form $\mathtt{0}^*\mathtt{1}^*$. If not, reject.
2. Count the number of $\mathtt{0}$'s and $\mathtt{1}$'s on the work tape.
3. If they are equal, accept; otherwise, reject." □

PATH is in *NL*

Example 37

Recall $PATH = \{\langle G, s, t\rangle : G$ is a directed graph with a path from s to $t\}$.
Show $PATH \in NL$.

Proof.

Consider
$N =$ "On input $\langle G, s, t\rangle$ where G is a directed graph with nodes s and t:

1. Repeat m times (m is the number of nodes in G)
   1. Nondeterministically select the next node for the path. If the next node is t, accept.
2. Reject.

N only needs to store the current node on the work tape. Hence N runs in space $O(\lg n)$. □

- We do not know if $PATH \in L$.

Configurations of TM's with Sublinear Space

Definition 38

Let M be a TM with a separate read-only input tape and w an input string. A configuration of M on w consists of a state, the contents of work tape, and locations of the two tape heads.

- Note that the input w is no longer a part of the configuration.
- If M runs in space $f(n)$ and $|w| = n$, the number of configurations of M on w is at most $|Q| \times n \times f(n) \times |\Gamma|^{f(n)} = n2^{O(f(n))}$.
- Note that when $f(n) \geq \lg n$, $n2^{O(f(n))} = 2^{O(f(n))}$.

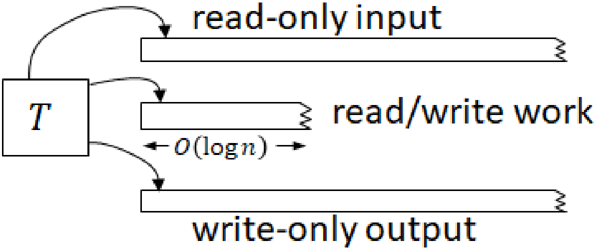
Savitch's Theorem Revisited

- Recall that we assume $f(n) \geq n$ in the theorem.
- We can in fact relax the assumption to $f(n) \geq \lg n$.
- The proof is identical except that we are simulating an NTM N with a read-only input tape.
- When $f(n) \geq \lg n$, the depth of recursion is $\lg(n2^{O(f(n))}) = \lg n + O(f(n)) = O(f(n))$. At each level, $\lg(n2^{O(f(n))}) = O(f(n))$ space is needed.
- Hence $NSPACE(f(n)) \subseteq SPACE(f^2(n))$ when $f(n) \geq \lg n$.

Log Space Reducibility

Definition 39

A log space transducer is a TM with a read-only input tape, a write-only output tape, and a read-write work tape. The work tape may contain $O(\lg n)$ symbols.



(Fig. from M. Sipser's class notes)

Log Space Reducibility

Definition 40

$f : \Sigma^* \to \Sigma^*$ is a <u>log space computable function</u> if there is a log space transducer that halts with $f(w)$ in its work tape on every input w.

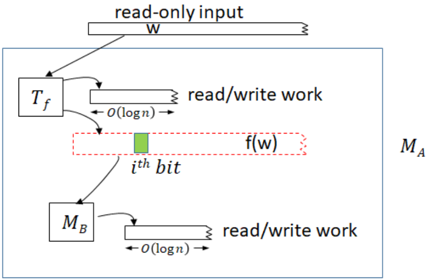
Definition 41

A language A is <u>log space reducible</u> to a language B (written $A \leq_L B$) if there is a log space computable function f such that $w \in A$ if and only if $f(w) \in B$ for every w.

Theorem 42

If $A \leq_L B$ and $B \in L$, $A \in L$.

(First attempt)



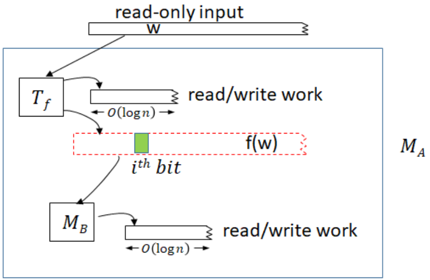
- Can we write down $f(w)$ on M_B's work tape?
  - No. $f(w)$ may need more than logarithmic space.

Properties about Log Space Reducibility

Theorem 42

If $A \leq_L B$ and $B \in L$, $A \in L$.

(First attempt)



- Can we write down $f(w)$ on M_B's work tape?
  - No. $f(w)$ may need more than logarithmic space.

Properties about Log Space Reducibility

Proof.

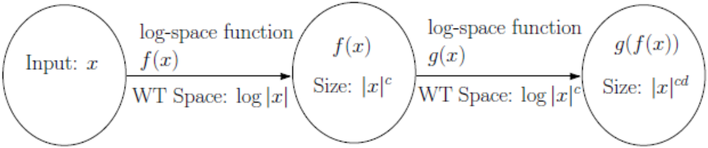
Let a TM M_B decide B in space $O(\lg n)$. Consider
M_A = "On input w:

1. Compute the first symbol of $f(w)$.
2. Simulate M_B on the current symbol.
3. If M_B ever changes its input head, compute the symbol of $f(w)$ at the new location.
   - More precisely, restart the computation of $f(w)$ and ignore all symbols of $f(w)$ except the one needed by M_B.
4. If M_B accepts, accepts; otherwise, reject.

□

Properties about Log Space Reducibility

- We know that polynomial-time reductions are transitive:
  If $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$
- We also crucially used the following similar property:
  If $A \leq_p B$ and $B \in P$, then $A \in P$
  If $A \leq_p B$ and $B \in NP$, then $A \in NP$
- Do we have similar results under $\leq_L$?
- Difficulty:



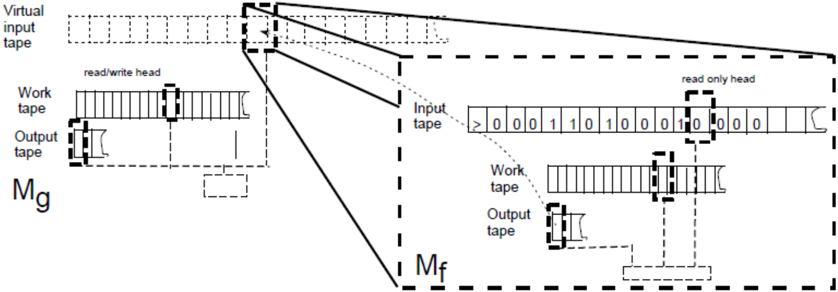
- Total space used $O(\log|x| + \log|x|^c) = O(\log|x|)$. Problem?
- We have to store intermediate result $f(x)$ of size $|x|^c$.

Transitivity of $\leq_L$

Goal: To compute the string $g(f(x))$, given x

- Imagine that we have computed $f(x)$, and its on Tape 1
- The tape-head for Tape 1 is at the start position.
- Now, given this imaginary input string, start computing $g(f(x))$ on Tape 2, just like before
- We know that the work tape Tape 2 needs $\log |f(x)|$ space
- At each step:
  - Read one bit of $f(x)$ from Tape 1 from tape-head position
  - Read one bit of work-tape from tape-head position
  - Move Tape 1, Tape 2 heads by transition function
  - Write one bit on Tape 2, maybe write one bit on Output tape
- Read one bit of $f(x)$ from Tape 1 from tape-head position
  - Don't have $f(x)$ lying around on the imaginary Tape 1
  - Instead, store position of Tape 1 head: $O(\log |f(x)|)$ space
  - Need to read $f(x)_i$: compute using $\log |x|$ space
  - Increment or decrement the pointer for Tape 1 head

Transitivity of $\leq_L$



NL-Completeness

Definition 43

A language B is NL-complete if

- $B \in NL$; and
- $A \leq_L B$ for every $A \in NL$.

- Note that we require $A \leq_L B$ instead of $A \leq_P B$.
- We will show $NL \subseteq P$ (Corollary 46).
- Hence every two problems in NL (except $\emptyset$ and Σ^*) are polynomial time reducible to each other (why?).

Corollary 44

If any NL-complete language is in L, then $L = NL$.

NL-Completeness

Theorem 45

PATH is NL-complete.

Proof.

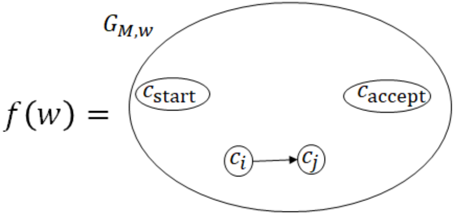
Let an NTM M decide A in $O(\lg n)$ space. We assume M has a unique accepting configuration. Given w, we construct $\langle G_{M,w}, s, t\rangle$ in log space such that M accepts w if and only if $G_{M,w}$ has a path from s to t. $G_{M,w}$ has

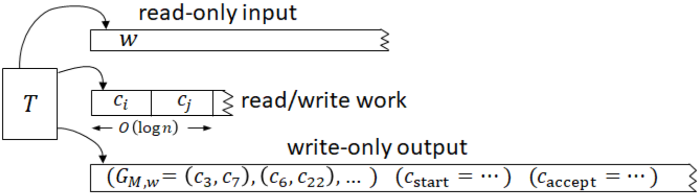
- Nodes: all configurations of M on w,
- Edges: (c_1, c_2) is in $G_{M,w}$ if c_1 yields c_2 in one step.
- s and t are the start and accepting configurations of M on w respectively.

Clearly, M accepts w iff $G_{M,w}$ has a path from s to t. It remains to show that $G_{M,w}$ can be computed by a log space transducer.
T = "on input w

- For all pairs (c_i, c_j) of configurations of M on w.
  - Output those pairs which are legal moves for M.
- Output c_{start} and c_{accept}"

NL-Completeness





$NL \subseteq P$

Corollary 46

$NL \subseteq P$.

Proof.

A TM using space $f(n)$ has at most $n2^{O(f(n))}$ configurations and hence runs in time $n2^{O(f(n))}$. A log space transducer therefore runs in polynomial time. Hence any problem in *NL* is polynomial time reducible to *PATH*. The result follows by $PATH \in P$. □

- The polynomial time reduction in the proof of Theorem 34 can be computed in log space.
- Hence *TQBF* is *PSPACE*-complete with respect to log space reducibility.

$NL = coNL$

Theorem 47 (Immerman - Szelepcsényi)

$NL = coNL$.

Proof.

(Idea) Give an NTM M deciding $\overline{PATH}$ in space $O(\lg n)$. The proof is nontrivial, involving some sort of a counting argument. If interested, check literature. □

L, NL, P, and $PSPACE$

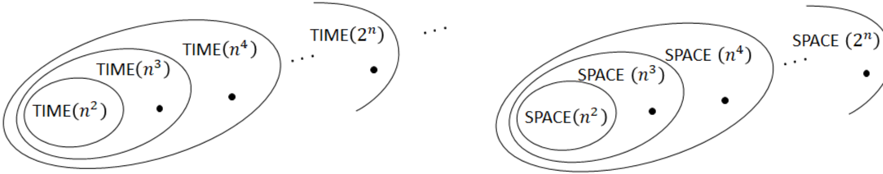
- The relationship between different complexity classes now becomes

$$L \subseteq NL = coNL \subseteq P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$$

- We will prove $NL \subsetneq PSPACE$ in the next chapter.
- Hence at least on inclusion is proper.
  - But we do not know which one.

Intractability

- Recall $P \subseteq NP \subseteq PSPACE = NSPACE$.
- Yet we have not proved any intractable problem.
  - A problem is intractable if it cannot be solved in polynomial time.
- In this chapter, the most difficult problem appears to be $TQBF \in PSPACE$.
- But we do not know if $P \stackrel{?}{=} PSPACE$.
- The time and space hierarchy theorems will show



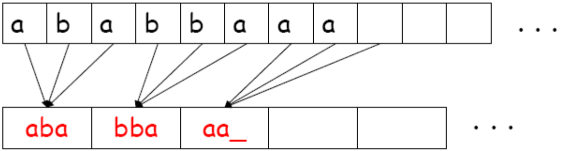
Linear Speedup

Theorem 48

(Linear Speedup - Time) Suppose k-tape TM M decides language L in time $f(n)$. Then for any $\epsilon > 0$, there exists a k-tape TM M' that decides L in time $\epsilon \cdot f(n) + n + 2$.

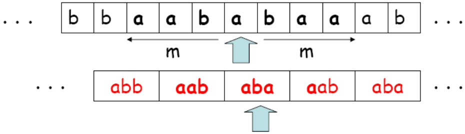
Proof Idea: Suppose $M = (Q, \Sigma, \Gamma ...)$

- (Step 1) Compress input (in $n + 2$ M-steps) onto fresh tape, compressing m ($m = \frac{1}{\epsilon}$) symbols into one. I.e., each symbol of M' corresponds to an m-tuple of tape symbols of M.



Linear Speedup (cont'd)

- (Step 2) Simulate M, m steps at a time, taking $6(f(n)/m)$ M'-steps



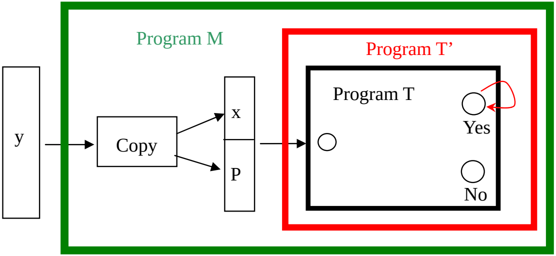
1. Read (in 4 M'-steps) symbols to the left, right and the current position and "store" in finite state control (using $|Q \times \{1, ..., m\}^k \times \Gamma^{3mk}|$ extra states). What is $\{1, ..., m\}^k$ for?
2. Simulate (in 2 M'-steps) the next m steps of M (as M can only modify the current position and one of its neighbours),
3. M' accepts (rejects) if M accepts (rejects).

Using a similar idea, the following also hold:

Theorem 49

(Linear Speedup - Space) If L is decided in space $f(n)$, then for any $\epsilon > 0$, there is a TM deciding L in space $\epsilon f(n) + 2$.

Recall the Diagonalization method for proving the halting problem



- Halt: T enters "Yes" $\Rightarrow$ Not Halt
- Not Halt: T enters "No" $\Rightarrow$ Halt

Diagonalization Method for Proving the Halting Problem

- Consider the language $HALT_{TM} = \{\langle M, x\rangle \mid M \text{ halts on input } x\}$.
- Suppose $HALT_{TM}$ is decidable via a decider D, consider the following table:

	$\langle M_1\rangle$	$\langle M_2\rangle$	$\langle M_3\rangle$	$\cdots$	$\langle M_i\rangle$	$\cdots$
M_1	○	×	×	$\cdots$	$\cdots$	$\cdots$
M_2	×	○	○	$\cdots$	$\cdots$	$\cdots$
M_3	×	×	×	$\cdots$	$\cdots$	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
M_i	×	×	×	$\cdots$	?	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

- Consider language $L = \{\langle M\rangle \mid D \text{ rejects } \langle M, \langle M\rangle\rangle\}$, i.e., calling D on $\langle M, \langle M\rangle\rangle$, if D accepts, $\langle M\rangle \notin L$; if D rejects, $\langle M\rangle \in L$.
- L can clearly be accepted by a TM, say M'.
- Suppose $M' = M_i$. What is the value of entry "$(M_i, \langle M_i\rangle)$"? Contradiction!

Space Hierarchy Theorem

Theorem 50

For any space constructible function $f : \mathbb{N} \rightarrow \mathbb{N}$, there is a language A decidable in $O(f(n))$ space but not in $o(f(n))$ space. In other words, $SPACE(o(f(n)) \subsetneq SPACE(f(n))$.

(Proof Idea)

- The attempt is to use an approach similar to the halting problem proof via diagonalization.
- We design a TM D that can simulate an arbitrary TM M on input w ($|w| = n$) for up to $2^{f(n)}$ steps of M,
  - if the simulation takes more than $2^{f(n)}$ steps, D rejects,
  - if M halts and accepts, D rejects,
  - if M halts and rejects, D accepts.
- Note: D needs a memory of length $f(n)$ (serving as a binary counter) to count up to $2^{f(n)}$ steps of M.

Space Hierarchy Theorem

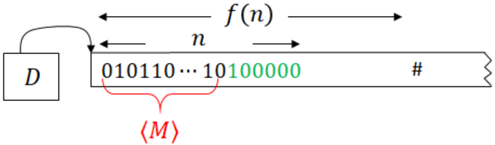
- Consider the language

$$L = \{\langle M\rangle \mid M \text{ rejects } \langle M\rangle \text{ using } f(n) \text{ space}\},$$

  i.e., taking the complement of the diagonal elements.
- Clearly $L \in SPACE(f(n))$ using D.
- Our goal is to show that L cannot be accepted by a TM using $o(f(n))$ space. Suppose otherwise M' accepts L using $o(f(n))$ space.
- Just like the halting problem proof, a contradiction relies on the presence of $(M', \langle M'\rangle)$ entry in the table. meaning that D can simulates M' on $\langle M'\rangle$ till completion.
  - On the surface, it seems okay as $o(f(n)) < f(n) = O(f(n))$
- Does the above argument really work?
  - It is possible that $d \times g(m) > f(m)$ even if $g(n) = o(f(n))$, for some m (e.g., $10^5 n > n^2$ for $n = 100$). If this is the case, D does not have enough space to simulate M' until halt.

Space Hierarchy Theorem

- To overcome the above difficulty, let $L = \{\langle M \rangle \texttt{10}^* \mid M$ rejects $\langle M \rangle \texttt{10}^*$ using $\leq f(n)$ space $\}$.
- By padding the input with 10^*, D simulates any M on an infinite number of inputs $\langle M \rangle 1$, $\langle M \rangle 10$, $\langle M \rangle 100$, ..., $\langle M \rangle 10^m$, ...
  - Eventually there must be a $\langle M \rangle 10^m$ so that $d \times g(|\langle M \rangle 10^m|) < f(|\langle M \rangle 10^m|)$, meaning that D has enough space to complete the simulation.



Space Hierarchy Theorem

Theorem 51

For any space constructible function $f : \mathbb{N} \to \mathbb{N}$, there is a language A decidable in $O(f(n))$ space but not in $o(f(n))$ space. In other words, $SPACE(o(f(n)) \subsetneq SPACE(f(n))$.

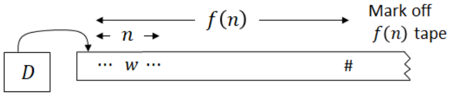
Proof.

Consider language $L = \{\langle M\rangle 10^* \mid M$ rejects $\langle M\rangle 10^*$ using $\leq f(n)$ space $\}$.
Consider $D =$ "On input w:

1. Compute $f(|w|)$ by space constructibility and mark off this much tape. If D ever attempts to use more space, reject.
2. If w is not of the form $\langle M\rangle 10^*$ for some TM M, reject.
3. Simulate M on w. If the simulation takes more than $2^{f(n)}$ M-steps, reject.
4. If M accepts, reject; if M rejects, accept."

Space Hierarchy Theorem

- What is a space constructible function?
  - Function $f : \mathbb{N} \to \mathbb{N}$ with $f(n)$ at least $O(\lg n)$ is called space constructible if the function that maps 1^n to the binary representation of $f(n)$ is computable in space $O(f(n))$. Equivalently, there is a TM that can mark off $f(n)$ cells when given an input of length n.



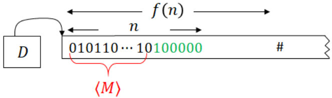
Proof (cont'd).

In Step 3, D simulates M in D's tape alphabet. The simulation hence introduces a constant factor of overhead (independent of $|w|$). That is, if M runs in $g(n)$ space, D runs in $dg(n)$ space for some constant d. Clearly, D is an $O(f(n))$ space TM. For example, if the alphabet of M is $\{0, \ldots, 9\}$ and that of D is $\{0, 1\}$, it takes 4 bits doe D to store a symbol of Σ, resulting in $4 \times g(n)$ memory cells needed for D to simulate M's tape. We next argue that L cannot be decided in $o(f(n))$. □

Space Hierarchy Theorem

Proof (cont'd).

Suppose a TM M' decides L in space $g(n)$ for some $g(n) \in o(f(n))$. Since $g(n) \in o(f(n))$, there is an n_0 that $dg(n) < f(n)$ for every $n \geq n_0$. Consider $\langle M' \rangle \mathtt{1}\mathtt{0}^{n_0}$. Since $dg(n_0) < f(n_0)$, D's simulation on M' has enough space and runs until M' halts, or tries to use more than $f(n)$ space of $2^{f(n)}$ steps. In the latter case, D rejects. M' accepts $\langle M' \rangle \mathtt{1}\mathtt{0}^{n_0}$ if and only if M' rejects $\langle M' \rangle \mathtt{1}\mathtt{0}^{n_0}$, as $L(D) = L$. □



- Why do we need to "pad" $\langle M \rangle$ with 10^*?
  - Suppose we let $L = \{\langle M \rangle \mid M \text{ rejects } \langle M \rangle \text{ using } \leq f(n) \text{ space }\}$. It is possible that $d \times g(m) > f(m)$ even if $g(n) = o(f(n))$, for some m (e.g., $10^5 n > n^2$ for $n = 100$). If this is the case, D does not accept $\langle M' \rangle$ as D does not have enough space to simulate M' until halt.
  - By padding the input with 10^*, D simulates any M on an infinite number of inputs $\langle M \rangle 1$, $\langle M \rangle 10$, $\langle M \rangle 100$, ..., $\langle M \rangle 10^m$, ...

Space Hierarchy Theorem

Corollary 52

Let $f_1, f_2 : \mathbb{N} \to \mathbb{N}$ with $f_1(n) \in o(f_2(n))$ and f_2 space constructible. $SPACE(f_1(n)) \subsetneq SPACE(f_2(n))$.

- We can show n^c is space constructible for any $c \in^{\geq 0}$.
- Observe that for any $\epsilon_1, \epsilon_2 \in \mathbb{R}^{\geq 0}$ with $\epsilon_1 < \epsilon_2$, there are $c_1, c_2 \in^{\geq 0}$ that $0 \leq \epsilon_1 < c_1 < c_2 < \epsilon_2$. Therefore

Corollary 53

For any $\epsilon_1, \epsilon_2 \in \mathbb{R}$ with $0 \leq \epsilon_1 < \epsilon_2$, $SPACE(n^{\epsilon_1}) \subsetneq SPACE(n^{\epsilon_2})$.

More Applications of Space Hierarchy Theorem

Corollary 54

$NL \subsetneq PSPACE$.

Proof.

By Savitch's theorem, $NL \subseteq SPACE(\lg^2 n)$. By space hierarchy theorem, $SPACE(\lg^2 n) \subsetneq SPACE(n)$. □

- Recall that $TQBF$ is $PSPACE$-complete. Hence $TQBF \notin NL$.

Corollary 55

$PSPACE \subsetneq EXPSPACE = \cup_k SPACE(2^{n^k})$.

- So far, we know

$$NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subseteq EXPSPACE.$$

Time Constructibility

Definition 56

$t : \mathbb{N} \to \mathbb{N}$ with $t(n)$ at least $O(n \lg n)$ is called time constructible if the function that maps 1^n to the binary representation of $t(n)$ is computable in time $O(t(n))$.

- That is, $t(n)$ is time constructible if there is an $O(t(n))$ time TM that always halts with the binary representation of $t(n)$ on input 1^n.

Theorem 57

For any time constructible function $t : \mathbb{N} \to \mathbb{N}$, there is a language A decidable in $O(t(n))$ time but not in $o(\frac{t(n)}{\lg t(n)})$ time. In other words, $TIME(o(\frac{t(n)}{\lg t(n)})) \subsetneq TIME(t(n))$.

Time Hierarchy Theorem

Proof.

Consider D = "On input w:

1. Compute $t(|w|)$ by time constructibility and store $\lceil t(n)/\lg t(n) \rceil$ in a binary counter. If this counter ever reaches 0, reject.
2. If w is not of the form $\langle M \rangle \mathtt{10}^*$ for some TM M, reject.
3. Simulate M on w for $\frac{t(n)}{\lg t(n)}$ steps (by decrementing the binary counter).
    - if M accepts, reject;
    - if M rejects; accept."

- Why do we lose a factor of $\lg t(n)$?
    - D can simulate M with a log factor time overhead due to the step counter.

Applications of Time Hierarchy Theorem

Corollary 58

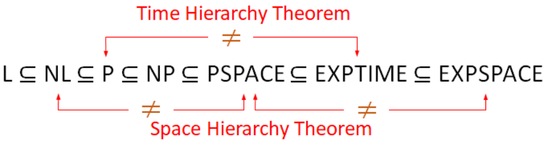
For $t_1, t_2 : \mathbb{N} \to \mathbb{N}$ *with* $t_1(n) \in o(t_2(n)/\lg t_2(n))$ *and* t_2 *time constructible.* $TIME(t_1(n)) \subsetneq TIME(t_2(n))$.

Corollary 59

For any $\epsilon_1, \epsilon_2 \in \mathbb{R}$ *with* $0 \leq \epsilon_1 < \epsilon_2$, $TIME(n^{\epsilon_1}) \subsetneq TIME(n^{\epsilon_2})$.

Corollary 60

$P \subsetneq EXPTIME = \cup_k TIME(2^{n^k})$.



A Provable "Natural" Intractable Problem

- A problem (language) is intractable if it cannot be solved in polynomial time. So, are those NP-complete problems "truly" intractable? (Notice that $P \subsetneq NP$ remains open.)
- As $P \subsetneq EXPTIME \subseteq EXPSPACE$, complete problems for *EXPTIME* and *EXPSPACE* are regarded as "truly" intractable.
- Are there "natural" complete problems for *EXPTIME* and *EXPSPACE*? (Being "natural" by NOT containing a TM encoding.)
- Equivalence of regular languages:
  - $\{\langle M_1, M_2\rangle \mid M_1, M_2$ are DFA, and $L(M_1) = L(M_2)\} \quad \in P$
  - $\{\langle M_1, M_2\rangle \mid M_1, M_2$ are NFA, and $L(M_1) = L(M_2)\}$ – *PSPACE*-complete
  - How about $\{ \langle R_1, R_2\rangle \mid R_1, R_2$ are regular expressions, and $L(R_1) = L(R_2) \}$? – *EXPSPACE*-complete
- The above suggests that regular expressions are more succinct (compact) than DFA/NFA for representing regular languages.