Complexity

doc. RNDr. Ondřej Čepek, Ph.D.

June 17, 2021

Contents

1 Repetition, NS hierarchy

1.1 Abbreviations

- TM - turing machine.
- DTM - deterministic turing machine.
- NTM - non-deterministic turing machine.
- DS - DSPACE
- NS - NSPACE
- PS - deterministic polynomial space
- DT - DTIME
- NT - NTIME
- QBF - quantifiable boolean formula

Theorem 1.1 (NT and DS relation).

$$\forall f : \mathbb{N} \to \mathbb{N} : NT(f(n)) \subseteq DS(f(n))$$

Proof. Construct TM with following algorithm:

1. k = 1
2. do
3. forall y vetev delky k
4. Simuluj M(x) dle y
5. If(Acc) prijmi
6. k++;
7. until vsechny simulace M(x) dle y ktera odmlitly // pokud vsechny vetve odmitli, neexistuje pr. vypocet
8. reject

M pracuje v case $g(n) = \mathcal{O}(f(n))$, vypocet v 4. skonci do g(n) kroku.
Prostor:
1) na ulozeni k potrebujeme g(n) prostoru, $k \leq g(n)$.
2) Na simulace M(x) dle y potrebujeme $\mathcal{O}(g(n))$ prostoru.
Korektnost:
Chceme aby det. TS prijimal stejny jazyk nako puvodni NTS. Pokud existuje pr. vypocet NTS, TS prijme dle nejakeho y. Opacne dojde k $k = g(n)$ a TS odmitne.
Zacykleni nemuze nastat, dle definice cas. slozitosti.
Technical details:
Constructed TM should have 2 tapes:
a) working tape
b) tape to store y (vector that encode branch of NTM) □

Note 1.2. If we are constrained to use single tape, any k tape machine can be compressed to 1 tape machine. We need 2 steps:

1. compress $\Sigma \to \Sigma^k$
2. How to deal with many heads of the original TM? $\Rightarrow$ simulate 1 step of the original machine by many steps in new TM. In total, time complexity is $\mathcal{O}(n^2)$ and space complexity is $\mathcal{O}(n)$.

Theorem 1.3 (NS and DT relation).

$$\forall f : \mathbb{N} \to \mathbb{N}, \forall L \in NS(f(n)) \Rightarrow \exists c_L : L \in DT\left(c_L^{f(n)}\right)$$

Proof. All accepting conf. leaves are bounded by the # of conf. However, # of all paths $2^{c_L^{f(n)}}$. How to avoid double exponent? $\Rightarrow$ Perform BFS in conf. graph. # of edges could be quadratic , however

$$\left(c_L^{f(n)}\right)^2 = \left(c_L^2\right)^{f(n)}$$

c_L^2 is another constant. □

Universal TM Input: (x, y), where x is Godel number of TM to be simulated. y is input of TM.
Alphabet: $\Sigma = \{0, 1\}$.
Working tapes: 3

1. Tape with transition table of M_x.
2. Tape with current state q
3. Working tape

If $S(n)$ is space used by $M_x \Rightarrow$ we need $\max(\lceil \log(t) \rceil, S(n), |x|)$.

Observation 1.4. If initial TM has k tapes and time complexity is $T(n)$ we can compress to 2 tapes with a cost of time complexity being $\mathcal{O}(T(n)\log(n))$.
// TODO write proof (moving blocks on tape)
Therefore simulating TM with multiple tapes could be reduces to 2 tapes, then simulate on Universal TM. Time complexity of Universal TM is dominated by finding the transition $\Rightarrow \mathcal{O}(|x| \cdot T(n))$.

Definition 1.5. Function f is *Space constructible* $\iff \exists$ TM with unary alphabet which marks exactly $|f(n)|$ cells on the working tape. E.g. $\log, e^x$, polynomials.

Theorem 1.6 (Space hierarchy). *Let S_1, S_2 are space constructible functions and*

$$S_1 \in o(S_2) \Rightarrow DS(S_1(n)) \subsetneq DS(S_2(n))$$

Proof. Construction by diagonalization $\Rightarrow$ find a language that is different from all in $\mathcal{O}(S_1)$. □

Definition 1.7. Function f is *Time constructible* $\iff \exists$ TM that does exactly $|f(n)|$ steps. You can think of it as alarm clock.

Note 1.8. Every time constructible function is space constructible.

Theorem 1.9 (Time hierarchy). *Let S_1, S_2 are time constructible functions and*

$$T_1 \cdot \log(T_1(n)) \in o(T_2) \Rightarrow DT(T_1(n)) \subsetneq DT(T_2(n))$$

Note that $\log(T_1(n))$ is required because of k to 2 tapes compression.

Proof. Construction by diagonalization $\Rightarrow$ find a language that is different from all in $\mathcal{O}(T_1)$. □

Theorem 1.10 (Savic). *Under some mild assumptions (space constructibility, functions bigger than $\log(n)$) following statement is true:*

$$NS(f(n)) \subseteq DS(f^2(n))$$

Proof. Find a path in configuration path. We cannot use neither BFS not DFS as the complexity is linear in edges which is exponential comparing to input. Therefore we use recursive algorithm which for all states K reachable by a path of length

$$\frac{c_L^{f(n)}}{2^i}$$

tries to find a path from $C_{init} \to K \to C_{accept}$.
As we divide path by 2 at every recursive call, recursion tree height is equal to $\log_2(n)$. Therefore time complexity is

$$\log_2(c_L^{f(n)}) = f(n) \cdot \log_2(c_L)$$

On each level of recursion we have to store C_{init}, K, C_{accept} which requires $\mathcal{O}(f(n))$ space. Having $\mathcal{O}(f(n))$ levels, total space complexity is $\mathcal{O}(f^2(n))$. □

Note 1.11. Time version of Savic would imply $P = NP$.

1.2 NS hierarchy

Lemma 1.12 (Translation lemma). *Let $S_1(n), S_2(n), f(n)$ be space constructible functions, also*

$$S_2(n) \geq n, f(n) \geq n$$

Then

$$NS(S_1(n)) \subseteq NS(S_2(n)) \Rightarrow NS(S_1(f(n))) \subseteq NS(S_2(f(n)))$$

Lemma allows to replace $n \to f(n)$.

Proof. Let $L_1 \in NS(S_1(f(n)))$ arbitrary, L_1 is recognized by NTS M_1 in space $S_1(f(n))$. We want to prove that $L_1 \in NS(S_2(f(n)))$ by constructing $L_2 \in NS(S_1(n))$ using padding. Define $L_2 := \{x\triangle^i | M_1$ accepts x in space $S_1(|x|+i\}$. Where $\triangle$ is a new symbol, that $\notin$ initial Σ.
Algorithm of $M_2, L(M_2) = L_2$ on input $x\triangle^i$ is the following:

1. Mark $S_1(|x|+i)$ cells on tape.

2. Simulate M_1.

3. if($M_1(x)$ accept $\wedge$ did not use more space than marked in 1) then accept

From the construction, M_2 recognizes L_2 using $S_1(n)$ space. Consequently, $L_2 \in NS(S_1(n))$. Combining with assumption of the lemma

$$L_2 \in NS(S_2(n))$$

And there exists a TM M_3 that recognizes L_2 using $S_2(n)$ space.
The last step is to construct M_4 that recognizes L_1 using $S_2(f(n))$ space. M_4 has 2 working tapes and has following algorithm (M_4 on input x):

1. Mark $f(n)$ cells on 1st tape.
2. Mark $S_2(f(n))$ cells on 2nd tape. Using 1st tape as "input".
3. foreach $x\triangle^i, i = 0, 1, \dots$
   simulate M_3 on input $x\triangle^i$. If head of M_3 is inside x M_4s head is in the same place. Otherwise, head of M_3 is inside padding, so M_4 uses counters to track M_3s position. Counter has length at most $1 + \log i$.
4. if($M_3(x)$ accept) then M_4 accept.
5. else if($1 + \log(i) \leq S_2(f(n))$) then $++i$;
   Counter did not overflow and is less than $S_2(f(n))$.
6. else if($1 + \log(i) > S_2(f(n))$) then reject;

If M_4 accepted input x, M_3 accepted $x\triangle^i$ for some $i \Rightarrow x\triangle^i \in L_2 \Rightarrow M_1$ accepts $x \Rightarrow x \in L_1$. On the other hand, if $x \in L_1 \Rightarrow x\triangle^i \in L_2$ for $i = f(|x|) - |x|$ therefore counter i requires

$$\log(f(|x|) - |x|) \leq S_2(f(|x|))$$

space.

$\square$

Note 1.13. We can also prove Translation Lemma for DS, NT, DT.

Theorem 1.14 (NS hierarchy for polynomials). *Let $\varepsilon > 0$, $r > 1$. Then*

$$NS(n^r) \subsetneq NS(n^{r+\varepsilon})$$

Proof. From the density of rationals:

$$\exists s, t \in \mathbb{N} : r \leq \frac{s}{t} \leq \frac{s+1}{t} \leq r + \varepsilon$$

It is sufficient to prove that:

$$NS(n^{\frac{s}{t}}) \subsetneq NS(n^{\frac{s+1}{t}})$$

Assume by contradiction

$$NS(n^{\frac{s+1}{t}}) \subseteq NS(n^{\frac{s}{t}})$$

Now we use 1.12 for

$$S_1 = \frac{s+1}{t}, S_2 = \frac{s}{t}, f(n) = n^{(s+i)t}, i = 0, 1 \dots, t$$

We get

$$\forall i : NS((n^{(s+i)t})^{(s+1)/t} \subseteq NS((n^{(s+i)t})^{s/t} \Rightarrow \forall i : NS(n^{(s+i)(s+1)}) \subseteq NS(n^{(s+i)s})$$

Now we write for all i:

- $i = 0 : NS(n^{s(s+1)}) \subseteq NS(n^{s^2})$
- $i = 1 : NS(n^{(s+1)(s+1)}) \subseteq NS(n^{(s+1)s})$
- $\dots$
- $i = s : NS(n^{(s+1)2s}) \subseteq NS(n^{(2s)s})$

From the exponents we conclude that for every inequality right side is a subset of left side. So we get a chain of subsets and can use Savic theorem to get a contradiction:

$$NS(n^{(s+1)2s}) \subseteq NS(n^{s^2}) \subseteq DS(n^{2s*s}) \subsetneq DS(n^{2s*s+2s}) \subseteq NS(n^{(s+1)2s})$$

As the beginning and the end are the same sets and the chain of subsets has strict inclusion.

□

2 TM with Oracle

Definition 2.1. Oracle TM is a DTM with an Oracle A (where A is a language) differs from an ordinary DTM by the following:

- Oracle tape (with same alphabet as TM)
- 3 special states: QUERY, YES, NO
- In QUERY state TM moves to YES state if word on the oracle tape $\in A$ (moves to NO o/w). After the answer oracle tape is erased (to reuse space in Space complexity).
- Language of the accepted word by an oracle TM M is $L(M, A)$.

Note 2.2. For NTM definition works the same.

Note 2.3. Ordinary DTM is the same as oracle DTM with $A = \emptyset$.

Consider now a comparison of the oracle DTM, when oracle language A is *not fixed in advance*. Computation forms a tree, that branches at every QUERY.

Observation 2.4. Consider NTM vs Oracle DTM.
"$\Rightarrow$". If NTM M has language $L(M)$, set oracle language $A = L(M)$. "$\Leftarrow$". If oracle language is not recognizable (e.g. HALT), we cannot simulate such NTM.

Definition 2.5. *Turing reducibility* - let A,B languages. We say that A is (deterministically) Turing reducible to B in poly time if there $\exists$ an oracle DTM M working in poly time st

$$A = L(M, B), A \leq^T B$$

Example 2.6. $A \in P \Rightarrow A \leq^T \emptyset$. Since we have poly time algorithm without any oracle.

Definition 2.7. Let A be a language, then

$$\mathbb{P}(A) = \{B | B \leq^T A\}$$

Definition 2.8. Let C be a set of languages then

$$\mathbb{P}(C) = \{B | \exists A \in C : B \leq^T A\}$$

Observation 2.9.

$$\mathbb{P}(\mathbb{P}) = \mathbb{P}$$

Proof. $\mathbb{P} \subseteq \mathbb{P}(\mathbb{P})$. Let $A \in \mathbb{P}$, use A as an oracle with 1 QUERY or use empty oracle.
$\mathbb{P}(\mathbb{P}) \subseteq \mathbb{P}$. Let $B \in \mathbb{P}(\mathbb{P}) \iff \exists A \in \mathbb{P} \exists ODTM\ M : B = L(M, A)$.
To prove the inclusion, we have to construct ordinary DTM that recognizes A. Such TM $\overline{M}$ simulates M and whenever M enters QUERY state, simulate M' to check if word w on oracle tape $\in L(M') = A$.
Now we have to check time complexity.
M makes $P(|x|)$ queries (for p polynomial), as the total number of steps is polynomial. Each query word length is at most $p_w(|x|) = t$. Every query has at most $p_q(|t|)$ steps. In total, query is $p_q(p_w(|x|))$. And the total time complexity of the TM is $p(p_q(p_w(|x|)))$. Which is also polynomial. □

Definition 2.10. *Turing reducibility (non-deterministic)* - let A,B languages. We say that A is non-deterministically Turing reducible to B in poly time if there ∃ an oracle NTM M working in poly time st

$$A = L(M, B), A \leq^{NP} B$$

Definition 2.11. Let A be a language, then

$$\mathbb{NP}(A) = \{B | B \leq^{NP} A\}$$

Definition 2.12. Let C be a set of languages then

$$\mathbb{NP}(C) = \{B | \exists A \in C : B \leq^{NP} A\}$$

Note 2.13. Relativised definition also works for other classes, e.g. EXPTIME.

Definition 2.14.

$$PS = \bigcup_{i=0}^{\infty} DS(n^i) = NPS = \bigcup_{i=0}^{\infty} NS(n^i)$$

Where the 2nd equality holds because of Savic theorem.

Definition 2.15. $PS(A) = \{B |\, B$ accepted by an oracle DTM working in poly space, st $B = L(M, A)$.
Also for class of languages C.

Note 2.16.

$$\mathbb{P} \subseteq \mathbb{NP} \subseteq PS$$

Where last inclusion hold because of $NT(f(n))\ subseteq DS(f(n))$.
Same proof but as for ordinary TM, but with oracle TM that shares same oracle language A.

Observation 2.17. What about $\mathbb{NP}(\mathbb{NP})$? Still an open question, depends on $\mathbb{P} = \mathbb{NP}$. We cannot simply plug NTM back to the original TM with oracle, as NTM serving as an oracle could have multiple accepting or rejecting leaves.

Definition 2.18. Consider a sequence

$$\Sigma_0^P, \Sigma_1^P, \Sigma_2^P, \ldots$$

Where $\Sigma_0^P = \mathbb{P}, \Sigma_{i+1}^P = \mathbb{NP}(\Sigma_i^P)$.
And Polynomial hierarchy(simplified) is:

$$PH = \bigcup_{i \geq 1} \Sigma_i^P$$

Theorem 2.19 (Polynomial hierarchy(simplified)).

$$PH \subseteq PS$$

In plain words, PH is only smth in between NP and PS.

Proof. By induction prove $\forall i : \Sigma_i \subseteq PS$.

- $i = 0 \Rightarrow \mathbb{P} \subseteq PS$
- $i \to i+1$, assume $\Sigma_i \subseteq PS$.

By definition: $\Sigma_{i+1} = \mathbb{NP}(\Sigma_i)$ Then

$$\mathbb{NP}(\Sigma_i) \subseteq \mathbb{NP}(PS)$$

We made set of oracles larger, set of recognized languages cannot shrink.
Now we use $\forall C : \mathbb{NP}(C) \subseteq PS(C)$.
Therefore

$$\mathbb{NP}(C) \subseteq PS(PS) \subseteq PS$$

Last inclusion is up to prove (similar to $\mathbb{P} = \mathbb{P}(\mathbb{P})$):

$$B \in PS(PS) \iff \exists A \in PS \exists DTMM : B = L(M, A)$$

Where M works in poly space.

$$A \in PS \iff \exists DTM\ M_A : A = L(M_A)$$

Where M_A works in poly space. Same in $\mathbb{P} = \mathbb{P}(\mathbb{P})$ proof replace oracle QUERY by DTM computation (which accepts or rejects)
The last thing is to check that used space is polynomial. Which is true since $\forall t \in QUERY$: $|t| \leq p(|x|)$ for some polynomial p. Space taken by DTM M' that computes the QUERY is $p_t(|t|) \leq p_t(p(|x|))$. Also, we can ask exponentially many QUERies, however the space is reused, therefore space is bounded by the largest QUERY. □

3 Polynomial Hierarchy

3.1 Time and Space classes relation

Reminder 3.1 (Space classes).

$$
\begin{aligned}
LOG &= DS(\log n) \\
NLOG &= NS(\log n) \\
POLYLOG &= \bigcup_{i \geq 0} DS(\log^i n) \\
PS &= \bigcup_{i \geq 0} DS(n^i) \\
NSSP &= \bigcup_{i \geq 0} NS(n^i) \\
EXPSPACE &= \bigcup_{i \geq 0} DS(2^{ni})
\end{aligned}
\tag{1}
$$

Reminder 3.2 (Time classes). No way to define LOG class, as we have to read the input.

$$
\begin{aligned}
\mathbb{P} &= \bigcup_{i \geq 0} DT(n^i) \\
\mathbb{NP} &= \bigcup_{i \geq 0} NT(n^i) \\
DEXT &= \bigcup_{i \geq 0} DT(2^{ni}) \\
NEXT &= \bigcup_{i \geq 0} NT(2^{ni}) \\
EXPTIME &= \bigcup_{i \geq 0} DT(2^{n^i}) \\
NEXPTIME &= \bigcup_{i \geq 0} NT(2^{n^i})
\end{aligned}
\tag{2}
$$

Exercise 3.3. Complexity classes

a) $NLOG \subseteq \mathbb{P}$

b) $PS = NPS$

c) $\mathbb{NP} \subseteq PS$

d) $PS = EXPTIME$

e) $NLOG \subsetneq PS \subsetneq EXPSPACE$

f) $\mathbb{P} \subsetneq DEXT \subsetneq EXPTIME$

Proof. a)

$$L \in NS(\log n) \Rightarrow \exists c_L : L \in DT(2^{c_L \log n}) = DT((2^{\log n})^{c_L}) = DT(n^{c_L}) \in \mathbb{P}$$

b) $PS \subseteq NPS$ trivial, as deterministic computation is a special case of non-deterministic.
$NPS \subseteq PS$

$$L \in NS \Rightarrow \exists i : L \in NS(n^i)$$

by Savic 1.10

$$L \in DS(n^{2i}) \Rightarrow L \in PS$$

c) we use time space relation 1.1

$$\forall L \in \mathbb{NP} \Rightarrow \exists i : L \in NT(n^i) \Rightarrow L \in DS(n^i) \Rightarrow \mathbb{NP} \subseteq PS$$

d)

$$L \in PS \Rightarrow \exists i : L \in DS(n^i) \subseteq NS(n^i) \Rightarrow \exists c_L : L \in DT(2^{c_L \log n}) \subseteq DT(2^{n^{i+1}}) \subseteq EXPTIME$$

e) by Savic 1.10

$$NS(\log n) \subseteq DS(\log^2 n)$$

by space hierarchy 1.6

$$\log^2 n \in o(n) \Rightarrow DS(\log^2 n) \subsetneq DS(n)$$

$$L \in PS \Rightarrow \exists i : L \in DS(n^i) \subseteq DS(2^n)$$

by space hierarchy 1.6

$$DS(2^n) \subsetneq DS(2^{2n})$$

f) by time hierarchy 1.9

$$\mathbb{P} \subseteq DT(2^n) \subsetneq DT(2^{2n}) \subseteq DEXT$$

as

$$2^n \log(2^n) = n2^n \in o((2^n)^2)$$

Then

$$DEXT \subseteq DT(2^{n^2}) \subsetneq DT(2^{n^3}) \subseteq EXPTIME$$

as

$$n^2(2^{n^2}) \in o(2^{n^3})$$

$\square$

Note 3.4.

$$NLOG \subseteq \mathbb{P} \subseteq \mathbb{NP} \subseteq PS$$

Also

$$NLOG \subsetneq PS$$

Therefore one of the inclusions in first relation should be strict, or all of them. Still open question.
Moreover, if we prove strict equalities in leftmost and rightmost inclusions $\Rightarrow \mathbb{P} \neq \mathbb{NP}$.

3.2 Polynomial hierarchy again

Definition 3.5. Consider a 3 sequences of classes

$$\Sigma_k, \Pi_k, \Delta_k$$

Where

1. $\Sigma_0 = \Pi_0 = \Delta_0 = \mathbb{P}$.
2. $\Sigma_{k+1} = \mathbb{NP}(\Sigma_k)$.
3. $\Pi_{k+1} = co - \mathbb{NP}(\Sigma_k)$.
4. $\Delta_{k+1} = \mathbb{P}(\Sigma_k)$.

And Polynomial hierarchy is:

$$PH = \bigcup_{i\geq 1} \Sigma_i^P (= \bigcup_{i\geq 1} \Pi_i^P = \bigcup_{i\geq 1} \Delta_i^P)$$

Note 3.6. L is a language alphabet τ

$$\overline{L} = \{x \in \tau^* |\ x \notin L\}$$

If C is a class of languages, then

$$L \in C \iff \overline{L} \in co - C$$

Properties 3.7 (Polynomial Hierarchy)**.**

Relations

a) $\Sigma_1 = \mathbb{NP}$.

b) $\Pi_k = co - \Sigma_k \wedge \Sigma_k = co - \Pi_k$.

c) $\Sigma_{k+1} = \mathbb{NP}(\Pi_k)$.

d) $\Delta_{k+1} = \mathbb{P}(\Pi_k)$.

e) $\Pi_{k+1} = co - \mathbb{NP}(\Pi_k)$.

f) $\Sigma_{k+1} = \mathbb{NP}(\Delta_{k+1})$.

g) $\Pi_{k+1} = co - \mathbb{NP}(\Delta_{k+1})$.

Proof. a) $\mathbb{NP}(\mathbb{P}) = \mathbb{NP}$ also $\mathbb{P}(\mathbb{P}) = \mathbb{P}$.
Proof by embedding oracle computation by simulating using DTM.
b) by definition, for $k \geq 1$

$$\Pi_k = co - \mathbb{NP}(\Sigma_{k-1}) = co - \Sigma_k$$

same for Σ_k by symmetry.
c) we can deduce $\overline{B}$ from questions to B by negating every request.
f)

$$\Sigma_{k+1} = \mathbb{NP}(\Sigma_k) = \mathbb{NP}(\Delta_{k-1}) = \mathbb{NP}(\mathbb{P}(\Sigma_k)) = \mathbb{NP}(\Delta_k)$$

Clearly

$$\mathbb{NP}(\Sigma_k) \subseteq \mathbb{NP}(\mathbb{P}(\Sigma_k))$$

as we made class of query languages larger.

$$L \in \mathbb{NP}(\mathbb{P}(\Sigma_k)) \Rightarrow \exists NTM\ M_n \exists E \in \mathbb{P}(\Sigma_k) : L = L(M_n, E)$$

Also

$$E \in \mathbb{P}(\Sigma_k) \iff \exists DTM\ M_d, \exists Q \in \Sigma_k : E = L(M_d, Q)$$

We want $L \in \mathbb{NP}(\Sigma_k)$. We proceed similarly as in $\mathbb{P}(\mathbb{P}) = \mathbb{P}$ by simulating M_n and every time it ask a query, use M_d to decide.
g)

□

Properties 3.8 (Polynomial Hierarchy - 2)**.**

Relations 2.

a) $\Delta_k = co - \Delta_k$

b) $\mathbb{P}(\Delta_k) = \Delta_k$

c) $\Sigma_k \cup \Pi_k = \Delta_{k+1}$

d) $\Delta_k = \Sigma_k \cap \Pi_k$

e) if $\Sigma_k \subseteq \Pi_k \vee \Pi_k \subseteq \Sigma_k \Rightarrow \Pi_k = \Sigma_k$

Proof. a) for deterministic computation we can negate every answer. Specifically this rule means $\mathbb{P} = co - \mathbb{P}$
b) $\Delta_k \subseteq \mathbb{P}(\Delta_k)$ trivially. Since we can use same language as oracle with 1 query??
$\mathbb{P}(\Delta_k) \subseteq \Delta_k$. Proof by induction on k:

$$k = 0, \mathbb{P}(\mathbb{P}) = \mathbb{P}$$

$k \geq 1$ we have $\mathbb{P}(\Delta_{k-1}) \subseteq \Delta_{k-1}$ By definition $\mathbb{P}(\Delta_k) \subseteq \Delta_k$ is equivalent to

$$\mathbb{P}(\mathbb{P}(\Delta_{k-1})) \subseteq \mathbb{P}(\Delta_{k-1})$$

Both c) and d) use the fact, that deterministic computation is a special case of non-deterministic.
c) proof consists of 2 steps

1. $\Sigma_k \subseteq \Delta_{k+1} = \mathbb{P}(\Sigma_k)$
   Same argument as in first inclusion in b). Ask single query to the same language.

2. $\Pi_k \subseteq \Delta_{k+1} = \mathbb{P}(\Sigma_k)$
   Since we can negate queries
   $$\mathbb{P}(\Sigma_k) = \mathbb{P}(\Pi_k)$$
   Therefore using same argument as above with single query
   $$\Pi_k \subseteq \mathbb{P}(\Pi_k)$$

d) proof consists of 2 steps

1. TODO check first inclusion
$$\Delta_k = \mathbb{P}(\Sigma_{k-1}) \subseteq \mathbb{NP}(\Sigma_{k-1}) = \Sigma_k$$
2.
$$\Delta_k \overset{a)}{=\!=} co-\Delta_k = co-\mathbb{P}(\Sigma_{k-1}) = co-\mathbb{P}(\Pi_{k-1}) \subseteq co-\mathbb{NP}(\Pi_{k-1}) = \Pi_k$$

e) Assume $\Sigma_k \subseteq \Pi_k$ we need to prove reverse.
$$L \in \Pi_k \iff \overline{L} \in co-\Pi_k = \Sigma_k \xrightarrow{assumption} \overline{L} \in co-\Sigma_k = \Pi_k \iff L \in \Sigma_k$$
□

Theorem 3.9 (NP = co-NP). *if* $A \in \mathbb{NP}$*-complete* $\wedge A \in co-\mathbb{NP}$*-complete* $\Rightarrow \mathbb{NP} \subseteq co-\mathbb{NP}$.

Proof. Let $B \in \mathbb{NP}$ arbitrary. By the $\mathbb{NP}$-completeness of A $\exists DTM$ transducer M_t st
$$x \in B \iff M_t(x) \in A$$
$$A \in co-\mathbb{NP} \iff \overline{A} \in \mathbb{NP} \Rightarrow \exists NTM\ M_n : \overline{A} = L(M_n)$$
Also by the properties of polynomial reduction
$$x \in \overline{B} \iff M_t(x) \in \overline{A}$$
$\overline{A}$ can be recognized by M_n, therefore we can recognize $\overline{B}$ by NTM which is a concatenation of M_t and M_n. Therefore $\overline{B} \in \mathbb{NP} \iff B \in co-\mathbb{NP}$.
Same proof is valid for oracle TM. □

Theorem 3.10 (NP $\neq$ co-NP).
$$\mathbb{NP} \cap\ co-\mathbb{NP} \Rightarrow \mathbb{NP} \cup\ co-\mathbb{NP} \subsetneq \Delta_2$$
Proof. Let
$$L = \{(F, F'|\ F \in SAT \wedge F' \in \overline{SAT}\}$$
We can construct DTM with following algorithm for L

1. if($F \in SAT$)
2. if($\overline{F} \in SAT$)
3. accept
4. else
5. reject
6. else
7. reject

Therefore $L \in \mathbb{P}(SAT) \subseteq \Delta_2 = \mathbb{P}(\Sigma_1) = \mathbb{P}(\Pi_k)$.
Define
$$L_2 = \{(F, 0)|\ F \in SAT\} \subseteq L$$
where 0 is e.g. $(x \wedge \neg x)$.
Trivially $L_2 \simeq SAT$ therefore L contains $\mathbb{NP}$-complete language.
Now assume by contradiction $L \in co-\mathbb{NP}$. Using previous proof we can deduce $\mathbb{NP} = co-\mathbb{NP}$. □

3.3 Constrained quantifiers

Definition 3.11.
$$\exists^{p(n)} x : R(x) := \exists x : |x| \leq p(n) \wedge R(x)$$

Definition 3.12.
$$\forall^{p(n)} x : R(x) := \forall x : |x| \leq p(n) \wedge R(x)$$

Definition 3.13. Let C be a class of languages, we define class of languages $\exists C$ as following:
$$A \in (\exists C) \overset{def}{\Longleftrightarrow} \exists B \in C, \exists p : x \in A \iff \exists^{p(|x|)} y : \langle x, y \rangle \in B$$

Note that if $C = \mathbb{P} \Rightarrow \exists \mathbb{P} = \mathbb{NP}$. Since y is a branch of NTM (certificate)

Definition 3.14. Let C be a class of languages, we define class of languages $\forall C$ as following:
$$A \in (\forall C) \overset{def}{\Longleftrightarrow} \exists B \in C, \forall p : x \in A \iff \forall^{p(|x|)} y : \langle x, y \rangle \in B$$

Note that if $C = \mathbb{P} \Rightarrow \forall \mathbb{P} = co - \mathbb{NP}$.

Lemma 3.15. *Let C be an arbitrary class of languages. Then*
$$co - \exists C = \forall (co - C) \tag{3}$$

As
$$A \in (\exists C) \iff \overline{A} \in (\forall (co - C))$$

$$C \subseteq \exists C \wedge C \subseteq \forall C \tag{4}$$

Proof. Equation (3):
Negating the definition
$$A \in (\exists C) \iff \exists B \in C, \exists p : x \in A \iff \exists^{p(|x|)} y : \langle x, y \rangle \in B$$

we get
$$x \in \overline{A} \iff \forall^{p(|x|)} y : \langle x, y \rangle \in \overline{B}$$

Which is the same as
$$\overline{A} \in \forall (co - C) \iff \exists \overline{B} \in co - C, \exists p : \forall^{p(|x|)} y : \langle x, y \rangle \in \overline{B}$$

Equation (4):
For $\exists C$ take $B = A \wedge y = \emptyset$. Clearly $\langle x, \emptyset \rangle \simeq x$.
For $\forall C$ take polynomial with degree 0, only $y = \emptyset$ is accepted. □

Properties 3.16 (Polynomial Hierarchy with quantifiers)**.** a) $\exists \mathbb{P} = \mathbb{NP}$

b) $\forall \mathbb{P} = co - \mathbb{NP}$.

c) $\forall k > 0 : \exists \Sigma_k = \Sigma_k$.

d) $\forall k > 0 : \forall \Pi_k = \Pi_k$.

e) $\forall k \geq 0 : \exists \Pi_k = \Sigma_{k+1}$.

f) $\forall k \geq 0 : \forall\Sigma_k = \Pi_{k+1}$.

Proof. a) "$\exists\mathbb{P} \subseteq \mathbb{NP}$"

$$A \in (\exists\mathbb{P}) \iff \exists B \in \mathbb{P}, \exists p : x \in A \iff \exists^{p(|x|)} y : \langle x,y \rangle \in B$$

Construct an NTM that accepts A ($L(M) = A$):

1. guess y: $|y| \leq p(|x|)$
2. run deterministic verification $\langle x,y \rangle \in B$

B is in $\mathbb{P}$ therefore verification is deterministic. As $|y|$ is bounded by $p(|x|)$ total time complexity is polynomial.
"$\mathbb{NP} \subseteq \exists\mathbb{P}$"

$$A \in \mathbb{NP} \iff \exists NTM\ M : L(M) = A$$

Also

$$x \in A \iff \exists^{p(|x|)} y$$

where y encodes the accepting branch of M on input x.
// todo why??
Consequently any language B of pairs $\langle x,y \rangle$ is in $\mathbb{P}$. So, by definition

$$A \in \exists\mathbb{P}$$

b) using lemma 3.15

$$\forall\mathbb{P} \xlongequal{\mathbb{P}=co-\mathbb{P}} \forall(co-\mathbb{P}) \xlongequal{3.15} co-\exists\mathbb{P} \xlongequal{a)} co-\mathbb{NP}$$

c) trivially $\Sigma_k \subseteq (\exists\Sigma_k)$. Now reversed.

$$A \in (\exists\Sigma_k) \iff \exists B \in \Sigma_k, \exists p : x \in A \iff \exists^{p(|x|)} y : \langle x,y \rangle \in B$$

We want

$$A \in \mathbb{NP}(\Sigma_{k-1}) = \Sigma_k$$

So, we construct an NTM M with oracle $C \in \Sigma_{k-1} : L(M,C) = A$, we also use

$$B \in \mathbb{NP}(\Sigma_{k-1} \iff \exists M_B : B = L(M_B, C)$$

algorithm of M:

1. guess y: $|y| \leq p(|x|)$
2. run M_B on $\langle x,y \rangle$.

Note that

$$\exists y \exists z \iff \exists\langle y,z \rangle$$

d)

$$\forall\Pi_k = \forall(co-\Sigma_k) \xlongequal{3.15} co-\exists\Sigma_k \xlongequal{c)} co-\Sigma_k = \Pi_k$$

Note that now we cannot prove e)
f) follows from e)

$$\forall\Sigma_k = \forall(co-\Pi_k) \xlongequal{3.15} co-\exists\Pi_k \xlongequal{e)} co-\Sigma_{k+1} = \Pi_{k+1}$$

□

4 Polynomial Hierarchy cont

Lemma 4.1. *Let C be an arbitrary class from PH. Then*

$$\forall A \in C \iff A^* \in C$$

Where

$$A^* = \{x|\ \exists n \exists y_1 \in A \dots \exists y_2 \in A : x = (y_1, \dots, y_n)\}$$

In other words, concatenation of the words from A are also in A.
Note that comma separators are important to stay in the same complexity class. We cannot guess the split by brute force.

Proof. Separately for every class in PH.
1) $C = \Sigma_0 = \Pi_0 = \Delta_0 = \mathbb{P}$.
$\Leftarrow$ trivially $A \subseteq A^*$ for $n = 1$. $x = (y)$.

$$\Rightarrow \exists DTM\ M : A = L(M)$$

We run M on all $y_1, y_2, \dots, y_n$. We accept $\iff$ all computations on y_i accepts.
2) $C = \Delta_k$. Same as in 1). We have DTM with an oracle TM.
3) $C = \Sigma_k, k > 0$. We have $\exists NTM\ M$ with oracle $D \in \Sigma_{k-1} : A = L(M, D)$. We simulate computation of M on y_1. In every accepting path, we run M on y_2. And so on.
4) $C = \Pi_k$ If there is at least one $y_i \in co - A$, whole string $(y_1, y_2, \dots, y_n) \in (co - A)^*$. Therefore we proceed roughly the same as in 3), but we proceed for rejecting leaves. □

Consequence 4.2. *If we have $A, B, T \in C \Rightarrow D \in C$ where*

$$D = \{x|\ \exists a \in A, \exists b \in B, \exists t \in T : x = (a, b, t)\}$$

Proof is the same, as of the lemma, but we have multiple TM.

Theorem 4.3 (Polynomial hierarchy consequences).

$$\exists \Pi_k = \Sigma_{k+1}$$

Proof.

$$\exists \Pi_k \subseteq \Sigma_{k+1}$$

$$A \in \exists \Pi_k \iff \exists B \in \Pi_k \exists p : x \in A \iff \exists^{p(|x|)} y : (x, y) \in B$$

Construct an $NTM\ M$ which works as following:

1. read x
2. guess $y : |y| \leq p(|x|)$.
3. ask oracle if $(x, y) \in B$

We get

$$A = L(M, B) \Rightarrow A \in \mathbb{NP}(\Pi_k) = \mathbb{NP}(\Sigma_k) = \Sigma_{k+1}$$

$$\exists \Pi_k \supseteq \Sigma_{k+1}$$

Proof by induction on k.

$k=0$.

$$\Sigma_1 \subseteq \exists\Pi_0 = \exists\mathbb{P} = \mathbb{NP}$$

induction hypothesis: $\Sigma_k \subseteq \exists\Pi_{k-1}$. Let $A \in \Sigma_{k+1}$ be arbitrary. Then

$$\exists NTM\ M \exists B \in \Sigma_k : A = L(M,B)$$

$x \in A \iff \exists$ accepting computation (poly long) of M on input x which asks if $z_1, z_2, \ldots, z_n \in B$ and if $w_1, \ldots, w_n \in co-B$. Alternatively

$$x \in A \iff \exists^{p(|x|)} y, \exists^{p(|x|)} z = (z_1, z_2, \ldots, z_n), \exists^{p(|x|)} w = (w_1, \ldots, w_n)$$

s.t. y encodes an accepting computation of M on X with positive queries $z_1, \ldots, z_n$ and negative queries $w_1, \ldots, w_n$.
We claim that the language L of pairs $(x,y) \in \mathbb{P} \subseteq \Pi_k$.
We know

$$z_i \in B^* \Rightarrow z \in B^* \in \Sigma_k \subseteq^{i.h.} \exists\Pi_{k-1}$$

$$w_i \in (co-B)^* \Rightarrow w \in B^* \in \Sigma_k \subseteq^{i.h.} \exists\Pi_{k-1}$$

So

$$x \in A \iff \exists^{p(|x|)} y \exists^{p(|x|)} z \exists^{p_1(|z|)} y \exists^{p(|x|)} w$$

□

Consequence 4.4 (Definition of PH by alternating quantifiers).

$$A \in \Sigma_k \iff \exists B \in \mathbb{P} \exists p : x \in A \iff \exists^{p(|x|)} y_1, \forall^{p(|x|)} y_2 \ldots : (x, y_1, y_2, \ldots, y_k) \in B$$

Also

$$A \in \Pi_k \iff \exists B \in \mathbb{P} \exists p : x \in A \iff \forall^{p(|x|)} y_1, \exists^{p(|x|)} y_2 \ldots : (x, y_1, y_2, \ldots, y_k) \in B$$

Proof. Proof by induction on k.
For $k=0$ we have no quantifiers.

$$A \in \mathbb{P} \exists B \in \mathbb{P}$$

we can take $A = B$.
$k \to k+1$. Let $A \in \Sigma_{k+1} = \exists\Pi_k$. Then

$$\exists\Pi_k \exists p : x \in A \iff \exists^{p(|x|)} y : (x,y) \in B$$

By the i.p.

$$\Rightarrow (x,y) \in B \iff \forall^{p(|x|)} y_1, \exists^{p(|x|)} y_2 \ldots : ((x,y), y_1, y_2, \ldots, y_k)$$

□

Example 4.5. Language from Σ_2. Optimization version (Boolean minimization).
Input: CNF F
Output: CNF $H : H \equiv F \wedge |H|$ is minimal (we can define minimal in many ways, e.g. minimal in bits to represent).

Now we convert into decision problem.
Input: CNF $F, k \in \mathbb{N}$
Question: $\exists CNF\ H : F \equiv H \wedge |H| \leq k$.
Problem is in Σ_k since

$$F \in BM \iff \exists^{|H| \leq |F|} H, \forall(x_1, \ldots, x_n) : H \leq k \wedge F(x_1, \ldots, x_n) = H(x_1, \ldots, x_n)$$

We also know $BM \in \Sigma_2 - complete$.

Note 4.6. If we have an hard problem, it can be encoded in CNF and solved by CNF machinery.
CNF solvers are highly optimized over more than 20 years.

5 Polynomial Hierarchy cont

Consequence 5.1 (Polynomial hierarchy collapse at level k). *if* $\Sigma_k = \Pi_k$ *for some* $k > 0$ *then*

$$\forall j \geq 0 : \Sigma_{k+j} = \Pi_{k+j} = \Sigma_k$$

Proof. By induction on j. For 0 is true by assumption.
Induction step:

$$\Sigma_{k+j+1} \overset{3.16\ c)}{=\!=\!=} \exists \Pi_{k+j} \overset{I.H.}{=\!=} \exists \Sigma_{k+j}$$

By 3.16 c)

$$\exists \Sigma_{k+j} = \Sigma_{k+j}$$

By I.H. again

$$\Sigma_{k+j} = \Sigma_k$$

Similarly for Π_{k+j+1}.

$$\Pi_{k+j+1} \overset{3.16\ f)}{=\!=\!=} \forall \Sigma_{k+j} \overset{I.H.}{=\!=} \forall \Pi_{k+j} \overset{I.H.}{=\!=} \Sigma_k$$

□

Consequence 5.2. *Either* $\forall k : \Sigma_k \subset \Sigma_{k+1}$ *or PH collapses.*

Proof. Assume

$$\Sigma_k = \Sigma_{k+1}$$

We know

$$\Sigma_{k+1} = \mathbb{NP}(\Sigma_k) = \mathbb{NP}(\Pi_k) \supset \Pi_k$$

Implies by assumption

$$\Pi_k \subseteq \Sigma_k \Rightarrow \Sigma_k = \Pi_k$$

Then by 5.1 PH collapses after k.
In particular for $k = 0$ we get

$$\mathbb{P} = \mathbb{NP} \Rightarrow PH = \mathbb{P}$$

□

Consequence 5.3.

$$\textit{If } \exists k \in \mathbb{N} : \mathbb{P} = \Sigma_0 \subset \Sigma_k \Rightarrow \mathbb{P} \subset \mathbb{NP}$$

Proof. By reversing previous condition. □

Definition 5.4. L is PSPACE-complete if:
$L \in PSPACE$ and
$\forall L_a \in PSPACE : L_a$ is poly time reducible to L.
Note that we use Time reducibility for Space class.

Lemma 5.5. *Let L be PS-complete and $L \in \Sigma_k$ then $PH = \Sigma_k$.*

Proof. Take $L_2 \in PS$ by the polynomial reduction, since

$$\exists DTM\ M_d : x \in L_2 \iff M(x) \in L$$

M_d is a transducer.
Also there is acceptor

$$\exists NTM\ M_n : L = L(M_n, D)$$

for some $D \in \Sigma_{k-1}$ Then we construct new NTM by concatenation of M_d and M_n.

$$L_2 \in PS$$

Therefore

$$PS \subseteq \Sigma_k$$

We already know that

$$PH \subseteq PS$$

Therefore

$$PH = \Sigma_k$$

□

Consequence 5.6. *if $PH = PS$ then*

$$\exists k \in \mathbb{N} : PH = \Sigma_k$$

Assuming that $\exists L \in PS$-complete.
Which implies, that if PH grows infinitely and no PS-complete is in PH. Then $PS \setminus PH$ contains all PS-complete languages.

Proof. We take L, by $PH = PS$

$$\exists k : L \in \Sigma_k$$

then by lemma 5.5

$$PH = PS$$

□

6 PS-complete lang

Definition 6.1 (QBF - quantifiable boolean formula).

1. if x is a variable then x is a QBF and x is a *free* variable
2. if E_1, E_2 are QBF then

$$\neg E_1, (E_1) \wedge (E_2), (E_1) \vee (E_2)$$

   are also QBFs.

   And the status of variables (free/bounded) does not change.

3. if E is a QBF then

$$\exists x(E), \forall x(E)$$

   are also QBFs. And all ocurrences of x become bounded. Status of other variables does not change.

Definition 6.2. QBF problem (language).
Input: QBF F with no free variables.
Question: $F = 1$??

How do we evaluate QBF with no free variables?

- $\exists x(E) \iff E_0 \vee E_1$
- $\forall x(E) \iff E_0 \wedge E_1$

Where E_0 is formula where every occurrence of x is replaced by 0. Similarly E_1.

Example 6.3.

$$\forall x(\forall x(\exists y(x \vee y)) \wedge \neg x)$$

by rules above

$$(\forall x(\exists y(x \vee y)) \wedge \neg 0) \wedge (\forall x(\exists y(x \vee y)) \wedge \neg 1)$$

Note 6.4. SAT - language of satisfiable CNFs. We can think of it as

$$\exists x_1 \exists x_2 \dots \exists x_n (F(x_1, x_2, \dots, x_n)$$

Therefore SAT is a special case of QBF.

Theorem 6.5. *QBF is in PS.*

Proof. We construct DTM to evaluate QBF without free variables as following

- $\neg(E) \rightarrow$ evaluate E and negate all results
- $(E_0) \vee (E_1) \rightarrow$ evaluate E_0, E_1 then by disjunction
- $(E_0) \wedge (E_1) \rightarrow$ evaluate E_0, E_1 then by conjunction
- $\exists(E) \rightarrow$ compute E_0, E_1, then compute $E_0 \vee E_1$
- $\forall(E) \rightarrow$ compute E_0, E_1, then compute $E_0 \wedge E_1$

We have at most n operators. We get binary tree that evaluates the formula. Where every branch is bounded by total length of initial formula.
$\mathcal{O}(n^2)$ is enough space for evaluation. □

Example 6.6.

$$F = \vee_{1\leq j\leq n}(x_i \to y_i) = \vee_{1\leq j\leq n}(\neg x_i \vee y_j)$$

Can be viewed as bipartite graph.
Claim: The resulting formula is shortest CNF representing F. By the completness of resolution. Length changed to $\Theta(n^2)$.
Now 2nd formula

$$H = (\exists z)[(\wedge_{1\leq i\leq n}(x_i \to z)) \wedge (\wedge_{1\leq j\leq n}(z \to y_j))]$$

H is an encoding of F with auxiliary variables. Can be viewed as bipartite graph but with single node in between parts.
However, H is shorter since it is $\Theta(n)$.

Proof. As every x implies every y models of F are:

$$(0,0,0,\ldots,*,*\ldots,*) \cup (*,*\ldots,*,1,1,\ldots,1)$$

Where $*$ represents arbitrary value.
If we rewrite H and substitute $0 \vee 1$ for z we get:

$$[(\wedge_{1\leq i\leq n}(x_i \to 0)) \wedge (\wedge_{1\leq j\leq n}(0 \to y_j))] \vee [(\wedge_{1\leq i\leq n}(x_i \to 1)) \wedge (\wedge_{1\leq j\leq n}(1 \to y_j))]$$

Therefore models are the same. □

Note 6.7. Trick with auxiliary variables is used, if we have a requirement of only one x_i to be 1. Which can be represented by formula:

$$\wedge_{1\leq i,j\leq n}(x_i \vee \neg x_j)$$

Which is $\Theta(n^2)$ and auxiliary variable makes it linear.

Theorem 6.8 (QBF is PS-hard). *QBF is PS-hard (sketch).*

Proof. Every $L \in PS$ arbitrary can be reduces to QBF in poly time.
By the definition of PS

$$\exists DTM\ M : L = L(M), \exists p(n)$$

M accepts L in polynomial space $p(n)$.
We have

$$2^{c_m p(n)}$$

configurations of M and every configuration can be encoded by string of length

$$c_m p(n) = m(n) := m$$

We assume, that is only 1 accepting configuration.
$x \in L \iff \exists$ path of length m in *configuration graph* from $C_0 \to Cacc$. Use similar algorithm as in Savic theorem 1.10, but encode computation in QBF.
Notation, where φ is an encoding of allowed transition in Cook-Levin theorem. We construct QBF ψ.

- $\psi_0(C, C') = 1 \iff \varphi_m(C, C')$ is satisfiable
- $\psi_i(C, C') = 1 \iff$ there exists path $C \to C'$ of length 2^i
- $\psi_m(C_0, C_{acc}) = 1 \iff x \in L$

Obvious idea that would not work

$$\psi_i(C, C') = \exists C_{int}(\psi_i(C, C_{int}) \wedge \psi_i(C_{int}, C'))$$

Since every such change doubles size of the formula, we end up with

$$|\psi_m| \in \Omega(2^m p(n))$$

Main idea

$$\psi_i(C, C') = \exists C_{int} \forall D_1, D_2[(D_1 = C \wedge D_2 = C') \vee (D_1 = C_{int} \wedge D_2 = C')] \Rightarrow \psi(D_1, D_2)$$

Formally, implication could be replaced by $\neg x \vee y$. Now

$$|\psi_i| = |\psi_{i-1}| + \mathcal{O}(m)$$

Therefore

$$|\psi_m| = \mathcal{O}(m^2)$$

Also, going from $\psi_{i-1} \to \psi_i$ we need 1 existential, 1 universal quantifier. In the end, ψ_m has m pairs of alternating existential and universal quantifiers.
Therefore $QBF \in \Sigma_m$. □

6.1 P-completeness

Note 6.9. If we use polynomial time reducibility, almost all languages(except trivial: empty and all words) are $\mathbb{P}$-complete.
Therefore we use a different reducibility.

Definition 6.10 (log-space reducibility). A is *log-space* reducible to B if $\exists$ DTM transducer M that works in log space (excluding input and output tape). Such that $x \in A \iff M(x) \in B$.

Definition 6.11 ($\mathbb{P}$-complete). L is $\mathbb{P}$-complete $\iff L \in \mathbb{P} \wedge \forall A \in \mathbb{P}\, A$ is log-space reducible to L.

Theorem 6.12 (P-complete vs LOG). *Let L be $\mathbb{P}$-complete and $L \in LOG = DS(\log(n)) \Rightarrow \mathbb{P} = LOG$.*

Proof. Since $c_n^{\log n} = (2^{\log n})^{\log c_n} = n^{\log c_n}$.

$$LOG \subseteq \mathbb{P}$$

We want

$$\mathbb{P} \subseteq LOG$$

Let $B \in \mathbb{P}$ arbitrary, we need log-space acceptor for B $\Rightarrow B \in LOG$. From L is $\mathbb{P}$-complete $\Rightarrow \exists$ log-space DTM $M_L : x \in B \iff M_L(x) \in L$. From $L \in LOG \Rightarrow \exists$ log-space DTM acceptor $M_{log} : L = L(M_{log})$.

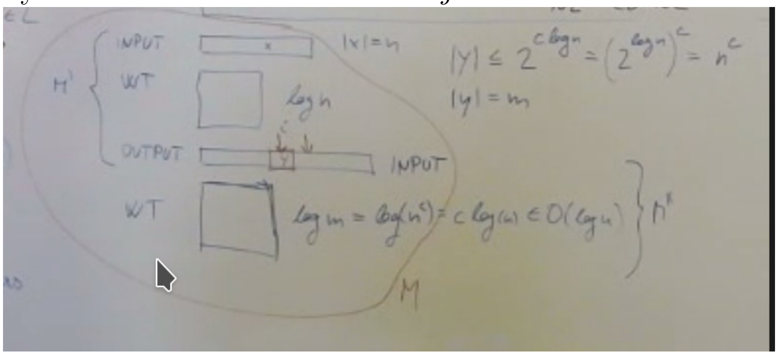
We cannot simply concatenate 2 machines, as output tape of the first machine M_L becomes work tape of the 2nd. Output tape is not guaranteed to be log-space. Let Y be the output of M_L

$$|Y| \leq 2^{c_M \log n} = n^{c_M}$$

Idea: keep just current symbol on output of M_L and the position. Then start the next step of M_{log}. Then restart M_L and discard output with position $< i$. Repeat.
Works, because we do not worry about Time, but space.
We need 2 counters $i, j \in \{1, \ldots, |Y|\}$. Which require

$$\log(n^c) = c \log n$$

Where i keeps the position, updates with every move of the head. j is reset to 0 and is incremented with every symbol M_{log} outputs.
Symbols are discarded until $i = j$.



$\square$

Consequence 6.13. *Let L be $\mathbb{P}$-complete and $L \in NLOG = NS(\log(n)) \Rightarrow \mathbb{P} = NL$. Proof is same, but acceptor is non deterministic.*

Q: what if we use log-space reducibility in $\mathbb{NP}$-complete definition? This is stricter, since if we can reduce in log-space, we can also reduce in polynomial time (by time and space comparison with 2^n).

Theorem 6.14 (Szelepcsenyi-Immerman).

$$NL = co - NL$$

Proof. $\square$