Homework Assignment: Advanced Complexity

Younesse Kaddar

- PDF Version
- http://younesse.net/Complexity/AssignmentComplexity

1. $\nabla \mathrm{NP}$, a new complexity class.

$$\nabla \mathrm{NP} \overset{\text{def}}{=} \{L_1 \backslash L_2 \mid L_1, L_2 \in \mathrm{NP}\}$$

1.

$$\begin{aligned}
\textsf{YesNoSAT} &\overset{\text{def}}{=} \{(F, G) \mid F \in \textsf{SAT}, G \notin \textsf{SAT}\} \\
&= \underbrace{\left\{(F, G) \middle| \begin{cases} F \in \textsf{SAT} \\ G \text{ any boolean expression} \end{cases}\right\}}_{\in \mathrm{NP}} \backslash \underbrace{\left\{(F, G) \middle| \begin{cases} F \text{ any boolean expression} \\ G \in \textsf{SAT} \end{cases}\right\}}_{\in \mathrm{NP}} \\
&\in \nabla \mathrm{NP}
\end{aligned}$$

The underbraced sets are in NP since:

- $\textsf{SAT} \in \mathrm{NP}$
- and the unconstrained boolean expression only requires the Turing machine to check that it is a syntactically well-formed boolean formula.

2.

$$\textsf{Prime} \in \mathrm{coNP} = \{ \underbrace{\textsf{All}}_{\in NP\text{: language associated with the problem that accepts everything}} \backslash L_2 \mid L_2 \in \mathrm{NP}\} \subseteq \nabla \mathrm{NP}$$

Indeed: The non-deterministic Turing machine which, for each input $n \in \mathbb{N}$:

- guesses an integer $d \in [2, n-1]$
- returns **true** if d is a divisor of n, **false** otherwise

recognizes in non-deterministic polynomial time $\underbrace{\overline{\textsf{Prime}}}_{\text{complement of Prime}} \overset{\text{hence}}{\in} \mathrm{NP}$, so that $\textsf{Prime} \in \mathrm{coNP}$.

3.

As 2^k is written with $k+1$ digits in binary:

$$\begin{aligned}
\vert e_n \vert &= \sum_{k=0}^{n-1} \underbrace{(k+1) + (k+2) + 1}_{\text{length of } 2^k \cup 2^{k+1}} + \underbrace{n-1}_{\text{number of "+"}} \\
&= 2 \sum_{k=0}^{n-1} k + 4n + n - 1 \\
&= 2 \frac{n(n-1)}{2} + 5n - 1 \\
&= n(n+4) - 1
\end{aligned}$$

$$\begin{aligned}
V(e_n) &= \left\{ \sum_{k=0}^{n-1} \varepsilon_k 2^k \middle| (\varepsilon_k) \in \{1, 2\}^n \right\} \\
&= \left\{ \sum_{k=0}^{n-1} 2^k + \sum_{k=0}^{n-1} (\varepsilon_k - 1) 2^k \middle| (\varepsilon_k)_{0 \le k \le n-1} \in \{1, 2\}^n \right\} \\
&= \left\{ 2^n - 1 + \sum_{k=0}^{n-1} \varepsilon'_k 2^k \middle| (\varepsilon'_k)_{0 \le k \le n-1} \in \{0, 1\}^n \right\}
\end{aligned}$$

But any integer in $[0, 2^n - 1]$ can be decomposed into a sum of the form $\sum_{k=0}^{n-1} \varepsilon'_k 2^k$, where $(\varepsilon'_k)_{0 \le k \le n-1} \in \{0, 1\}^n$ (decomposition in base 2).

So

$$V(e_n) = [\![2^n - 1, \, 2^{n+1} - 2]\!]$$

4.

$$\mathsf{IsolVal} = \underbrace{\{(e,n) \mid n \in V(e)\}}_{\in \mathrm{NP}} \backslash \underbrace{\Big(\{(e,n) \mid n-1 \in V(e)\} \cup \{(e,n) \mid n+1 \in V(e)\} \Big)}_{\in \mathrm{NP}} \in \nabla\mathrm{NP}$$

Indeed:

> **Lemma**: NP is closed under union (resp. intersection).

Proof: Let $L_1, L_2 \in \mathrm{NP}$ respectively recognized by $\mathscr{M}_1, \mathscr{M}_2$.

Let $\mathscr{M}$ be the Turing machine which, on input w:

1. runs $\mathscr{M}_1$ on w and accepts (resp. rejects) if w is (resp. is not) accepted
2. runs $\mathscr{M}_2$ on w and accepts (resp. rejects) if w is (resp. is not) accepted
3. otherwise rejects (resp. accepts).

$\mathscr{M}$ clearly recognizes $L_1 \cup L_2$ (resp. $L_1 \cap L_2$), and runs in polynomial time, since $\mathscr{M}_1$ and $\mathscr{M}_2$ do. So the result follows.

- $\{(e,n) \mid \; n \in V(e)\} \in \mathrm{NP}$:
  - The non-deterministic Turing machine guesses an element of $V(e)$ and checks if it is equal to n.
  - The "guessing" process can be recursively specified as follows:
    - for a NE of the form m: it picks m
    - for a NE of the form $e_1 + e_2$: it guesses an element in e_1, another one in e_2 and sums both of them
    - for a NE of the form $e_1 \cup e_2$: it non-deterministically chooses between e_1 and e_2 and guesses an element in it

  Likewise, $\{(e,n) \mid \; n-1 \in V(e)\}, \{(e,n) \mid \; n+1 \in V(e)\} \in \mathrm{NP}$
- As NP is closed under union (lemma), $\{(e,n) \mid \; n-1 \in V(e)\} \cup \{(e,n) \mid \; n+1 \in V(e)\} \in \mathrm{NP}$

5.

$$\mathsf{AlmostSAT} = \underbrace{\{S \overset{\text{def}}{=} C_1 \wedge \cdots \wedge C_n \mid S \backslash C_i \in \mathsf{SAT}\}}_{\in \mathrm{NP}} \backslash \underbrace{\{S \overset{\text{def}}{=} C_1 \wedge \cdots \wedge C_n \mid S \in \mathsf{SAT}\}}_{\in \mathrm{NP}} \in \nabla\mathrm{NP}$$

Indeed:

The first set is in NP: one runs the Turing machine $\mathscr{M}$ recognizing SAT on each $S \backslash C_i$ and accepts *if and only if* all of them are satisfiable: this is done in polynomial time, since a linear number of executions of $\mathscr{M}$ (running in *polynomial* time) are performed.

6.

Let $\mathsf{All} \in NP$ be the language associated with the problem that accepts **everything**, $\emptyset \in \mathrm{NP}$ the **empty** language (they are in NP: the machines immediately accept or reject). Then

$$\begin{cases} \mathrm{NP} = \{L_1 \backslash \emptyset \mid L_1 \in \mathrm{NP}\} \subseteq \nabla\mathrm{NP} \\ \mathrm{coNP} = \{\mathsf{All} \backslash L_2 \mid L_2 \in \mathrm{NP}\} \subseteq \nabla\mathrm{NP} \end{cases}$$

so

$$\mathrm{NP} \cup \mathrm{coNP} \subseteq \nabla\mathrm{NP}$$

7.

For all $L \overset{\text{def}}{=} L_1 \backslash L_2, \; L' \overset{\text{def}}{=} L'_1 \backslash L'_2 \in \nabla\mathrm{NP}$ (where $L_1, L_2, L'_1, L'_2 \in \mathrm{NP}$):

$$L \cap L' = (L_1 \cap L_2) \backslash (L'_1 \cup L'_2) \in \nabla\mathrm{NP}$$

Indeed:

- $L_1 \cap L_2 \in \mathrm{NP}$, since NP is closed under intersection (cf. lemma, question 4)
- $L'_1 \cup L'_2 \in \mathrm{NP}$, since NP is closed under union (cf. same lemma)

2. A few simple $\nabla\mathrm{NP}$-complete problems

8.

As $\mathsf{YesNoSAT} \in \nabla\mathrm{NP}$ (question 1), it suffices to show that $\mathsf{YesNoSAT}$ is $\nabla\mathrm{NP}$-hard.

For all $L \overset{\text{def}}{=} L_1 \backslash L_2 \in \nabla\mathrm{NP}$ (where $L_1, L_2, L'_1, L'_2 \in \mathrm{NP}$): as SAT is NP-complete (Cook-Levin theorem), there exist two logspace reductions r_1, r_2 such that:

$$\begin{cases} \forall w,\ w \in L_1 \iff r_1(w) \in \mathsf{SAT} \\ \forall w,\ w \in L_2 \iff r_2(w) \in \mathsf{SAT} \end{cases}$$

Let

$$r \overset{\text{def}}{=} w \longmapsto (r_1(w), r_2(w))$$

Then for all w:

$$\begin{aligned} w \in L_1 \backslash L_2 &\iff w \in L_1 \wedge w \notin L_2 \\ &\iff r_1(w) \in \mathsf{SAT} \wedge r_2(w) \notin \mathsf{SAT} \\ &\iff (r_1(w), r_2(w)) \in \mathsf{YesNoSAT} \end{aligned}$$

Moreover, r runs clearly in logspace, as r_1 and r_2 do.

So

$$\forall L \in \nabla\mathrm{NP},\ L \preccurlyeq_{\mathrm{L}} \mathsf{YesNoSAT}$$

and

> $\mathsf{YesNoSAT}$ is $\nabla\mathrm{NP}$-complete.

9.

$\mathsf{BestClique} \in \nabla\mathrm{NP}$:

$$\mathsf{BestClique} = \mathsf{Clique} \backslash \underbrace{\{(G,k) \mid (G,k+1) \in \mathsf{Clique}\}}_{\in\, \mathrm{NP}} \\ \in \nabla\mathrm{NP}$$

Indeed:

$\mathsf{Clique} \in \mathrm{NP}$: guess a set S of vertices, check if $|S| \geq k$, then check whether all vertices in S are connected by an edge (it take quadratic non-deterministic time, hence polynomial non-deterministic time). Analogously, $\{(G,k) \mid \ (G,k+1) \in \mathsf{Clique}\} \in \mathrm{NP}$.

$\mathsf{BestClique}$ is $\nabla\mathrm{NP}$-hard:

Let us first show that $\mathsf{BestClique}$ is NP-hard (it will come in handy at questions 16 and 17).

> **Lemma**: There exists a logspace reduction r from $\mathsf{3\text{-}SAT}$ to $\mathsf{BestClique}$ such that for all 3-CNF boolean formula φ with m clauses:
>
> $$\begin{cases} \varphi \in \mathsf{3\text{-}SAT} \Longrightarrow r(\varphi) \overset{\text{def}}{=} (G, m) \in \mathsf{BestClique} & \circledast \\ \varphi \notin \mathsf{3\text{-}SAT} \Longrightarrow (G, m-1) \in \mathsf{BestClique} & \circledast\circledast \end{cases}$$
>
> In particular, $\varphi \in \mathsf{3\text{-}SAT} \iff r(\varphi) \in \mathsf{BestClique}$, so $\mathsf{BestClique}$ is NP-hard

We will use the same reduction as the one seen last year (http://younesse.net/Calculabilite/TD7/ Réductions> EX1 and EX3):

For each clause C of r literals in φ, we add r nodes G, each one labeled with a literal from C. We don't connect any nodes stemming from the same clause.

Then, we put edges between each pair of nodes coming from distinct clauses, except for the pairs of the form $(x, \neg x)$, so that any clique in G is of size smaller (or equal) than m.

One easily sees that:

- if φ is satisified by a valuation v: the clique comprised, for each clause C, of one vertex corresponding to one *satisfied* literal, is of size m (and it thereby *maximal*)
- if G has a largest clique c of size m: then c has exactly one node from each clause (at most one since any nodes from a same clause are not connected, and one because it is of size m). Then by setting the corresponding literals to true (and everything else to false), the resulting valuation satifies φ, since each clause has a satisfied literal.

Then, to ensure that if φ is unsatisfied, the largest clique is of size $m-1$: we modify G by artificially taking the union with a new clique of size $m-1$

NB: the reduction is clearly logspace, since:

- m can be computed by reading the input, with one counter
- one can build G by scanning through each clause, with a constant number of pointers

$\nabla\mathrm{NP}$-hardness

We reduce $\mathsf{BestClique}$ from $\mathsf{YesNoSAT}$.

Let φ_1, φ_2 be two CNF formulas having respectively m_1 and m_2 clauses. We can ensure that $m_1 \neq m_2$ by possibly adding new tautological clauses (i.e. of the form $x \vee \neg x$, where x is a fresh variable), which doesn't change the satisfiability of the formulas.

By denoting by r the reduction introduced in the previous lemma:

- $r(\varphi_1) \overset{\text{def}}{=} (G_1, m_1)$ has a largest clique of size:
  - m_1 if φ_1 is satisfiable
  - $m_1 - 1$ otherwise

And similarly for $r(\varphi_2 \overset{\text{def}}{=} (G_2, m_2)$.

Then, we define:

$$G \overset{\text{def}}{=} G_1 \times G_1$$

That is:

- the vertex set of G is the cartesian product of the vertex sets of G_1 and G_2
- in G, (u_1, u_2) and (v_1, v_2) are adjacent if and only if
  - u_1 is adjacent with v_1
  - u_2 is adjacent with v_2

It follows that:

$$(\varphi_1, \varphi_2) \in \mathsf{YesNoSAT} \iff (G, m_1(m_2 - 1)) \in \mathsf{BestClique}$$

Indeed, one cannot have $m_1(m_2 - 1) = (m_1 - 1)m_2$, since it would imply that $\frac{m_1}{m_1 - 1} = \frac{m_2}{m_2 - 1}$, but the function $n \longmapsto \frac{n}{n-1}$ is injective and $m_1 \neq m_2$.

The reduction is logspace since:

- one computes m_1 and m_2 with two counters
- one adds possibly one new tautological formula
- to build $G \overset{\text{def}}{=} G_1 \times G_2$, one only keeps a constant amount of pointers on the working tape (e.g. the current nodes of G_1 and G_2 considered)

On the whole, as $\mathsf{YesNoSAT}$ has been proven to be $\nabla\mathrm{NP}$-hard in **question 8**, so is $\mathsf{BestClique}$.

> As $\mathsf{BestClique} \in \nabla\mathrm{NP}$ and $\mathsf{BestClique}$ is $\nabla\mathrm{NP}$-hard, $\mathsf{BestClique}$ is $\nabla\mathrm{NP}$-complete.

10.

As $\mathsf{IsolVal}$ has been shown to be in $\nabla\mathrm{NP}$ at **question 4**, one has to show that it is $\nabla\mathrm{NP}$-hard.

For all $L \overset{\text{def}}{=} L_1 \backslash L_2 \in \nabla\mathrm{NP}$ (where $L_1, L_2, L_1', L_2' \in \mathrm{NP}$): as $\mathsf{SubsetSum}$ is NP-complete, there exist two logspace reductions r_1, r_2 such that:

$$\begin{cases} \forall w,\ w \in L_1 \iff r_1(w) \overset{\text{def}}{=} (\{a_1, \cdots, a_k\}, t) \in \mathsf{SubsetSum} \\ \forall w,\ w \in L_2 \iff r_2(w) \overset{\text{def}}{=} (\{a_1', \cdots, a_l'\}, t') \in \mathsf{SubsetSum} \end{cases}$$

If $t \geq t'$

For all w, let $r(w) \overset{\text{def}}{=} (e, n)$ be defined by:

- $$e \overset{\text{def}}{=} \overbrace{3a_1 \cup 0 + \cdots + 3a_k \cup 0}^{\overset{\text{def}}{=} e_1} \\ \cup \underbrace{\left(3a_1' \cup 0 + \cdots + 3a_l' \cup 0 + 3(t - t') + 1\right)}_{\overset{\text{def}}{=} e_2}$$
- $$n \overset{\text{def}}{=} 3t$$

It follows that:

If $(\{a_1, \cdots, a_k\}, t) \in \mathsf{SubsetSum}$ and $(\{a_1', \cdots, a_l'\}, t') \notin \mathsf{SubsetSum}$:

then $n \overset{\text{def}}{=} 3t \in V(e_1) \subseteq V(e)$, and

- $n - 1 = 3t - 1 \notin V(e_1) \cup V(e_2) = V(e)$, since $n - 1 \equiv 2 \mod 3$ and
  - $\forall m \in V(e_1),\ m \equiv 0 \mod 3$
  - $\forall m \in V(e_2),\ m \equiv 1 \mod 3$
- $n + 1 = 3t + 1 \notin V(e_1) \cup V(e_2) = V(e)$, since $n + 1 \equiv 1 \mod 3$ and
  - $\forall m \in V(e_1),\ m \equiv 0 \mod 3$
  - if $n + 1 \in V(e_2)$, then there exists $J \subseteq [1, l]$ such that

$$3 \sum_{j \in J} a_j' + 3(t - t') + 1 = 3t + 1$$

i.e

$$\sum_{j\in J} a'_j = t'$$

which contradicts $(\{a'_1, \cdots, a'_l\}, t') \notin \mathsf{SubsetSum}$

so $r(w) \in \mathsf{IsolVal}$

If $(\{a_1, \cdots, a_k\}, t) \notin \mathsf{SubsetSum}$ or $(\{a'_1, \cdots, a'_l\}, t') \in \mathsf{SubsetSum}$:

- If $(\{a_1, \cdots, a_k\}, t) \notin \mathsf{SubsetSum}$, then
  - $n \notin V(e_1)$ (subset sum condition)
  - $n = 3t \notin V(e_2)$ (as argued before, due to $n \equiv 0 \mod 3$)

  so $n \notin V(e)$
- If $(\{a'_1, \cdots, a'_l\}, t') \in \mathsf{SubsetSum}$, then
  - $n + 1 = 3t + 1 \in V(e_2)$, since a subset sum sums to t'

in either case, $r(w) \notin \mathsf{IsolVal}$

If $t' > t$

We proceed similarly, with $r(w) \overset{\text{def}}{=} (e, n)$ where:

- $$\begin{aligned} e \overset{\text{def}}{=} & \overbrace{3a_1 \cup 0 + \cdots + 3a_k \cup 0 + 3(t'-t)}^{\overset{\text{def}}{=} e_1} \\ & \cup \underbrace{\left(3a'_1 \cup 0 + \cdots + 3a'_l \cup 0 + 1\right)}_{\overset{\text{def}}{=} e_2} \end{aligned}$$
- $$n \overset{\text{def}}{=} 3t'$$

The proof is perfectly analogous, since the elements of the sets $V(e_1)$ and $V(e_2)$ have still the same value modulo 3.

On the whole, we have shown that

$$\begin{aligned} w \in L_1 \backslash L_2 & \iff (\{a_1, \cdots, a_k\}, t) \in \mathsf{SubsetSum} \text{ and } (\{a'_1, \cdots, a'_l\}, t') \notin \mathsf{SubsetSum} \\ & \iff r(w) \in \mathsf{IsolVal} \end{aligned}$$

Moreover, r runs clearly in logspace, as all we do is comparing t and t', before using pointers to write the NE directly on the output tape.

> $\mathsf{IsolVal}$ is $\nabla\mathrm{NP}$-complete.

3. A more complex reduction

11.

$$S^Z \overset{\text{def}}{=} C^Z \wedge \bigwedge_i C_i^Z \wedge \bigwedge_{i \neq j} D_{i,j}^Z$$

where

- $C^Z \overset{\text{def}}{=} z_1 \vee \cdots \vee z_n$
- $C_i^Z \overset{\text{def}}{=} z_1 \vee \cdots \vee z_{i-1} \vee \neg z_i \vee z_{i+1} \vee \cdots \vee z_n$
- $D_{i,j}^Z \overset{\text{def}}{=} \neg z_i \vee \neg z_j$

If $n = 0$

Then

- $C^Z \overset{\text{def}}{=} \bot$
- $[1, n] = \emptyset$

so that:

$$S^Z \overset{\text{def}}{=} \bot \wedge \underbrace{\bigwedge_{i \in \emptyset} C_i^Z}_{=\top} \wedge \underbrace{\bigwedge_{i \neq j \in \emptyset} D_{i,j}^Z}_{=\top}$$

and the CNF-form of S^Z is:

$$S^Z \overset{\text{def}}{=} \bot$$

Therefore $S^Z \in \mathsf{AlmostSAT}$, since S^Z is unsatisfiable, and $S \backslash \bot$, which is the empty conjunction (that is, $\top$) is satisfiable.

If $n > 0$

S^Z is unsatisfiable

By contradiction, if there existed a valuation v satisfying S^Z:

- as v would satisfy $C^Z \overset{\text{def}}{=} z_1 \vee \cdots \vee z_n$, there would exist $i \in [1, n]$ such that $v(z_i) = \top$
- but then, v satisfying $C_i^Z \overset{\text{def}}{=} z_1 \vee \cdots \vee z_{i-1} \vee \neg z_i \vee z_{i+1} \vee \cdots \vee z_n$ would yield another $j \neq i$ such that $v(z_j) = \top$
- which would contradict v satisfying $D_{i,j}^Z \overset{\text{def}}{=} \neg z_i \vee \neg z_j$

$S^Z \backslash C^Z$ is satisfiable

Setting all variables to $\bot$ then satisfies $\bigwedge_i C_i^Z \wedge \bigwedge_{i \neq j} D_{i,j}^Z = S^Z \backslash C^Z$.

$S^Z \backslash C_k^Z$ is satisfiable

Setting all variables to $\bot$ except z_k (which is set to $\top$) then satisfies $C^Z \wedge \bigwedge_{i \neq k} C_i^Z \wedge \bigwedge_{i \neq j} D_{i,j}^Z = S^Z \backslash C_k^Z$.

$S^Z \backslash D_{k,l}^Z$ is satisfiable

Setting all variables to $\bot$ except z_k and z_l (which are set to $\top$) then satisfies $C^Z \wedge \bigwedge_i C_i^Z \wedge \bigwedge_{\substack{i \neq j \\ i,j \notin \{k,l\}}} D_{i,j}^Z = S^Z \backslash D_{k,l}^Z$.

It has been shown that

> $$S^Z \in \mathsf{AlmostSAT}$$

12.

Let us reduce $\mathsf{AlmostSAT}$ from coSAT.

First, we reduce $\mathsf{co3\text{-}SAT}$ from coSAT: as $\mathsf{3\text{-}SAT}$ is NP-hard, $\mathsf{co3\text{-}SAT}$ is also coNP-hard.

Indeed: for all $L \in \mathrm{coNP}$, there exist a reduction r of $\overline{L} \in \mathrm{NP}$ from $\mathsf{3\text{-}SAT}$, so that

$$\forall w,\ w \in \overline{L} \Longleftrightarrow r(w) \in \mathsf{3\text{-}SAT}$$

that is

$$\forall w,\ w \in L \Longleftrightarrow w \notin \overline{L} \Longleftrightarrow r(w) \notin \mathsf{3\text{-}SAT} \Longleftrightarrow r(w) \in \mathsf{co3\text{-}SAT}$$

so that r is reduction of L from $\mathsf{co3\text{-}SAT}$.

So we want to show

$$\mathsf{coSAT} \preccurlyeq_{\mathrm{L}} \mathsf{co3\text{-}SAT} \preccurlyeq_{\mathrm{L}} \mathsf{AlmostSAT}$$

by reducing $\mathsf{AlmostSAT}$ from $\mathsf{co3\text{-}SAT}$.

Let $\varphi \overset{\text{def}}{=} \bigwedge_i \underbrace{\mathscr{C}_i}_{\overset{\text{def}}{=} l_i^1 \vee l_i^2 \vee l_i^3}$ be a CNF formula.

Firstly, one defines a formula $\widetilde{\varphi}$ out of φ such that $\widetilde{\varphi}$ has no tautological clause, i.e. no clause containing x and $\neg x$ for a variable x: φ and the resulting $\widetilde{\varphi}$ are then equisatisfiable.

This can be done in logspace, as all the clauses are of size smaller (or equal) than 3: one scans through all the clauses, by writing the literals l_i^k of the current examined clause $\mathscr{C}_i$ on the working tape, and one checks if $\mathscr{C}_i$ is tautological: as it happens, $\mathscr{C}_i$ is not considered in the following reduction ("ignored" or "eliminated" in a way).

So from now on, we will work on $\widetilde{\varphi}$, and we can assume, without loss of generality, that $\widetilde{\varphi} = \varphi$, so that φ has no tautological clauses.

Let r be defined as:

$$
\begin{aligned}
r(\varphi) \overset{\text{def}}{=} & \bigwedge_i \overbrace{\bigvee_{j \neq i} z_j \vee \mathscr{C}_i}^{\overset{\text{def}}{=} E_i^Z} \\
& \wedge \bigwedge_{j \neq i} \overbrace{\neg z_i \vee \neg z_j}^{\overset{\text{def}}{=} D_{i,j}^Z} \\
& \wedge \bigwedge_{i,r} \overbrace{\bigvee_{j \neq i} z_j \vee \overline{l_i^r} \vee \neg z_i}^{\overset{\text{def}}{=} C_{i,r}^Z}
\end{aligned}
$$

where

- the z_i are fresh variables
- $\overline{l_i^r}$ is the negation of the literal l_i^r

φ and $r(\varphi)$ are equisatisfiable

if φ is satisfied by a valuation v

then the valuation obtained out of v by setting all the z_i to false satisfies $r(\varphi)$:

- each E_i^Z are satisfied due to $\mathscr{C}_i$ being satisfied
- each $D_{i,j}^Z$ and $C_{i,r}^Z$ are all satisfied as well since all the z_i are set to false

if φ is unsatisfiable

then, by contradiction, let us assume that there exists a valuation v satisfying $r(\varphi)$.

NB: By abuse of notation, we extend v to any clause, so that: if $C = \bigvee_i l_i, \; v(C) \overset{\text{def}}{=} \bigvee_i v(l_i)$

1. For each i: as $v(E_i^Z)$ is true and $v(\mathscr{C}_i)$ is false (φ unsatisfiable): there exists $i_0 \neq i$ such that $v(z_{i_0}) = \top$
2. Then as, for all $j \neq i_0$, $D_{i_0,j}^Z$ is satisfied: for all $j \neq i_0, \; v(z_j) = \perp$
3. So $v(\mathscr{C}_{i_0}) = \top$ is true, since $v(E_{i_0}^Z) = \top$ and for all $j \neq i_0$, $v(z_j) = \perp$
4. But for all r, as $v(C_{i_0,r}^Z) = \top$ and:
   - for all $j \neq i_0$, $v(z_j) = \perp$
   - $v(\neg z_{i_0}) = \perp$

   it follows that $v(\overline{l_{i_0}^r}) = \top$ for all r, so that $v(\mathscr{C}_{i_0}) = \perp$, which contradicts the point 3.

For all i_0, $r(\varphi) \backslash E_{i_0}^Z$ is satisfiable

Let v be a valuation such that

- $v(z_{i_0}) = \top$
- $\forall j \neq i_0, \; v(z_j) = \perp$
- $\forall r, \; v(\overline{l_{i_0}^r}) = \top$
- associates anything ($\top$ or $\perp$) to all the other variables

NB: v is well-defined over the $\overline{l_{i_0}^r}$ because *no clause in φ is tautological*, so that no two literals $\overline{l_{i_0}^r}, \overline{l_{i_0}^{r'}}$ are such that one is a variable and the other the negation thereof.

v satisfies $r(\varphi) \backslash E_{i_0}^Z$:

- for all $i \neq i_0$, for all r, E_i^Z and $C_{i,r}^Z$ are satisfied because of z_{i_0}
- each $D_{i,j}^Z$ is satisfied because z_{i_0} is the only z_k that is set to true by v: all the others are set to false
- for all r, $C_{i_0,r}^Z$ is satisfied because of $\overline{l_{i_0}^r}$

For all i_0, j_0, $r(\varphi) \backslash D_{i_0,j_0}^Z$ is satisfiable

By setting both z_{i_0} and z_{j_0} to $\top$, and all the other z_k to $\perp$, one easliy checks that $r(\varphi) \backslash D_{i_0,j_0}^Z$ is satisfied.

For all i_0, r_0, $r(\varphi) \backslash C_{i_0,r_0}^Z$ is satisfiable

By setting:

- z_{i_0} and $\overline{l_{i_0}^{r_0}}$ to $\top$
- all the other z_k and $\overline{l_{i_0}^r}$ to $\perp$

one checks that $r(\varphi) \backslash C_{i_0,r_0}^Z$ is satisfied:

- for all $i \neq i_0$, for all r, E_i^Z and $C_{i,r}^Z$ are satisfied because of z_{i_0}
- each $D_{i,j}^Z$ is satisfied because z_{i_0} is the only z_k that is set to true: all the others are set to false
- $E_{i_0}^Z$ is satisfied because of $\overline{l_{i_0}^{r_0}}$ satisfying $\mathscr{C}_{i_0}$

On the whole, we have shown that:

$$\varphi \in \textsf{co3-SAT} \Longleftrightarrow r(\varphi) \in \textsf{AlmostSAT}$$

The reduction is logspace since:

- one computes the number of clauses of φ with a counter
- With finite number of pointers: for each clause $\mathscr{C}_i$, one writes the E_i^Z and the $C_{i,r}^Z$ on the output tape, by remembering which is the current z_i with a counter
- then, one writes the $D_{i,j}^Z$ with possibly another counter

Finally, as $\textsf{coSAT} \preccurlyeq_{\mathrm{L}} \textsf{co3-SAT}$ and $\textsf{co3-SAT} \preccurlyeq_{\mathrm{L}} \textsf{AlmostSAT}$:

> $$\textsf{co-SAT} \preccurlyeq_{\mathrm{L}} \textsf{AlmostSAT}$$

13.

We proceed in the same way as the previous question, the only difference being that we add the clause $C^Z \overset{\text{def}}{=} \bigvee_i z_i$ to $r(\varphi)$ (with the same notations as before), so that the new $r(\varphi)$ is defined as:

$$\begin{aligned} r(\varphi) \overset{\text{def}}{=} & \bigwedge_i \overbrace{\bigvee_{j \neq i} z_j \vee \mathscr{C}_i}^{\overset{\text{def}}{=} E_i^Z} \\ & \wedge \bigwedge_{j \neq i} \overbrace{\neg z_i \vee \neg z_j}^{\overset{\text{def}}{=} D_{i,j}^Z} \\ & \wedge \bigwedge_{i,r} \overbrace{\bigvee_{j \neq i} z_j \vee \overline{l_i^r} \vee \neg z_i}^{\overset{\text{def}}{=} C_{i,r}^Z} \\ & \wedge \underbrace{\bigvee_i z_i}_{\overset{\text{def}}{=} C^Z} \end{aligned}$$

Let us checks that

$$\varphi \in \textsf{SAT} \Longleftrightarrow r(\varphi) \in \textsf{AlmostSAT}$$

in a similar fashion:

$r(\varphi)$ is unsatisfiable

By contradiction: if a valuation v satisfies $r(\varphi)$

1. because of C^Z, there exists i_0 such that $v(z_{i_0}) = \top$
2. but then, because of the $D_{i,j}^Z$, all the z_j for $j \neq i_0$ are set to $\bot$…
3. … which means, because of $E_{i_0}^Z$, that $v(\mathscr{C}_{i_0}) = \top$
4. Owing to $C_{i,r}^Z$ for all r: $v(\overline{l_{i_0}^r}) = \top$ (since $v(\neg z_{i0})$ and $v(z_j)$ for $j \neq i_0$ are equal to $\bot$)
5. which yields a contradiction, similarly as before: as $v(\overline{l_{i_0}^r}) = \top$ for all r, $v(\mathscr{C}_{i_0}) = \bot$, which contradicts the point 3.

if φ is satisfiable, $r(\varphi) \backslash E_{i_0}^Z,\ r(\varphi) \backslash E_{i_0}^Z,\ r(\varphi) \backslash D_{i_0,j_0}^Z,\ r(\varphi) \backslash C_{i_0,r_0}^Z \text{ and } r(\varphi) \backslash C^Z$ are satisfiable

For

- $r(\varphi) \backslash E_{i_0}^Z$
- $r(\varphi) \backslash E_{i_0}^Z$
- $r(\varphi) \backslash D_{i_0,j_0}^Z$
- $r(\varphi) \backslash C_{i_0,r_0}^Z$

the proofs go exactly as in previous question, since:

1. we didn't assume any satisfiability property of φ back then, to show these results
2. and in these demonstrations, the clause C^Z (which was not there before) would systematically have been satisfied, since always at least one the z_i was set to true.

For $r(\varphi) \backslash C^Z$: the proof has already been done in the previous question, just after the definition of $r(\varphi)$, when we showed "φ satisfiable implies $r(\varphi)$ satisfiable".

if φ is unsatisfiable $r(\varphi) \backslash C^Z$ is unsatisfiable

Again, the proof has already been done in the previous question, just after the definition of $r(\varphi)$, when we showed "φ unsatisfiable implies $r(\varphi)$ unsatisfiable".

On the whole, we have shown that

$$\varphi \in \mathsf{SAT} \iff r(\varphi) \in \mathsf{AlmostSAT}$$

and this reduction, as in the previous question (the argument remain the same) is still in logspace.

Thus

> $$\mathsf{SAT} \preccurlyeq_{\mathrm{L}} \mathsf{AlmostSAT}$$

14.

If $S \overset{\text{def}}{=} \bigwedge_i C_i, S' \overset{\text{def}}{=} \bigvee_i C_i'$ are CNF formulas, one defines

$$r((S, S')) \overset{\text{def}}{=} \left(\bigwedge_i (C_i \vee x)\right) \wedge \left(\bigwedge_i (C_i' \vee \neg x)\right)$$

where x a fresh variable.

If $(S, S') \in \mathsf{doubleAlmostSAT}$, one easliy checks that:

- $r((S, S'))$ is unsatisfiable, as S and S' are
- $r((S, S'))$ minus any clause is satisfiable:
  - $r((S, S'))\backslash(C_i \vee x)$ is satisfied by the valuation obtained out of the valuation satisfying $S\backslash C_i$ and which sets x to false (so that $\bigwedge_i (C_i' \vee \neg x)$ is satisfied)
  - the other case is symmetric

If S or S' is not in $\mathsf{AlmostSAT}$:

- If S or S' is satisfiable: so is $r((S, S'))$ (by setting x appropriately, as before).
- If $S\backslash C_i$ is unsatisfiable: so is $r((S, S'))\backslash(C_i \vee x)$, since any valuation satisfying $\bigwedge_j (C_j \vee \neg x)$ can be restricted (by forgetting x) into a valuation satisfying $\bigwedge_j C_j = S$
- the other case is symmetric

so $r((S, S'))$ is not in $\mathsf{AlmostSAT}$ either.

We have shown that

$$(S, S') \in \mathsf{doubleAlmostSAT} \iff r((S, S')) \in \mathsf{AlmostSAT}$$

and the reduction is trivially in logspace: one only scans through the clauses and possibly adds one extra fresh variable (one counts (with one counter) the total number of variables to do so).

Thus

> $$\mathsf{doubleAlmostSAT} \preccurlyeq_{\mathrm{L}} \mathsf{AlmostSAT}$$

15.

As $\mathsf{YesNoSAT}$ is $\nabla\mathrm{NP}$-complete (question 8), any language in $\nabla\mathrm{NP}$:

- can be turned (in logspace) into an instance thereof
- which can itself be turned (in logspace) into an intance of $\mathsf{doubleAlmostSAT}$ (questions 12 and 13)
- which can itself be turned (in logspace) into an intance of $\mathsf{AlmostSAT}$ (question 14).

As all these reductions are logspace, we use the property that $\preccurlyeq_{\mathrm{L}}$ is transitive (as seen in class) to conclude that $\mathsf{AlmostSAT}$ is $\nabla\mathrm{NP}$-hard.

As $\mathsf{AlmostSAT} \in \nabla\mathrm{NP}$ (question 5),

> $\mathsf{AlmostSAT}$ is $\nabla\mathrm{NP}$-complete.

16.

We will show the contrapositive.

We have shown at question 9 that $\mathsf{BestClique}$ is $\nabla\mathrm{NP}$-complete (since we proved that it is NP-hard). So as $\mathrm{coNP} \subseteq \nabla\mathrm{NP}$, $\mathsf{BestClique}$ is coNP-hard.

But:

> **Lemma**:
>
> 1. if a coNP-hard language L is in NP, then $\mathrm{coNP} = \mathrm{NP}$
> 2. if an NP-hard language L is in coNP, then $\mathrm{coNP} = \mathrm{NP}$

Proof of 1.:

$\mathrm{coNP} \subseteq \mathrm{NP}$

For any language $L' \in \mathrm{coNP}$: L' can be reduced to L. But as $L \in \mathrm{NP}$, it follows that $L' \in \mathrm{NP}$.

$\mathrm{NP} \subseteq \mathrm{coNP}$

For any language $L' \in \mathrm{NP}$, as $\overline{L'} \in \mathrm{coNP}$, $\overline{L'}$ can be reduced to L by a logspace reduction r.

Thus,

$$\forall w,\ w \in \overline{L'} \Longleftrightarrow r(w) \in L$$

which implies that

$$\forall w,\ w \in L' \Longleftrightarrow w \notin \overline{L'} \Longleftrightarrow r(w) \notin L \Longleftrightarrow r(w) \in \overline{L}$$

That is, L' can be logspace reduced to $\overline{L}$. But as $L \in \mathrm{NP}$, $\overline{L} \in \mathrm{coNP}$, and the result follows.

The proof of 2. is symmetric.

By applying the first point of this lemma to $L = \mathsf{BestClique}$, it follows that $\mathrm{coNP} = \mathrm{NP}$.

17.

Again, we prove the contrapositive.

As $\mathsf{BestClique}$ has been shown (question 9) to be NP-hard, by applying the first point of the previous lemma to $L = \mathsf{BestClique}$, it follows that $\mathrm{coNP} = \mathrm{NP}$.

18.

If $\mathrm{coNP} \neq \mathrm{NP}$, by questions 16 and 17: $\mathsf{BestClique} \notin \mathrm{NP} \cup \mathrm{coNP}$. But by question 9, $\mathsf{BestClique} \in \nabla\mathrm{NP}$. Thus, $\mathsf{BestClique} \in \nabla\mathrm{NP} \setminus (\mathrm{NP} \cup \mathrm{coNP})$, and since $\mathrm{NP} \cup \mathrm{coNP} \subseteq \nabla\mathrm{NP}$ (question 6):

$$\mathrm{NP} \cup \mathrm{coNP} \subsetneq \nabla\mathrm{NP}$$