Homework 3

Name: 方嘉聪 ID: 2200017849

Problem 1.(18 points). Prove that the following language is in **P**.

Horn-statisfiability:

$$\mathrm{CNF}_H = \{\langle\phi\rangle \mid \phi \text{ is a statisfiable Horn formula}\}$$

A Horn clause is a clause with at most one postive literal and any number of negative literals. A Horn formula is a propositional formula formed by conjunctions of Horn clauses. ◀

Answer. 设一个 formula 为 $\phi = \bigwedge_i \left(\bigvee_j v_{i,j}\right)$, 其中 $v_{i,j}$ 可以 p 或 $\neg p$. 考虑如下的多项式时间算法算法判断 ϕ 是否是 Horn 可满足的:

Input: ϕ

Output: ϕ is Horn-satisfiable?

1: **while** ϕ contains some clause **do**
2:     **if** all the clauses currently existing contain negative literals **then**
3:         Set all the negative literals to be false.
4:         **return True**
5:     **else if** there exists an empty clause **then**
6:         **return False**
7:     **else**
8:         at lease one clause contains a positive literal.
9:         Set the positive literal p to be true.
10:         Remove the clause containing p and remove all $\neg p$ in other clauses.
11: **return True**

第 8-10 行中, 如果一个 clause 中有 p, 那么令 $p = 1$, 这个 clause 已满足可以删去, 而 $\neg p = 0$ 不会影响所在 clause 的满足性, 故可以删去. 第 5 行中如果存在 clause 为空, 那么显然不满足. 由于每次循环至少有一个 clause 被删除, 而每次循环是多项式时间的 (遍历常数次整个 formula 即可), 故整个算法是多项式时间的. 所以 $\mathrm{CNF}_H \in \mathbf{P}$. 证毕. ◁

Problem 2.(14 points). Suppose $L_1, L_2 \in \mathbf{NP}$. Then is $L_1 \cap L_2 \in \mathbf{NP}$? What about $L_1 \cup L_2$? ◀

Answer. 考虑通过 **NP** 的两种定义来证明 $L_1 \cap L_2 \in \mathbf{NP}, L_1 \cup L_2 \in \mathbf{NP}$.

(1) **Verifier definition**: 由于 $L_1, L_2 \in \mathbf{NP}$, 不妨设符号集为 Σ. 那么 (对 $i = 1, 2$) 存在多项式时间的函数 $p_i : \mathbb{N} \to \mathbb{N}$, 与多项式时间的 TM M_i 使得

$$L_i = \{x \mid \exists u_i \in \Sigma^{p_i(|x|)}\ s.t.\ M_i(x, u_i) = 1\}$$

那么根据定义有:

$$L_1 \cup L_2 = \{x \mid \exists u \in \widehat{\Sigma}^{(p_1(|x|)+p_2(|x|)+1)}, \text{where } u = u_1 \# u_2, \\ s.t.\ u_1 \in \Sigma^{p_1(|x|)} \wedge u_2 \in \Sigma^{p_2(|x|)} \wedge (M_1(x, u_1) = 1 \vee M_2(x, u_2) = 1)\}$$

其中 $\widehat{\Sigma} := \Sigma \cup \{\#\}$, $\#$ 是一个特殊的分割符. 设 $p: \mathbb{N} \to \mathbb{N}$ 为 $p(|x|) = p_1(|x_1|) + p_2(|x_2|) + 1$, 那么 p 显然是多项式时间的. 类似的, 存在一个多项式时间的 TM M(分别模拟运行 M_1, M_2, 再对结果取或即可), 使得 M 可以判断 u 是否满足条件. 所以 $L_1 \cup L_2 \in \mathbf{NP}$.
类似的我们来证明 $L_1 \cap L_2 \in \mathbf{NP}$. 我们有:

$$\begin{aligned} L_1 \cap L_2 = \{x \mid & \exists u \in \widehat{\Sigma}^{(p_1(|x|)+p_2(|x|)+1)}, \text{where } u = u_1\#u_2, \\ & s.t.\ u_1 \in \Sigma^{p_1(|x|)} \wedge u_2 \in \Sigma^{p_2(|x|)} \wedge (M_1(x, u_1) = 1 \wedge M_2(x, u_2) = 1)\} \end{aligned}$$

同样的, 设 $p: \mathbb{N} \to \mathbb{N}$ 为 $\widehat{p}(|x|) = p_1(|x_1|) + p_2(|x_2|) + 1$, 那么 $\widehat{p}$ 显然是多项式时间的. 类似的, 存在一个多项式时间的 TM M(分别模拟运行 M_1, M_2, 再对结果取与即可), 使得 M 可以判断 u 是否满足条件. 所以 $L_1 \cap L_2 \in \mathbf{NP}$.

(2) **NDTM definition**: 由于 $L_1, L_2 \in \mathbf{NP}$, 不妨设符号集为 Σ. 那么 (对 $i = 1, 2$) 存在一个多项式时间 $T_i(n)$ 的 NDTM N_i 使得 $L(N_i) = L_i$. 考虑如下算法:

Algorithm 2 Verifier for $L_1 \cup L_2$

Input: x(Suppose $|x| = n$)
Output: $x \in L_1 \cup L_2$?
1: Run $N_1(x)$ for $T_1(n)$ steps, output is b_1.
2: Run $N_2(x)$ for $T_2(n)$ steps, output is b_2.
3: **return** $b_1 \vee b_2$

容易发现, 该算法的时间复杂度为 $O(T_1(n) + T_2(n)) = O(T(n))$, 其中 $T(n)$ 是一个多项式时间函数. 所以 $L_1 \cup L_2 \in \mathbf{NP}$. 类似的对于 $L_1 \cap L_2 \in \mathbf{NP}$, 有如下的算法:

Algorithm 3 Verifier for $L_1 \cup L_2$

Input: x(Suppose $|x| = n$)
Output: $x \in L_1 \cup L_2$?
1: Run $N_1(x)$ for $T_1(n)$ steps, output is b_1.
2: Run $N_2(x)$ for $T_2(n)$ steps, output is b_2.
3: **return** $b_1 \wedge b_2$

故 $L_1 \cap L_2 \in \mathbf{NP}$. 证毕.

◁

Problem 3.(12 points). Prove that

$$L = \{w \mid w \text{ is a binary representation of a prime number}\} \in \mathbf{NP}$$

Tips: A natural number n is a prime number if and only if for any prime factor p of $n-1$, there exists an $a \in \{2, \cdots, n-1\}$ such that $a^{n-1} = 1 \bmod n$ and $a^{(n-1)/p} \neq 1 \bmod n$. ◀

Answer. 注: 下面是一个错误的解答!!!

对于 $x = n \in \mathbb{Z}$, 那么 $|x| = \log n$. 考虑 certificate 为 a 和不超过 $n-1$ 的所有素因子的二进制序列 (两两之间用特殊的分隔符 $\#$ 分开), 记为 u. 注意到素因子的个数不超过 $\log n$, 那么:

$$|u| = O((\log n + 1)|x|) = O(\log^2 n) = \mathrm{Poly}(\log n)$$

这样的证书是合法的. 考虑如下的 verifier 算法 (构造图灵机 M):

Algorithm 4 **Verifier for** L

如下构造图灵机 M:
Input: $x = n$, certificate u
  按照特殊的分隔符 # 将 u 分为 a 与若干个素因子 $p_1, p_2, \cdots, p_k$.
  对于 u 中的每个素因子 p, 验证 $a^{n-1} = 1 \bmod n$ 及 $a^{(n-1)/p} \neq 1 \bmod n$.
  如果上述条件都满足, 则 M **accept**, 否则 M **reject**.

对 $\forall p$, 验证 $a^{n-1} = 1 \bmod n$ 以及 $a^{(n-1)/p} \neq 1 \bmod n$ 可以用快速幂实现, 时间复杂度是 $O(\log n)$. 所以上述的算法是 $O(\log n \cdot \log n) = \mathrm{Poly}(\log n)$ 的, 且满足 $\forall x \in L \iff \exists u,\ s.t.\ M(x, u) = 1$. 故 $L \in \mathbf{NP}$. 证毕.

正确的做法应该使用递归的证书, 具体见下:

考虑 n 对应的证书为 $[a, c(n)]$, 其中

$$c(n) := \{[p_1, a_1, c(p_1)]\#[p_2, a_2, c(p_2)]\#\cdots\#[p_k, a_k, c(p_k)]\}$$

其中 p_i 为素因子, a_i 为 hint 用以判断的指数, $c(p_i)$ 为 p_i 对应的证书.

考虑如下定义的图灵机 M:

Input: $x = n$, certificate $[a, c(n)]$
Output: $x \in L$?
遍历进行分割并验证证书形式, 反复使用除法验证 $n-1$ 的所有素因子是否恰为 $p_1, p_2, \cdots, p_k$.
递归调用 M, 输入为 $[p_i, a_i, c(p_i)]$
对于每个 p_i, 验证 $a^{n-1} = 1 \bmod n$ 以及 $a^{(n-1)/p_i} \neq 1 \bmod n$.
如果上述条件都满足, 则 M **accept**, 否则 M **reject**.

下面证明一下 M 的运行时间是 $\mathrm{Poly}(\log n)$ 的.

- 考虑 $n \geq 2$, 设素因数分解为 (考虑以 2 为底)

$$n = \prod_{i=1}^{k} p_k^{m_k} \implies \log n = \sum_{i=1}^{k} m_k \log p_k \geq \sum_{i=1}^{k} \log p_k \geq k \implies k \leq \log n$$

  即素因子的个数是 $O(\log n)$, 进而 $|[p_1, p_2, \cdots, p_k]| = O(\log^2 n)$. 下面证明总的证书的长度 $L(n)$ 是 $\mathrm{Poly}(\log n)$ 的. 首先我们有

$$L(n) \leq c \log^2 n + \sum_{p|n-1} L(p)$$

  我们归纳的证明 $L(n) = O(\log^3 n)$, 当 $n = 2$ 时, $L(2) = O(1)$, 假设对于 $n \leq k-1$ 成立, 那么对于 $n = k$(注意 k 是奇数):

$$\begin{aligned} L(k) &\leq c \log^2 k + \sum_{p|k-1} L(p) \leq c \log^2 k + \sum_{p|k-1, p>2} O(\log^3 p) + O(1) \\ &\leq c \log^2 k + O(\log^3 k) \leq c' \log^3 k \end{aligned}$$

  故 $L(n) = O(\log^3 n)$.

- 对于 M 的运行时间, 分为验证证书合法、检验素因子、递归调用、检查 a_i 条件, 类似的可以用归纳法证明 M 的运行时间是 $O(\log^4 n)$ 的.

$\lhd$

Problem 4.(14 points). Let **HALT** be the Halting language. Show that **HALT** is **NP**-hard. Is it **NP**-complete? ◀

Answer. 由于 **HALT** 是不可计算的, 所以 **HALT** 不是 **NP**-complete 的. 下面我们来证明 **HALT** 是 **NP**-hard 的. 只需证明 $\forall A \in \mathbf{NP}, A \leq_p \mathbf{HALT}$, 即存在多项式时间的函数 $f(\cdot)$ 使得 $\forall x \in A$ 等价于 $f(x) = \langle M, x \rangle \in \mathbf{HALT}$:

设存在 NDTM N 使得 $L(N) = A$, 考虑如下的算法 (规约):

如下构造图灵机 M:
  Input: x
  Procedure:
    模拟运行 $N(x)$, 看是否被接受.
    如果 accept, 则 M **halt**, 否则 M **loop forever**.

注意到 N 是多项式时间的, 虽然 M 不是多项式时间的 (可能 loop forever), 但上述的算法是一个多项式时间的规约 (给定 x, 生成 $\langle M, x \rangle$ 是多项式时间的), 且满足 $\forall x \in A \iff f(x) = \langle M, x \rangle \in \mathbf{HALT}$. 故对 $\forall A \in L, A \leq_p \mathbf{HALT}$, 即 **HALT** 是 **NP**-hard 的. 证毕. $\lhd$

Problem 5.(14 points). Show that, if **NP** = **P**, then every language $A \in \mathbf{P}$, except $A = \varnothing$ and $A = \Sigma^*$, is **NP**-complete. ◀

Answer. 首先证明 $\forall A \neq \varnothing$ or Σ^*, $A \in \mathbf{NP}$-complete, 即需要证明对于 $\forall B \in \mathbf{NP}$,

$$\forall \text{ string } w \in B \iff f(w) \in A \text{ where f}(\cdot) \text{ is a poly-time computable function.}$$

由于 $A \neq \varnothing$ or Σ^*, 那么 $\exists$ string x, y $s.t.$ $x \in A \wedge y \notin A$. 那么对 $\forall s \in \Sigma^*$, 由于 $\mathbf{P} = \mathbf{NP}$, 存在一个多项式时间的确定性图灵机 M 可以判定 x 是否属于 **NP**, 考虑如下的算法:

如下构造 $f(\cdot): \Sigma^* \to \Sigma^*$:
  Input: s
  如果 $M(s) = 1$, 则返回 x, 否则返回 y.

那么显然 $f(\cdot)$ 是多项式时间的, 且满足 $\forall s \in \Sigma^*, s \in B \iff f(s) \in A$. 所以 A 是 **NP**-complete 的.

下面证明 $A = \varnothing$ or Σ^* 不是 **NP**-complete 的.

对于 $\forall B \in \mathbf{NP}$, 当 $A = \varnothing$ 时, $\forall x \in \Sigma^*$, 对任意的多项式时间的函数 $f: \Sigma^* \to \Sigma^*$, $f(x) \notin A$ 恒成立, 所以不存在满足条件的规约 $f(\cdot)$ 使得 $B \leq_p A$, 即 $\varnothing$ 不是 **NP**-complete 的.
当 $A = \Sigma^*$ 时, 无论 $x \in \Sigma^*$ 是否属于 B, $f(x) \in A$ 恒成立, 所以不存在满足条件的规约 $f(\cdot)$ 使得 $B \leq_p A$, 即 Σ^* 不是 **NP**-complete 的. 证毕. $\lhd$

Problem 6.(28 points). Let ϕ be a 3-CNF. An $\neq$-assignment to the variables of ϕ is one where each clause contains two literals with **unequal truth values**. In other words, an $\neq$-assignment satisfies ϕ without assigning three true literals in any clauses.

a.) Show that the negation of any $\neq$-assignment to ϕ is also an $\neq$-assignment.

b.) let $\neq$-SAT be the collection of 3CNFs that have an $\neq$-assignment. Show that we obtain a polynomial-time reduction from 3SAT to $\neq$-SAT by replacing each clause

$$c_i = (y_1 \vee y_2 \vee y_3)$$

with the two clauses

$$(y_1 \vee y_2 \vee z_i) \quad \text{and} \quad (\overline{z}_i \vee y_3 \vee b)$$

where z_i is a new variable for each clause c_i and b is a single addtional new variable.

c.) Conclude that $\neq$-SAT is **NP**-complete.

◀

Answer. a.) 一个 ϕ 的 $\neq$-assignment, 使得每个 clause 中有两个 literal 的真值不相等. 考虑这样的赋值的否定, 则每个 clause 中仍有两个 literal 的真值不相等 (各自取反后仍不相等), 所以否定的 $\neq$-assignment 仍然是 $\neq$-assignment. 证毕.

b.) 设 $\phi \in$ 3-SAT, 考虑一个可满足 ϕ 的赋值, 那么所有 clause $c_i = (y_1 \vee y_2 \vee y_3)$, $y_j (j = 1, 2, 3)$ 不能全为 0. 显然替换是多项式时间的, 且得到的是一个 3CNFs(记为 ϕ'), 分类讨论证明 $\phi' \in \neq$-SAT:

y_1	y_2	y_3	z_i	$\overline{z}_i$	b	$y_1 \vee y_2 \vee z_i$	$\overline{z}_i \vee y_3 \vee b$
1	1	1	0	1	0	1	1
1	1	0	0	1	0	1	1
1	0	1	1	0	0	1	1
1	0	0	0	1	0	1	1
0	1	1	1	0	0	1	1
0	1	0	0	1	0	1	1
0	0	1	1	0	0	1	1

注意到对 $y_j (j = 1, 2, 3)$ 的所有可能情况, 按照上表来确定 z_i, b 的赋值, 这样得到的新赋值首先满足了 ϕ' 的每个 clause, 其次每个 clause 中有两个 literal 的真值不相等. 所以 $\phi' \in \neq$-SAT. 证毕.

c.) 由上一问知 3-SAT $\leq_p \neq$-SAT. 注意到类似 3-SAT 的证明, 给定一个 formula, 可以将其 $\neq$-assignment 作为证书, 通过多项式时间的算法来判定是否是可满足的, 故 $\neq$-SAT $\in$ **NP**, 由于 $\neq$-SAT 是 **NP**-hard 的, 所以 $\neq$-SAT 是 **NP**-complete 的. 证毕.

◁