Vous pouvez rédiger votre copie en français ou en anglais.

When giving reductions or algorithms, explain the algorithm at a high level of abstraction but do not forget corner cases or use vague notation. When arguing the correctness of an algorithm, make a precise claim about your construction before proving said claim. In the proof, you may abstain from proving trivial observations only if you do not miss the harder parts of the proof.

Exercise 1 : Time linear in machine size

We choose some way of describing nondeterministic Turing machines $M, M'$, etc. via some encoding $\lfloor M \rfloor, \lfloor M' \rfloor, \ldots$ over some alphabet $A$. Such an encoding just lists the control states of machine $M$, its alphabet, its transition rules, etc., in some natural way. In particular, it is easy (i.e., logspace computable) to decide if a given string $x \in A^*$ is the encoding of some machine. The length $|\lfloor M \rfloor|$ is the “size” of the machine.

The problem $\textsc{LINEARHALT}$ asks, given $\lfloor M \rfloor$ of size $n$, whether $M$ accepts the empty string in time at most $n$.

1. Show that $\textsc{LINEARHALT}$ is $\textsc{NP}$-complete.

Exercise 2 : Counting complexity

We recall that a parsimonious reduction between two counting problems $F : A^* \to \mathbb{Z}$ and $G : B^* \to \mathbb{Z}$, is a logspace-computable mapping $r : A^* \to B^*$ such that $F(x) = G(r(x))$ for all $x \in A^*$. We use $\leq_{\text{par}}$ to denotes reducibility via parsimonious reductions.

We also recall that $\#\textsc{SAT}$ is the problem, given a set of boolean variables $X = \{x_1, \ldots, x_n\}$ and a boolean formula $\phi$ over $X$ (not necessarily in clausal form), to compute the number of valuations of $X$ that satisfy $\phi$. We write $\#\textsc{3SAT}$ for $\#\textsc{SAT}$ restricted to formulas in clausal form where each clause has at most three literals.

$\textsc{CIRCUITSAT}$ is the satisfiability problem for circuits like the following example.

A circuit $C$ is an acyclic directed graph with input gates $x_1, \ldots, x_m$ and nand-gates $g_1, \ldots, g_{\ell}$. The input gates have in-degree 0 and the nand-gates can have any number of inputs. Given a valuation $v : \{x_1, \ldots, x_m\} \to \{\top, \bot\}$ of the input gates, the boolean value (written $v(g)_C$) of any gate $g$ in $C$ is determined in the usual way. In the above example, the value of $g_3$ under $v$ is $\neg (v(x_1) \land \neg v(x_2) \land \neg v(x_4))$, i.e., $\neg (v(x_1) \land v(x_2) \land v(x_4))$, and the value of $g_4$ is $\neg \bigwedge \emptyset$, i.e., $\bot$.

Formally, $\textsc{CIRCUITSAT}$ asks, given a circuit $C$ and a designated output gate $g$, whether there is a valuation with $v(g)_C = \top$, and $\#\textsc{CIRCUITSAT}$ is the counting version, asking how many valuations yield $v(g)_C = \top$. 

---

Examen Partiel pour Advanced Complexity (M1)

Nov. 9th, 2022. Durée: 2h
2. Show that \#CIRCUITSAT, \#SAT and \#3SAT are equivalent under parsimonious reductions.

Recall that, for a square matrix \( M \), its \textit{permanent} \( P(M) \) is defined as \( P(M) \defeq \sum_{\sigma \in \text{Sym}(n)} \prod_{i=1}^{n} M_{i,\sigma(i)} \) where \( \text{Sym}(n) \) is the group of all permutations of the set \( \{1, 2, \ldots, n\} \).

With \( \text{PERM}_{0,1} \) we denote the problem of computing the permanent of a square matrix whose entries are all among 0 and 1.

3. Give a reduction showing that \( \text{PERM}_{0,1} \leq_{\text{par}} \#\text{SAT} \).

\textbf{Exercise 3 : SPARSE languages.}

We fix an alphabet \( \Sigma \) with at least two letters. \( \Sigma^n \) is the set of words of length \( n \), and \( \#E \) denotes the number of elements in the finite set \( E \). For simplicity, we shall assume that \( \Sigma = \{0, 1\} \), although this is not strictly necessary.

A language \( L \subseteq \Sigma^* \) is \textit{sparse} if and only if there is a polynomial \( p \) such that \( \#(L \cap \Sigma^n) \leq p(n) \) for all \( n \in \mathbb{N} \). One also say that \( L \) “has polynomial density”. Let \( \text{SPARSE} \) be the class of all sparse languages.

4. For \( k \in \mathbb{N} \), we define \( L_k = \{ u \in \{0, 1\}^* : \mid u \mid_1 = k \} \), where \( \mid u \mid \) is the length of \( u \) and \( \mid u \mid_1 \) is the number of times the letter 1 occurs in \( u \). Show that \( L_k \in \text{SPARSE} \cap L \) for any \( k \in \mathbb{N} \).

5. Show that \( \text{SPARSE} \) contains some undecidable languages (Indication : Only look for very simple examples).

6. Given \( L, L_1, L_2 \in \text{NL} \cap \text{SPARSE} \), do we have :
   (1) \( L_1 \cdot L_2 \in \text{NL} \)? (2) \( L_1 \cdot L_2 \in \text{coNL} \)? (3) \( L_1 \cdot L_2 \in \text{SPARSE} \)?
   (4) \( L^* \in \text{NL} \)? (5) \( L^* \in \text{coNL} \)? (6) \( L^* \in \text{SPARSE} \)?

   Recall that \( L_1 \cdot L_2 = \{ uv : u \in L_1, v \in L_2 \} \) and that \( L^* = \{ \epsilon \} \cup L \cup L \cdot L \cup L \cdot L \cdot L \cup \cdots \)

   We say that a language belongs to \( \text{UNARY} \) if it is included in \( \{1\}^* \).

7. Show that \( \text{UNARY} \subseteq \text{SPARSE} \).

8. Show that if \( \text{SPACE}(2^{o(n)}) = \text{TIME}(2^{o(n)}) \), then \( \text{PSPACE} \cap \text{UNARY} \subseteq \text{P} \) and \( \text{NPSPACE} \cap \text{UNARY} \subseteq \text{P} \).

We write \( L \leq_P L' \) when \( L \) is polynomial-time reducible to \( L' \).

9. Show that, if \( \text{P} = \text{NP} \), then \( \text{SAT} \) is polynomial-time reducible to some non-empty sparse language.