Exercise XI, Theory of Computation 2025

These exercises are for your own benefit. Feel free to collaborate and share your answers with other students. Solve as many problems as you can and ask for help if you get stuck for too long.

These problems are taken from various sources at EPFL and on the Internet, too numerous to cite individually.

More Problems on Complexity Classes

1 Define $\mathbf{coNP} = \{L : \overline{L} \in \mathbf{NP}\}$ as the class of languages whose complements are in **NP**. Show that if any **NP**-complete problem lies in **coNP**, then $\mathbf{NP} = \mathbf{coNP}$.

Solution: Suppose $L \in \mathbf{coNP}$ is **NP**-complete. We only show that $\mathbf{NP} \subseteq \mathbf{coNP}$ as the other inclusion, $\mathbf{coNP} \subseteq \mathbf{NP}$, is analogous. Let $L' \in \mathbf{NP}$. Since L is **NP**-complete, we have $L' \leq_p L$ or equivalently $\overline{L'} \leq_p \overline{L}$. But $\overline{L} \in \mathbf{NP}$ so we get that $\overline{L'} \in \mathbf{NP}$. This is equivalent to $L' \in \mathbf{coNP}$ as desired.

2 Denote by **RE** the class of recognisable languages. We say that a language L is **RE**-*complete* iff $L \in \mathbf{RE}$ and for every $L' \in \mathbf{RE}$ we have $L' \leq_m L$. Show that HALT is **RE**-complete.

Solution: We have seen that $A_{TM} \leq_m \text{HALT}$, so it suffices to show instead that A_{TM} is **RE**-complete, which is what we will do. We know that $A_{TM} \in \mathbf{RE}$ so it remains to show that $L \leq_m A_{TM}$ for every $L \in \mathbf{RE}$. Let $L \in \mathbf{RE}$ be arbitrary. Then there is some Turing machine M that recognises L, that is, $w \in L \iff M$ accepts w. We define a reduction $L \leq_m A_{TM}$ by $f(w) = \langle M, w \rangle$, which is clearly computable. Now $w \in L \iff M$ accepts $w \iff \langle M, w \rangle \in A_{TM} \iff f(w) \in A_{TM}$.

Problems on Circuit Complexity

3 Complete the proof of Circuit-SAT $\leq_p$ SAT from the lecture by finding an equivalent CNF formula for each of the three logical predicates

$$y \leftrightarrow (x \vee z), \quad y \leftrightarrow (x \wedge z) \quad \text{and} \quad y \leftrightarrow \neg x.$$

Solution: One can find an equivalent CNF formula by applying standard boolean algebra identities. We provide the details for the first item only.

$$\begin{aligned} y \leftrightarrow (x \vee z) &= (y \to (x \vee z)) \wedge ((x \vee z) \to y) \\ &= (\neg y \vee x \vee z) \wedge (\neg(x \vee z) \vee y) \\ &= (\neg y \vee x \vee z) \wedge (\neg x \wedge \neg z) \vee y) \\ &= (\neg y \vee x \vee z) \wedge (\neg x \vee y) \wedge (\neg z \vee y) \end{aligned}$$

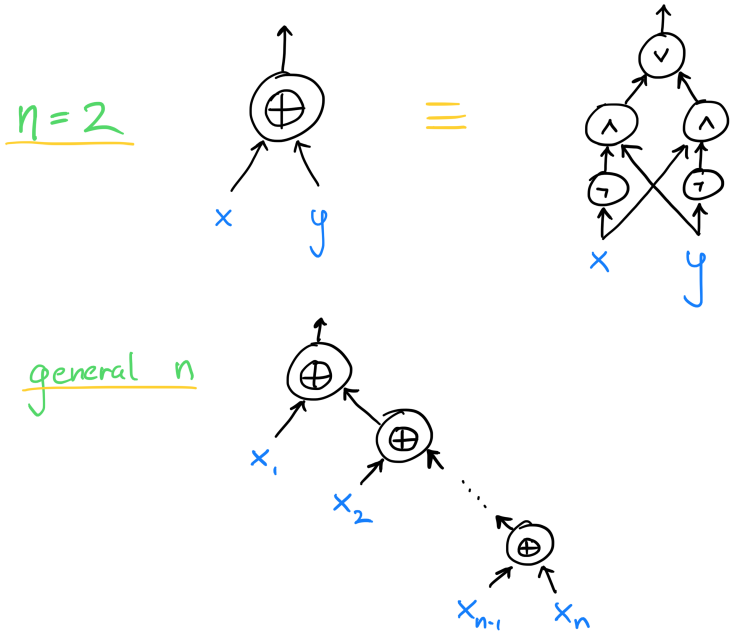
$$y \leftrightarrow (x \wedge z) = (\neg y \vee x) \wedge (\neg z \vee z) \wedge (\neg x \vee \neg z \vee y)$$

$$y \leftrightarrow \neg x = (\neg x \vee \neg y) \wedge (x \vee y)$$

4 The function $\textsc{Xor}_n \colon \{0,1\}^n \to \{0,1\}$ outputs 1 iff the number of 1-bits in the input is odd. Show that $\textsc{Xor}_n$ can be computed with a boolean circuit (gates $\vee$, $\wedge$, $\neg$) of size $O(n)$.

Hint: *Construct a circuit for $n = 2$ and then use many copies of that circuit for general n.*

Solution: Proof by picture:



5 Let φ be any DNF formula over n variables that computes $\textsc{Xor}_n$. Recall that $\varphi = T_1 \vee \cdots \vee T_m$ where each T_j is a *term*, that is, a conjunction of literals.

5a* Show that any term T_j either contains n distinct variables or is *contradictory*, meaning that it contains x_i and $\overline{x}_i$ for some variable x_i.

Hint: *Use the fact that the value of $\textsc{Xor}_n$ is flipped if we flip the value of any x_i.*

Solution: Suppose T_j is not contradictory. Then it accepts some input $x \in \{0,1\}^n$ so that $T_j(x) = 1$ and hence $\varphi(x) = \textsc{Xor}_n(x) = 1$. Suppose for the sake of contradiction that T_i does not contain all n variables. Without loss of generality, assume x_i is not included. Consider $x' \in \{0,1\}^n$ which is the same as x but with the value of variable x_i flipped. We still have that $T_j(x') = 1$ and hence $\varphi(x') = \textsc{Xor}_n(x') = 1$. But this is a contradiction since x and x' have different parities. Hence we conclude that T_j contains all n variables.

5b Thus, show that φ contains at least 2^{n-1} terms.

Solution: Problem 5a implies that each term of φ can accept at most one input. But there are 2^{n-1} many inputs x with $\textsc{Xor}_n(x) = 1$ and hence φ must contain one term for each such x.

Note that problems 3–4 together imply that circuits can be much more expressive than CNFs.