Problem 1: Reductions

Consider the following languages:

- \text{formulaSAT} := \{f : f \text{ is a satisfiable Boolean formula}\}.
- \text{SAT} := \{f : f \text{ is a satisfiable CNF formula}\}.
- \text{CLIQUE} := \{(G, k) : G \text{ contains a } k\text{-clique}\}. A graph \(G\) is said to contain a \(k\)-clique if there is a set of \(k\) vertices in \(G\) such that each of these vertices has an edge to each other vertex (a complete subgraph of size \(k\)).
- \text{INDSET} as described in the lecture (independent set problem; party problem).
- \text{co-TAUTO} := \{f : f \text{ is not a tautology}\}. A tautology is a Boolean formula that is true for any assignments of truth-values to the variables.

For each pair \(A, B \in \{\text{formulaSAT}, \text{SAT}, \text{CLIQUE}, \text{co-TAUTO}, \text{INDSET}\}\), show that \(A \leq_p B\). (I.e., that \(A\) is polynomial-time Karp reducible to \(B\).)

Note: None of these reductions need an elaborate construction like we did in the practice for showing that \(\text{SAT} \leq_p \text{INDSET}\). Each proof should just be a few lines. You are allowed to use facts we already showed, e.g., \(\text{SAT} \leq_p \text{INDSET}\) and \(\text{SAT}\) is \(\text{NP}\)-complete.

Problem 2: NP with unbounded certificates

An important condition in the definition of \(\text{NP}\) is that the certificate has polynomially-bounded length. Without that condition, the definition would look as follows (the following is not an established class!):

\[\text{Definition 1 (NP with unbounded witnesses)}\] A language \(L\) is in \(\text{hugeNP}\) iff there exists a polynomial-time Turing machine \(M\) such that for all \(x\) it holds that:

\[x \in L \iff \exists u \in \{0, 1\}^* : M(x, u) = 1.\]

Note: \(u\) is not restricted in its length in this definition.

Show that \(\text{HALT} \in \text{hugeNP}\) where \(\text{HALT}\) is the Halting Problem.\(^2\)

Hint: A certificate for \((m, \alpha) \in \text{HALT}\) could be the string \(1^n\) where \(n\) is the number of steps that \(M_m(\alpha)\) runs. The notation \(1^n\) means a bitstring consisting of \(n\) ones (e.g., \(1^{10} = 1111111111\)). So you just need to construct the TM \(M\) from Definition 1 and explain why it is polynomial-time.

---

\(^1\)We call \(M\) polynomial-time iff there exists a polynomial-time \(p\) such that for all \(x, u\), the running time of \(M(x, u)\) is bounded by \(p(|x| + |u|)\).

\(^2\)That is, \(\text{HALT} = \{(m, \alpha) : M_m(\alpha) \text{ terminates}\}\) where \(M_m\) is the TM with description \(n\).