Homework 9: Complexity Theory
Due: December 7, 2023

Problem 1. Consider the language:

\[ k\text{MSAT} = \{ \phi \mid \phi \text{ is a Boolean formula with at least } k \text{ satisfying assignments} \} \]

For fixed \( k \), show that \( k\text{MSAT} \) is NP-complete.

Solution.

- \( k\text{MSAT} \in \text{NP} \): We’ll construct a non-deterministic decider for \( k\text{MSAT} \). Given an input Boolean formula \( \phi \) with \( n \) variables and \( m \geq n \) literals:
  1. Non-deterministically pick \( k \) possible disjoint truth value assignments to the variables in \( \phi \).
  2. For a given branch, if the \( k \) selected assignments \( \phi \) evaluates to true then accept, otherwise reject

  - Correctness: If \( \phi \) has \( k \) satisfying assignments then there will be one non-deterministic execution branch that selects these \( k \), and the algorithm will correctly identify them as satisfying. If \( \phi \) has no more than \( k - 1 \) satisfying assignment, in all possible execution branches at least one assignment will not be a satisfiable assignment for \( \phi \) and it will be rejected as being \( k \)-satisfiable.

  - Running time: In each nondeterministic branch, the algorithm evaluates \( \phi \) for each of the \( k \) nondeterministically selected truth assignments. Each evaluation can be computed in polynomial time with respect to \( m \). So overall the algorithm has non-deterministic polynomial time \( kO(m) = O(m) \).

- \( k\text{MSAT} \) is NP-hard: we set up a polynomial time reduction from SAT, which we already know to be NP-hard. In particular, we construct a new Boolean expression \( \phi' = (\phi) \land (y_1 \lor y_2 \lor \ldots \lor / y_k) \), where all the \( y_i \)’s are new Boolean variable that does not appear in \( \phi \).

  - Correctness: we need to prove that \( \phi' \) has \( k \) satisfying assignments iff \( \phi \) is satisfiable. We break down the two directions:
    * \( \phi \) is satisfiable implies \( \phi' \) is \( k \)-satisfiable: \( \phi' \) is satisfiable by setting all the variables except \( y_1, y_2, \ldots, y_k \) as in the satisfying assignment for \( \phi \) and by setting setting each \( y_i \) to true or to false.
    * \( \phi' \) is \( k \)-satisfiable implies \( \phi \) is satisfiable: Clearly any satisfying assignment for \( \phi' \) is also satisfying \( \phi \) ignoring \( y_1, y_2, \ldots, y_k \).

  - Running time: We obtain \( \phi' \) by modifying it by adding just \( k = O(1) \) variables. Hence the reduction requires constant time.

\[ \square \]
Problem 2. Consider the languages:

\[ \text{BF}_k = \{ \phi \mid \phi \text{ is a satisfiable Boolean formula in CNF and each variable appears at most } k \text{ times} \} \]

Show that BF$_3$ is NP-complete.

Solution.

- The problem of deciding whether any \( \phi \in \text{BF}_3 \) is satisfiable is in \( \text{NP} \): We can use the same nondeterministic solver we showed for SAT as any \( \phi \in \text{BF}_3 \) is a Boolean formula.

- We show that 3SAT \( \leq_p \) BF$_3$. First an example: \( \phi = (p \lor q) \land (p \lor r) \land (\lnot p \lor s) \land (p \lor t) \). This formula is not in 3CNF, but the general case will be treated below. This formula is replaced by \( \phi' = (p \lor q) \land (\lnot p \lor p_1) \land (p_1 \lor r) \land (\lnot p_1 \lor p_2) \land (\lnot p_2 \lor s) \land (\lnot p_2 \lor p_3) \land (p_3 \lor t) \land (\lnot p_3 \lor p) \). Note that the \( (\lnot p \lor p_1), (\lnot p_1 \lor p_2), (\lnot p_2 \lor p_3), (\lnot p_3 \lor p) \) are expressing that \( p, p_1, p_2, p_3 \) are all equivalent. Hence in \( \phi' \) \( p_1 \lor r \) is satisfiable iff only if \( (p \lor r) \), is satisfies in \( \phi \). Thus, \( \phi \) is satisfiable iff \( \phi' \) is, but in \( \phi' \) every variable occurs at most 3 times (I.e., \( \phi' \in \text{BF}_3 \)).

In the general case, we define a reduction \( f \) from 3SAT to BF$_3$ as follows: Given a Boolean formula \( \phi \)

\[- \text{ While there exists a Boolean variable that occurs more than 3 times do:} \]

\[- \quad 1. \text{Pick the first variable that occurs more than 3 times from the left. Suppose it is } p \text{ and that it occurs at } n \text{ places } (x_1 \lor \phi_1) \land \ldots \land (x_n \lor \phi_n). \]

\[- \quad 2. \text{For each occurrence of } p \text{ create a new Boolean variable not used before called } p_i. \]

\[- \quad 3. \text{Remove all the } (x_1 \lor \phi_1) \land \ldots \land (x_n \lor \phi_n) \text{ clauses and replace them with } (p_1 \lor \phi_1) \land (\lnot p_1 \lor p_2) \land (p_2 \lor \phi_2) \land (\lnot p_2 \lor p_3) \ldots \land (p_n \lor \phi_n) \land (\lnot p_n \lor p_1). \]

\[- \text{ return } \phi \]

**Correctness:** Clearly \( f(\phi) \in \text{BF}_3 \). By the previous considerations we immediately also have that \( \phi \) is satisfiable iff \( f(\phi) \) is.

**Worst case running time analysis:** Assume \( \phi \) has at most \( n \) Boolean variables and \( m \geq n \) literals. In each iteration of the cycle one variable that appears more than three times in \( \phi \) is removed. The variables added to \( \phi \) in each iteration are not repeated more than 3 times by construction. Hence there are at most \( n \) iterations of the cycle. In each iteration of the cycle, we remove the at most \( m \) occurrences of the variable. In the worst case each occurrence occurs in a distinct clause. For each clause in which the variable occurs, we (a) replace the variables with a new one and we created a new clause in the form \( (\lnot x_i \lor x_{i+1}) \). Hence, in each iteration of the cycle, we introcuce at most \( m \) new variables and \( O(m) \) new literals.

We can conclude that the reduction requires at most \( O(nm) \) operations, and, hence, is a polynomial time reduction.