# Homework 9

## Solution Key

This homework must be typed in LaTeX and handed in via Gradescope.

Please ensure that your solutions are complete, concise, and communicated clearly. Use full sentences and plan your presentation before you write. Except in the rare cases where it is indicated otherwise, consider every problem as asking you to prove your result.

### Problem 1

In the "$k$-Multiple Boolean Satisfiability Problem" $kMSAT$, given a Boolean formula $\phi$ with $m$ variables and $n \geq m$ literals, we want to decide whether it has at least $k$ satisfying assignments, where $k = O(1)$. (I.e., at least $k$ distinct truth-value assignments to its variable for which $\phi$ is satisfied.) Prove that $kMSAT$ is NP-complete.

**Solution**: To prove that $k$-SAT is NP complete we need to show that is in NP and that it is in NP-hard.

- $k - SAT \in NP$: we show a non-deterministic solver for $k$-SAT. Given an input Boolean formula $\phi$ with $n$ variables and $m \geq n$ literals:
  1. Nondeterministicly pick $k$ possible truth value assignments to the variables in $\phi$.
  2. If for all selected assignments $\phi$ evaluates to true, then conclude that $\phi$ has at least two satisfying assignments
  3. Otherwise, reject

  – **Correctness:** If $\phi$ has two satisfying assignments then those two will be selected in one of the non-deterministic execution branches, 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 most one satisfying assignment will be non-deterministically chosen and in the algorithm does reject $\phi$ as being $k$-satisfiable.

  – **Running time:** In each nondeterministic branch, the algorithm evaluates $\phi$ for each of the two 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 $kO(m) = O(m)$.

- $k$-SAT 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 $\phi' = (\phi) \wedge (y_1 \vee y_2 \vee \ldots \vee \cancel{y}_k)$, 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 assigments 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 assigment 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.

**Problem 2**

Let $BF_k$ denote the set of Boolean formulas in Conjunctive Normal Form such that each variable appears in at most $k$ places (i.e., in at most $k$ literals).

1. Show that the problem of deciding whether a Boolean Formula in $BF_2$ is satisfiable is in $P$.
2. Show that the problem of deciding whether a Boolean Formula in $BF_3$ is satisfiable is $NP-Complete$.

**Solution Part a**: Let $\phi \in BF_2$. Since each variable $x_i$ can occur in at most two literals we must be in one of the following cases:

1. $x_i$ appears only once either as $x_i$ or $\neg x_i$: The we can set $x_i$ (or $\neg x_i$) as true, and remove all the clause in which it appears. We can do so as the assignment of $x_i$ is guarantee to satisfy the clause and not interfere with the satisfiability of other clauses.
2. $x_i$ appears twice but in both cases it is either $x_i$ or $\neg x_i$: Following considerations similar to those in the case above we can remove all the at most two clauses in which it it appears. Setting the value of $x_i$ to TRUE (or FALSE) will automatically make the corresponding clauses satisfied with no risk of contradiction.
3. $x_i$ and $\neg x_i$ appear in the same clause: we can remove the clause as it is a tautology.
4. $x_i$ and $\neg x_i$ appear in $\phi$ into two different clauses: Suppose we have $\phi = (x_i \vee \phi_1) \wedge (\neg x_i \vee \phi_2) \vee \phi_3$ we can easily verify that $\phi$ is satisfiable iff $\phi' = (\phi_1 \wedge \phi_2) \vee \phi_3$ is. Hence we can remove $x_i$ and merge the two clauses.

We can apply the previous reasoning to gradually simplify the initial Boolean formula $\phi$ to an eqquivalent (and equisatifiable) one $\phi^*$ in which we have a cojunction of single literals. We cna then check that $\phi^*$ by checking for any variable we have $\phi^* = x_i \wedge \ldots \wedge \neg x_i$. If that is the case we conclude that $\phi^*$ and, hence, $\phi$ is not satisfiable. Otherwise we conclude that $\phi$ is satisfiable.

**Analysis:** We can simplify $\phi$ to $\phi^*$ by processing one of the $n$ variables at a time. Adjusting $\phi$ for each variable requires $O(1)$ operations. The final check over $\phi^*$ can be completed in worst case $O(n^2)$ time. The worst case running time og the proposed algorithm is therefore $O(n^2)$, which implies that the problem of deciding whether a formula in $BF_2$ is satisfiable is in $P$.

**Solution part b:**

- the problem of deciding wether any $\phi \in BF_3$ is satisfiable is in $NP$: We can use the same nondeterministic solver we showed for SAT as any $\phi \in BF_3$ is a Boolean formula.
- We show that $3SAT \leq_p BF3$. First an example: $phi = (p \vee q) \wedge (p \vee r) \wedge (\neg p \vee s) \wedge (p \vee t)$. This formula is not in 3CNF, but the general case will be treated below. This formula is replaced by $\phi' = (p \vee q) \wedge (\neg p \vee p_1) \wedge (p_1 \vee r) \wedge (\neg p_1 \wedge p_2) \wedge (\neg p_2 \vee s) \wedge (\neg p_2 \vee p_3) \wedge (p_3 \vee t) \wedge (\neg p_3 \vee p)$. Note that the $(\neg p \vee p_1), (\neg p_1 \wedge p_2), (\neg p_2 \vee p_3), (\neg p_3 \vee p)$ are expressing that $p, p_1, p_2, p_3$ are all equivalent. Hence in $\phi'$ $(p_1 \vee r)$ is satisfied iff and only if $(p \vee r)$, is satisfies in $\phi$. Thus, $\phi$ is satisfiable iff $\phi'$ is, but in $\phi'$ every variable occurs at most 3 times (I.e., $\phi' \in BF_3$).

  In the general case, we define a reduction $f$ from $3SAT$ to $CNF3$ as follows: Given a Boolean formula $\phi$

- While there exists a Boolean variable that occurs more than 3 times do:
  1. Pick the first variable that occurs more than 3 times from the left. Suppose it is $p$ and that it occurs at $n$ places $(x_1 \vee \phi_1) \wedge \ldots \wedge (x_n \vee \phi_n)$.
  2. For each occurrence of $p$ create a new Boolean variable not used before called $p_i$.
  3. Remove all the $(x_1 \vee \phi_1) \wedge \ldots \wedge (x_n \vee \phi_n)$ clauses and replace them with $(p_1 \vee \phi_1) \wedge (\neg p_1 \vee p_2) \wedge (p_2 \vee \phi_2) \wedge (\neg p_2 \vee p_3) \ldots \wedge (p_n \vee \phi_n) \wedge (\neg p_n \vee p_1)$.
- return $\phi$

**Correctness:** Clearly $f(\phi) \in BF_3$. 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 clasuse in the form $(\neg x_i \vee x_{i+1})$. 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.