A novel characterization of the complexity class $\Theta^P_k$ based on counting and comparison

Thomas Lukasiewicz$^a$ and Enrico Malizia$^a$

$^a$Department of Computer Science, University of Oxford, United Kingdom

firstname.lastname@cs.ox.ac.uk

Abstract

The complexity class $\Theta^P_k$, which is the class of languages recognizable by deterministic Turing machines in polynomial time with at most logarithmic many calls to an NP oracle, received extensive attention in the literature. Its complete problems can be characterized by different specific tasks, such as deciding whether the optimum solution of an NP problem is unique, or whether it is in some sense “odd” (e.g., whether its size is an odd number). In this paper, we introduce a new characterization of this class and its generalization $\Theta^P_{k+1}$ to the $k$-th level of the polynomial hierarchy. We show that problems in $\Theta^P_k$ are also those whose solution involves deciding, for two given sets $A$ and $B$ of instances of two $\Sigma^P_{k-1}$-complete (or $\Pi^P_{k-1}$-complete) problems, whether the number of “yes”-instances in $A$ is greater than those in $B$. Moreover, based on this new characterization, we provide a novel sufficient condition for $\Theta^P_k$-hardness. We also define the general problem $\text{Comp-Valid}_k$, which is proven here $\Theta^P_{k+1}$-complete. $\text{Comp-Valid}_k$ is the problem of deciding, given two sets $A$ and $B$ of quantified Boolean formulas with at most $k$ alternating quantifiers, whether the number of valid formulas in $A$ is greater than those in $B$. Notably, the problem $\text{Comp-Sat}$ of deciding whether a set contains more satisfiable Boolean formulas than another set, which is a particular case of $\text{Comp-Valid}_1$, demonstrates itself as a very intuitive $\Theta^P_2$-complete problem. Nonetheless, to our knowledge, it eluded its formal definition to date. In fact, given its strict adherence to the count-and-compare semantics here introduced, $\text{Comp-Valid}_k$ is among the most suitable tools to prove $\Theta^P_k$-hardness of problems involving the counting and comparison of the number of “yes”-instances in two sets. We support this by showing that the $\Theta^P_2$-hardness of the Max voting scheme over $m$CP-nets is easily obtained via the new characterization of $\Theta^P_k$ introduced in this paper.

1 Introduction

In the quest of characterizing the exact computational complexity of problems, many complexity classes have been defined, and hard problems for them have been, and are currently being, sought. Among these classes, the polynomial-(time) hierarchy (PH) [34] aims at accurately classifying problems whose complexity lies between the classes of languages recognizable by deterministic Turing machines in polynomial time and in polynomial space. To this end, the notion of computation with oracles [1] (introduced initially in [5] with the name of query machine) is used in [34] to analyze the complexity of problems that can be solved by a Turing machine in P or in NP with the aid of an oracle to decide, at unit cost, strings of languages belonging to PH. In particular, the classes $\Delta^P_k$, $\Sigma^P_k$, and $\Pi^P_k$, for $k \geq 0$, constitute the hierarchy: $\Delta^P_0 = \Sigma^P_0 = \Pi^P_0 = P$, and, for $k \geq 1$, $\Delta^P_{k+1} = \text{P}^{\Sigma^P_k}$, $\Sigma^P_{k+1} = \text{NP}^{\Sigma^P_k}$, and $\Pi^P_{k+1} = \text{co-}\Sigma^P_{k+1}$ (more details are given in Section 2). These classes have proven themselves to be useful tools to classify the complexity of numerous natural problems, which have been shown to be complete for some classes of PH [1]. This highlights the strong relevance that the concept of computation with oracles has in complexity theory.

After the introduction of PH, research in computational complexity theory individuated some problems whose complexity could not be precisely captured by the classes of PH. One of them is $\text{Odd-Clique}$ [55], which is the problem of deciding whether the size of the largest cliques of a graph is odd. Clearly, $\text{Odd-Clique}$ is in $\Delta^P_2$, because a P machine can decide $\text{Odd-Clique}$ via a binary search aided by an oracle in NP. However, $\text{Odd-Clique}$ is not $\Delta^P_2$-hard. In fact, only logarithmically many queries need to be issued to the oracle, and this could be a clue for $\text{Odd-Clique}$ not being among the hardest problems of $\Delta^P_2$, which, instead, for their solution require P machines performing polynomially many calls to their NP oracles. Therefore, to precisely
capture this kind of problems, a class of languages that can be recognized by a P machine issuing at most logarithmic many calls to an NP oracle, also denoted P^{NP}[O(\log n)] (where \([O(\log n)]\) denotes the logarithmic restriction on the maximum number of allowed oracle calls), was defined \[27\,23\,21\,37\,9\].

Unlike the nondeterministic levels \(\Sigma_k^P = \text{NP}^{\Sigma_{k-1}^P}\) of PH, for which bounding the allowed number of calls to the oracle do not impose any factual constraint on the computational power of the machine, i.e., \(\text{NP}^{\Sigma_{k-1}^P}[O(1)] = \text{NP}^{\Sigma_{k-1}^P}[O(\log n)] = \text{NP}^{\Sigma_{k-1}^P}\), for the deterministic levels of the hierarchy, this does not seem to be the case. In fact, clearly, \(\text{P}^{\Sigma_{k-1}^P}[O(1)] \subseteq \text{P}^{\Sigma_{k-1}^P}[O(\log n)] \subseteq \text{P}^{\Sigma_{k-1}^P}\), however, it is currently unknown whether the inclusion is strict. On the other hand, ODD-CLIQUE was proven to be complete for the class \(\Theta_2^P = \text{P}^{\text{NP}[O(\log n)]}\) [35]. This supported the widely accepted conjectures that \(\Theta_2^P \neq \Delta_2^P\) and that, for all \(k \geq 2\), \(\Theta_k^P \neq \Delta_k^P\) [37], where \(\Theta_k^P = \text{P}^{\Sigma_{k-1}^P}[O(\log n)]\), which suggested to include the classes \(\Theta_k^P\) as constitutional components of PH [36, 37].

In recent years, many natural problems have been shown to be complete for \(\Theta_2^P\) (see, e.g., [35, 23, 21, 37, 6]). These \(\Theta_2^P\)-complete problems are usually characterized by the task of:

1. deciding whether the optimum value (i.e., maximum or minimum value) of an NP problem belongs to a set of values (or intervals) given in input [35];
2. deciding whether the optimum value of an NP problem is odd/even [35, 21, 4, 23, 33];
3. deciding whether the optimum solution of an NP problem is unique [21];
4. comparing the optimum solutions of two instances of NP-complete problems [33]; see also [6] for more references. Another characterization of a \(\Theta_2^P\)-complete (resp., \(\Theta_2^P\)-complete) problem is based on directed trees whose nodes are parametric queries to an NP (resp., \(\Sigma_{k-1}^P\)) oracle, where the directed tree encodes the dependence structure among the oracle queries [9]. Furthermore, \(\Theta_2^P\) is captured by first-order logic extended by Henkin quantifiers [10].

More recently, also various problems in voting theory and computational social choice were shown to be complete for \(\Theta_2^P\) [29, 19, 15, 25]. These complexity results come as no surprise to us, since, as we will show in this paper, \(\Theta_2^P\) is also the class of problems involving the task of counting “yes”-instances of NP sets, followed by a comparison of the counts (which is similar to what would be done in a voting procedure that requires counting (and comparing) ballots to determine the winner) [7]. A result suggesting that problems in \(\Theta_2^P\) can be characterized in this way appeared only in [33]. However, to our knowledge, neither a single complete problem based on this idea has ever been shown (even in [33] itself there is no problem characterized in this way), nor this characterization has ever been pushed forward or extended to \(\Theta_2^P\).

Note that this semantics of counting and comparing the number of “yes”-instances of two NP-complete problems is very different from (4) above. Indeed, deciding, given two graphs \(G\) and \(H\), whether the smallest vertex covers of \(G\) are smaller than the smallest vertex covers in \(H\) (which is a problem analyzed in [33]) is very different from this paper’s deciding, given two sets \(G = \{G_1, p_1, \ldots, G_n, p_n\}\) and \(H = \{H_1, q_1, \ldots, H_m, q_m\}\) of pairs \((\text{graph, integer})\), whether the number of graphs \(G_i\) having a vertex cover not bigger than \(p_i\) is greater than the number of graphs \(H_j\) having a vertex cover not bigger than \(q_j\).

Showing this “counting of yes-instances and comparison” characterization of \(\Theta_2^P\) is exactly what we pursue in this paper. In particular, we will show that problems whose solution requires the comparison of the number of “yes”-instances of two sets containing instances of \(\Sigma_{k-1}^P\)-complete or \(\Pi_{k-1}^P\)-complete languages, are \(\Theta_k^P\)-hard. To our knowledge, this characterization of \(\Theta_k^P\) is new in the literature, and also its specialization to \(\Theta_2^P\), as already mentioned, has not been extensively investigated so far. Moreover, this result allows us to provide a new sufficient condition for a problem to be \(\Theta_2^P\)-hard. Interestingly, we also show that problems requiring the comparison of the number of “yes”-instances of a set containing instances of a \(\Sigma_{k-1}^P\)-complete problem and a set containing instances of a \(\Pi_{k-1}^P\)-complete problem are computationally easier, since they can be solved in subclasses of \(\Theta_k^P\) (more specifically, in \(\Sigma_{k-1}^P\) or \(\Pi_{k-1}^P\)).

Furthermore, by exploiting the characterization given in this paper, we also define the following general \(\Theta_{k+1}^P\)-complete problem \(\text{COMP-VALID}_k\) (observe the different subscripts, \(k + 1\) and \(k\), respectively): given a pair \((A, B)\) of sets of quantified Boolean formulas with at most \(k\) alternating quantifiers, decide whether the number of valid formulas in \(A\) is greater than those in \(B\). To our knowledge, this is the first time that such a problem is proposed and shown to be \(\Theta_{k+1}^P\)-complete.

In addition, \(\text{COMP-SAT}\), which is a particular case of \(\text{COMP-VALID}_1\), is a very intuitive \(\Theta_2^P\)-complete problem whose hardness lies in the difficulty of comparing the number of satisfiable Boolean formulas in two sets. Thus, it is a very good candidate for reductions to prove \(\Theta_2^P\)-hardness of problems involving the comparison of the number of “yes”-instances of two sets containing instances of NP-complete problems. \(\text{COMP-SAT}\) is the first \(\Theta_2^P\)-complete problem of this kind. In fact, it was successfully used to prove the \(\Theta_2^P\)-hardness of a voting problem.

Footnotes:

1Note that in the literature, \(\Theta_k^P\) was proven to be equivalently characterized by different definitions (see, e.g., [6, 37]), and thus \(\Theta_k^P\)-complete problems are often shown to be complete for apparently different (but actually identical) classes.

2Interestingly, several problems in computational game theory are complete for various classes of PH and \(\Delta_k^P\) in particular. Among them, there are a number of different tasks related to the solution concepts of coalition games (see, e.g., [13, 14, 15, 16]).
Decision problems are maps from strings (encoding the input instance over a fixed alphabet, e.g., the binary alphabet \{0, 1\}) to the set \{"yes", "no"\}. For a decision problem (or, equivalently, a language) \(L\), \(\chi_L\) denotes the \textit{characteristic function} of \(L\), which is the function that, for a string \(s\), \(\chi_L(s) = 1\), if \(s\) is a "yes"-instance of \(L\), and \(\chi_L(s) = 0\), if \(s\) is a "no"-instance of \(L\). Deciding a language (or a problem) \(L\) means, for a given instance \(s\), deciding whether \(s\) is a "yes"-instance of \(L\) or not. For a language \(L\), \(\overline{L}\) is the language complement to \(L\) if and only if all the "yes"-instances of \(L\) are "no"-instances of \(\overline{L}\), and all the "no"-instances of \(L\) are "yes"-instances of \(\overline{L}\).

A (deterministic) Turing machine \(M\) decides a language \(L\), if \(M\) halts in an accepting state on an input string \(s\) if and only if \(\chi_L(s) = 1\). \textit{Nondeterministic} Turing machines are Turing machines that, at some points of their computation, may not have just one single next action to perform, but a \textit{choice} between several possible next actions. A nondeterministic Turing machine \(M\) decides a problem \(L\), if, on any input string \(s\), (i) if \(\chi_L(s) = 1\), there is at least one sequence of choices leading \(M\) to halt in an accepting state (such a sequence is called accepting computation path); and (ii) if \(\chi_L(s) = 0\), all possible sequences of choices of \(M\) lead to a rejecting state.

A complexity class is a set of languages that can be decided by Turing machines of a specific sort (i.e., either deterministic or nondeterministic) within a given bound of computational resources. These computational resources characterizing complexity classes are essentially computation time, working space, and, as we will see later, the possibility to access to a computation oracle. For a complexity class \(C\), co-\(C\) denotes the class of languages whose complements are in \(C\). With a slight abuse of terminology, we say that a Turing machine \(M\) belongs to a complexity class \(C\), if \(M\) is of the sort and uses the amount of computational resources characterizing the class \(C\).

The class \(P\) is the set of decision problems that can be solved by a deterministic Turing machine in polynomial time with respect to the input size, i.e., with respect to the length of the string that encodes the input instance. For a given input string \(s\), its size is usually denoted by \(|s|\).

The class of decision problems that can be solved by nondeterministic Turing machines in polynomial time is denoted by \(NP\). They enjoy a remarkable property: any "yes"-instance \(s\) has a \textit{certificate} for being a "yes"-instance, which has polynomial length and can be checked in deterministic polynomial time (in \(|s|\)). For example, deciding whether a Boolean formula \(\phi(X)\) over the Boolean variables \(X = \{x_1, \ldots, x_n\}\) is satisfiable, i.e., whether there exists some truth assignment to these variables making \(\phi\) true, is a well-known problem in \(NP\); in fact, any satisfying truth assignment for \(\phi\) is clearly a certificate that \(\phi\) is a "yes"-instance, i.e., that \(\phi\) is satisfiable. On the other hand, the problem of deciding whether a Boolean formula \(\phi\) is not satisfiable is in co-NP. Clearly, the class \(P\) is contained in both \(NP\) and co-NP, i.e., \(P \subseteq NP \cap co-NP\).

We will also refer to a type of computation called computation with \textit{oracles}. Intuitively, oracles are subroutines that are supposed to have unit cost. A Turing machine \(M^A\) with oracle, is a Turing machine that during its computation can ask to its oracle to decide a string at unitary cost. The definition of the machine \(M^A\) is independent from its oracle, and the symbol "\(A\)" indicates that different oracles for different languages can be "attached" to \(M\). If \(A\) is a language, by \(M^A\) we denote that the oracle attached to \(M^A\) decides \(A\). If \(C\) is a (deterministic or a nondeterministic) time complexity class, and \(A\) is a language, \(C^A\) denotes the class of languages that can be decided by Turing machines of the sort and the time bound of \(C\) that can moreover query an oracle for \(A\). By extension, for a time complexity class \(C\), and a generic complexity class \(D\), \(C^D\) denotes

---

4If \(C\) is a space complexity class, it is more difficult to define computation with oracles [20, 17].
the class of languages that can be decided by Turing machines of the sort and the time bound of \( C \) that can moreover query an oracle for a language in \( D \). In the following, when we say that a Turing machine \( M \) queries an oracle in \( D \), or a \( D \) oracle, we mean that \( M \) queries an oracle for a language in \( D \).

The classes \( \Sigma^p_k \), \( \Pi^p_k \), and \( \Delta^p_k \), forming the polynomial hierarchy (PH) \([34]\), are defined as follows: \( \Sigma^p_0 = \Pi^p_0 = \Delta^p_0 = \text{P} \), and, for all \( k \geq 1 \), \( \Sigma^p_k = \text{NP}^{\Sigma^p_{k-1}} \), \( \Pi^p_k = \text{P}^{\Sigma^p_{k-1}} \), and \( \Pi^p_0 = \text{co-Sigma}^0_0 \). Here, \( \Sigma^p_k \) (resp., \( \Delta^p_k \)) is the class of languages recognizable by nondeterministic (resp., deterministic) polynomial-time Turing machines with an oracle to recognize, at unit cost, a language in \( \Sigma^p_{k-1} \). Note that \( \Sigma^p_k = \text{NP}^{\Sigma^p_k} = \text{NP} = \text{NP}^0 = \text{co-Sigma}^0_0 = \text{co-NP} \), and \( \Delta^p_k = \text{P}^{\Sigma^p_k} = \text{P} = \text{P}^0 = \text{P} = \text{P}^0 \). Sometimes, a bound is imposed on the number of the allowed oracle calls, highlighted in brackets besides the oracle class. For example, \( \text{P}^{\Sigma^p_k \cdot \log(n)} \) denotes the class of languages recognizable by a deterministic polynomial-time Turing machine that is allowed to query a \( \Sigma^p_{k-1} \) oracle at most logarithmic many times (in the size of the input). In particular, classes \( \Theta^p_k = \text{P}^{\Sigma^p_k \cdot \log(n)} \) were proposed to be included in the standard definition of the PH as well \([36, 37]\). To this end, we pose \( \Theta^p_0 = \text{P} \), and observe that \( \Theta^p_0 \) and \( \Delta^p_k \) are deterministic classes, as the machine calling the oracle is deterministic. This implies that \( \Theta^p_k \) and \( \Delta^p_k \) are closed under complement, i.e., \( \Theta^p_k = \text{co-Theta}^p_k \) and \( \Delta^p_k = \text{co-Delta}^p_k \). Given their definitions, for all \( k \geq 1 \), the relationships among the mentioned classes are as follows (see, e.g., \([36, 37]\)): \( \Sigma^p_k \cup \Pi^p_k \leq \Theta^p_{k+1} \leq \Delta^p_{k+1} \leq \Sigma^p_{k+1} \cap \Pi^p_{k+1} \).

### 2.2 Prototypical hard problems

We now recall the notion of reducibility among decision problems. A decision problem \( L_1 \) is \((\text{Karp})\ reducible\) to a decision problem \( L_2 \), denoted by \( L_1 \leq L_2 \), if there is a computable function \( h \) (called reduction) such that, for every string \( s \), \( h(s) \) is defined, and \( s \) is a "yes"-instance of \( L_1 \) if and only if \( h(s) \) is a "yes"-instance of \( L_2 \), i.e., for all strings \( s \), \( \chi_{L_1}(s) = \chi_{L_2}(h(s)) \). This type of reduction is called Karp reduction. A decision problem \( L_1 \) is \((\text{polynomial})\ (\text{Karp})\ reducible\) to a decision problem \( L_2 \), denoted by \( L_1 \leq_p L_2 \), if there is a polynomial-time (Karp) reduction from \( L_1 \) to \( L_2 \). In this paper we will consider only polynomial-time Karp reductions.

A decision problem \( L \) is \((\text{hard})\ for\ a\ class\ \( C \) of the \( \text{PH} \) at any level \( k \geq 1 \), i.e., beyond \( \text{P} \), or \( C \)-hard, if every problem in \( C \) is polynomially reducible to \( L \); if \( L \) is hard for \( C \) and belongs to \( C \), then \( L \) is \((\text{complete})\ for\ \( C \), or \( C \)-complete. Thus, problems that are complete for \( C \) are the most difficult problems in \( C \). In particular, they cannot belong to some lower class in the hierarchy, unless some collapse of the hierarchy’s levels occurs.

An \( n \)-ary Boolean function \( f \) is a mapping \( f : \{\text{true, false}\}^n \to \{\text{true, false}\} \) from the \( n \)-dimensional Boolean space to a Boolean value. A way to represent \( n \)-ary Boolean functions is through \( \text{Boolean formulas} \ \phi(X) \) over the set of Boolean variables \( X = \{x_1, \ldots, x_n\} \). Boolean formulas are inductively constructed from Boolean variables via the unary Boolean operator \( \neg \) and the binary Boolean operators \( \land \) and \( \lor \). Boolean variables \( x_1, \ldots, x_n \) and their negations \( \neg x_1, \ldots, \neg x_n \) are called \( \text{liters} \). A \( \text{clause} \) and a \( \text{term} \) are a disjunction and a conjunction of \( \text{liters} \), respectively. A Boolean formula is in \((\text{conjunctive normal form}) \) (or \( \text{CNF} \)), if it is a conjunction of \( \text{clauses} \), while it is in \((\text{disjunctive normal form}) \) (or \( \text{DNF} \)), if it is a disjunction of \( \text{terms} \). For example, \( \gamma_1(x_1, x_2, x_3, x_4) = (x_1 \lor \neg x_2 \land x_3) \lor (x_2 \lor \neg x_3 \lor \neg x_4) \) is a \( \text{CNF} \) formula, while \( \gamma_2(x_1, x_2, x_3, x_4) = (x_1 \land \neg x_2 \land x_3) \lor (x_2 \land \neg x_3 \land \neg x_4) \) is a \( \text{DNF} \) formula.

Deciding the satisfiability of \( \text{Boolean formulas} \) is the prototypical \( \text{NP}-\text{complete} \) problem, which remains \( \text{NP}-\text{hard} \) even if only \( \text{CNF} \) formulas are considered \([3, 22]\). We denote this problem by \( \text{SAT} \). The complemenetary problem \( \text{UNSAT} \) of deciding whether a given \( \text{Boolean formula} \) is not satisfiable is \( \text{co-NP}-\text{complete} \). It remains co-\( \text{NP}-\text{hard} \) even if only \( \text{CNF} \) formulas are considered, and it is the equivalent to the problem \( \text{TAUT} \) of deciding whether a \( \text{3DNF} \) formula is a tautology.

We next define the prototypical \( \Sigma^p_k \) and \( \Pi^p_k \)-complete \( \text{QBF} \)\( Q_k, k \)-problems as follows: given a quantified Boolean formula (QBF) \( \Phi = (Q_1 X_1)(Q_2 X_2) \cdots (Q_k X_k) \phi(X_1, X_2, \ldots, X_k) \), where

- \( Q_1, Q_2, \ldots, Q_k \) is a sequence of \( k \) alternating quantifiers \( Q_i \in \{\exists, \forall\} \), and
- \( \phi(X_1, X_2, \ldots, X_k) \) is a (non-quantified) Boolean formula over \( k \) disjoint sets \( X_1, X_2, \ldots, X_k \) of Boolean variables,

decide whether \( \Phi \) is valid. The problem \( \text{QBF}_\exists, k \) is \( \Sigma^p_k \)-complete \([34, 38]\), while \( \text{QBF}_\forall, k \) is \( \Pi^p_k \)-complete \([34, 38]\). These problems remain hard for their respective classes even if \( \phi(X_1, X_2, \ldots, X_k) \) is \( \text{in 3CNF} \), when \( Q_k = \exists \), and if \( \phi(X_1, X_2, \ldots, X_k) \) is \( \text{in 3DNF} \), when \( Q_k = \forall \) \([34, 38]\).

\(^6\)Observe that \( \gamma_1(x_1, x_2, x_3, x_4) \) and \( \gamma_2(x_1, x_2, x_3, x_4) \) are not \( \text{equivalent} \). Nevertheless, \( \text{Boolean formulas in CNF and DNF} \) are actually \( \text{linked} \), and they can be \( \text{transformed} \) from one form to the other. In particular, translating a positive \( \text{CNF} \) formula into an equivalent minimal \( \text{DNF} \) one (or vice versa) involves a process called \( \text{dualization} \), which is currently unknown to be feasible in \( \text{output-polynomial time} \) (for more on this, see, e.g., \([11, 12]\) and references therein).
We denote by $\text{QBF}_{k,3}^{\text{CNF}}$ (resp., $\text{QBF}_{k,3}^{\text{DNF}}$) the problem of deciding the validity of formulas $\Phi = (Q_1 X_1) \ldots (Q_k X_k) \phi(X_1, \ldots, X_k)$, where $Q_k$ is $\exists$ (resp., $\forall$), and $\phi(X_1, \ldots, X_k)$ is in 3CNF (resp., 3DNF). For odd $k$, $\text{QBF}_{1,3}^{\text{CNF}}$ (resp., $\text{QBF}_{1,3}^{\text{DNF}}$) is complete for $\Sigma_k^p$ (resp., $\Pi_k^p$), while, for even $k$, $\text{QBF}_{k,3}^{\text{CNF}}$ (resp., $\text{QBF}_{k,3}^{\text{DNF}}$) is complete for $\Pi_k^p$ (resp., $\Sigma_k^p$). Observe that $\text{QBF}_{1,3}^{\text{CNF}}$ (resp., $\text{QBF}_{1,3}^{\text{DNF}}$) is equivalent to SAT (resp., TAUT).

2.3 A previous characterization of $\Theta_k^p$

Wagner \[35\] \[37\] analyzed extensively the properties of $\Theta_k^p$ and underlined that his results can be generalized to upper levels of the PH. In particular, we report below two key results of Wagner \[37\] in their generalized form to $\Theta_k^p$.

A first result intuitively states that, for any language $B \in \Theta_k^{p+1}$, the task of deciding $B$ can be faithfully transformed into the task of deciding, for a suitable language $C \in \Sigma_k^p$ and a suitable sequence $(y_1, \ldots, y_n)$ of $C$'s instances with $\chi_C(y_i) \geq \ldots \geq \chi_C(y_n)$, whether the maximum index $i$ for which $y_i$ is a “yes”-instance of $C$ is odd.

The formulation of the following lemma results from a combination of the statements of the equivalent lemmas in \[37\] \[3\] \[4\].

**Lemma 2.1** ([37] Corollary 6.4], [3] Lemma 12, [4] consequence of Theorem 8). Let $B$ be a problem. Then, $B \in \Theta_k^{p+1}$ if and only if there exist a problem $C \in \Sigma_k^p$ and a polynomial-time computable function $f$ such that, for all instances $x$ of $B$, $f(x) = (y_1, \ldots, y_{|f(x)|})$ is a sequence of $C$’s instances with $\chi_C(y_1) \geq \ldots \geq \chi_C(y_{|f(x)|})$, and $\chi_B(x) = 1 \Leftrightarrow \max \{i \mid 1 \leq i \leq p(|x|), \chi_C(y_i) = 1\}$ is odd.

Note that, in the statement of the following lemma, the polynomial, $p(|x|)$, is a polynomial, because the function $f$ is polynomial-time computable.

A second result gives a sufficient condition for the $\Theta_k^{p+1}$-hardness of a problem $B$. Intuitively, this result states that $B$ is $\Theta_k^{p+1}$-hard, if there exists a reduction from $B$ to the problem of deciding, for a given set $\{x_1, \ldots, x_n\}$ of instances of a $\Sigma_k^p$-complete problem $A$, whether the maximum index $i$ for which $x_i$ is a “yes”-instance of $A$ is odd. Interestingly, the result holds even if it is assumed that $\chi_A(x_1) \geq \ldots \geq \chi_A(x_n)$.

**Lemma 2.2** ([37] Theorem 5.2], [37] Theorem 7.1). Let $A$ be a $\Sigma_k^p$-complete problem, and let $B$ be a problem. Then, $B$ is $\Theta_k^{p+1}$-hard, if there exists a polynomial-time computable function $f$ such that, for all sets $X = \{x_1, \ldots, x_n\}$ of instances of $A$, $\{|x_i; \chi_A(x_i) = 1\} =$ odd $\Leftrightarrow \chi_B(f(X)) = 1$. The $\Theta_k^{p+1}$-hardness of $B$ remains proven even if sets $X$ are assumed to be such that $\chi_A(x_1) \geq \ldots \geq \chi_A(x_n)$.

3 A new characterization of $\Theta_k^p$ and its hard problems

In this section, we provide a new characterization of $\Theta_k^p$ based on the counting-and-comparison semantics. In particular, we first give an overview of the results obtained, and then we look at the details of the proofs.

3.1 Overview of results

The first results that we obtain are the analogues of those that are reported in this paper as Lemmas 2.1 and 2.2 and are broad general theoretical results for the complexity classes $\Theta_k^p$.

In particular, on the one hand, we show that, for any language $B \in \Theta_k^{p+1}$, the task of deciding $B$ can be faithfully transformed into the task of deciding, for two suitable languages $C_1, C_2 \in \Sigma_k^p$ and two suitable sequences $(y_1, \ldots, y_n)$ and $(z_1, \ldots, z_m)$ of $C_1$’s and $C_2$’s instances, respectively, with $\chi_{C_1}(y_1) \geq \ldots \geq \chi_{C_1}(y_n)$ and $\chi_{C_2}(z_1) \geq \ldots \geq \chi_{C_2}(z_m)$, whether the maximum index $i$ for which $y_i$ is a “yes”-instance of $C_1$ is bigger than the maximum index $j$ for which $z_j$ is a “yes”-instance of $C_2$.

On the other hand, we show also that, a problem $B$ is $\Theta_k^{p+1}$-hard, if there exists a reduction from $B$ to the problem of deciding, for a given pair of sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ of instances of two (not necessarily distinct) problems $A_1$ and $A_2$, respectively, being both of them either $\Sigma_k^p$-complete or $\Pi_k^p$-complete, whether the number of “yes”-instances of $A_1$ in $X$ is greater than the number of “yes”-instances of $A_2$ in $Y$. Interestingly, the hardness holds even if sets $X$ and $Y$ are assumed to be such that $\chi_{A_1}(x_1) \geq \ldots \geq \chi_{A_1}(x_n)$ and $\chi_{A_2}(y_1) \geq \ldots \geq \chi_{A_2}(y_m)$.

The previous result is shown via an intermediate theorem stating that, if $A_1$ and $A_2$ are two (not necessarily distinct) $\Sigma_k^p$-complete (or $\Pi_k^p$-complete) problems, for any pair of sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ of instances of $A_1$ and $A_2$, respectively, deciding whether the number of “yes”-instances of $A_1$ in $X$ is greater than the number of “yes”-instances of $A_2$ in $Y$.

\[\text{Note the difference in the subscripts of the notations } \text{QBF}_{Q_1,k} \text{ and } \text{QBF}_{Q_2,k}^{\text{CNF}} \text{ (resp., } \text{QBF}_{Q_2,k}^{\text{DNF}}). \text{ In the former notation, } Q_1 \text{ is the first quantifier of the sequence, and, for notational convenience, we place “}Q_1\text{” before “}k\text{” in the subscript. On the other hand, in the latter notation, } Q_k \text{ is the last quantifier of the sequence, and, for notational convenience, we place “}Q_k\text{” after “}k\text{” in the subscript.}\]
than the number of “yes”-instances of \(A_2\) in \(Y\) is \(\Theta_{k+1}^P\)-hard. Clearly, this problem is also in \(\Theta_{k+1}^P\), and hence \(\Theta_{k+1}^P\)-complete. Therefore, problems in \(\Theta_{k+1}^P\) can actually be exactly characterized also by this semantics of counting and comparison. We also show that if \(A_1\) and \(A_2\) are one in \(\Sigma_k^P\) and the other in \(\Pi_k^P\), then comparing the number of “yes”-instances of the two sets is actually a problem belonging to subclasses of \(\Theta_{k+1}^P\). In particular, if \(A_1\) is in \(\Sigma_k^P\), and \(A_2\) is in \(\Pi_k^P\), then the comparison can be done in \(\Sigma_k^P\). Symmetrically, if \(A_1\) is in \(\Pi_k^P\), and \(A_2\) is in \(\Sigma_k^P\), then the comparison can be done in \(\Pi_k^P\).

After these general results, we define the problem \(\text{Comp-Valid}_k\) that is based on the idea of counting and comparison, and we prove it \(\Theta_{k+1}^P\)-complete. \(\text{Comp-Valid}_k\) is defined as follows. Given a pair \((A, B)\) of sets of QBFs with at most \(k\) alternating quantifiers, decide whether the number of valid formulas in \(A\) is greater than the number of valid formulas in \(B\). We stress here that the formulas in \(A\) and \(B\) are neither restricted to have the same outermost quantifier, nor to have the same number of alternating quantifiers. To our knowledge, this is the first time in the literature that such a problem \(\text{Comp-Valid}_k\) is defined and shown to be \(\Theta_{k+1}^P\)-complete. Furthermore, we also prove that the hardness of \(\text{Comp-Valid}_k\) holds even if \(|A| = |B|\), all formulas in \((A, B)\) are instances of \(\text{QBF}_{k,\Sigma}^\text{CNF}\) (resp., \(\text{QBF}_{k,\Pi}^\text{DNF}\)), have the same number of clauses (resp., terms), and, for each \(1 \leq d \leq k\), quantifiers \(Q_d\) of all formulas in \((A, B)\) are defined on the very same set of variables. As it will emerge from the proof, the \(\Theta_{k+1}^P\)-hardness of \(\text{Comp-Valid}_k\) requires that the number of formulas in the sets \(A\) and \(B\) with actually \(k\) alternating quantifiers is unbounded.

By combining the results of the theorems proven in this section, Figure 1 summarizes the complexity of \(\text{Comp-Valid}_k\), according to the various types of formulas contained in \(A\) and \(B\).

| Formulas in \(A\) | Formulas in \(B\) | \(k\) is odd | \(k\) is even |
|-------------------|-------------------|--------------|--------------|
| \(\text{QBF}_{k,\Sigma}^\text{CNF}\) | \(\Theta_{k+1}^P\) | \(\Pi_k^P\) | \(\Pi_k^P\) |
| \(\text{QBF}_{k,\Pi}^\text{DNF}\) | \(\Sigma_k^P\) | \(\Theta_{k+1}^P\) | \(\Theta_{k+1}^P\) |

Figure 1: Summary of the complexity results for \(\text{Comp-Valid}_k\) when specific types of formulas are in sets \(A\) and \(B\). \(\text{Comp-Valid}_k\) is actually complete for the respective complexity classes shown in the tables.

The problem \(\text{Comp-Valid}_1\), when all formulas are furthermore restricted to be instances of \(\text{QBF}_{k,\Sigma}^\text{CNF}\) (i.e., SAT), is equivalent to the problem \(\text{Comp-Sat}\) introduced in \([25]\), which is: Given a pair \((A, B)\) of sets of 3CNF formulas, decide whether the number of satisfiable formulas in \(A\) is greater than the number of satisfiable formulas in \(B\). By the results proven in this paper, \(\text{Comp-Sat}\) is \(\Theta_2^P\)-complete, and it is \(\Theta_2^P\)-hard even if all the formulas have the same number of clauses and are defined over the same set of variables.

### 3.2 Derivation of the general results

In this section, we prove the results anticipated above. In fact, besides the concepts of verification of optimum solutions, “oddity” of optimum solutions, uniqueness of optimum solutions, and comparison of optimum solutions (see, e.g., \([35, 21, 6, 33]\)), we show that a problem is \(\Theta_k^P\)-hard if, for its solution, it is required to count the number of “yes”-instances of two sets \(A\) and \(B\) of instances of \(\Sigma_k^P\)-complete, or \(\Pi_k^P\)-complete, languages, and compare the computed numbers.

The following result is the analogue of Lemma 2.1 for the new characterization of \(\Theta_k^P\). Intuitively, it states that, for any language \(B \in \Theta_{k+1}^P\), the task of deciding \(B\) can be faithfully transformed into the task of deciding, for two suitable languages \(C_1, C_2 \in \Sigma_k^P\) and two suitable sequences \((y_1, \ldots, y_m)\) and \((z_1, \ldots, z_m)\) of \(C_1\)'s and \(C_2\)'s instances, respectively, with \(\chi_{C_1}(y_1) \geq \cdots \geq \chi_{C_1}(y_m)\) and \(\chi_{C_2}(z_1) \geq \cdots \geq \chi_{C_2}(z_n)\), whether the maximum index \(i\) for which \(y_i\) is a “yes”-instance of \(C_1\) is bigger than the maximum index \(j\) for which \(z_j\) is a “yes”-instance of \(C_2\).

The idea behind the proof is the following. By Wagner's Lemma 2.1, since \(B \in \Theta_{k+1}^P\), there is a language \(C \in \Sigma_k^P\) and a polynomial-time computable function \(f\) such that, for any instance \(x\) of \(B\), \(f(x)\) is a sequence \((w_1, \ldots, w_p)\) of instances of \(C\) with \(\chi_C(w_1) \geq \cdots \geq \chi_C(w_p)\), and \(x\) is a “yes”-instance of \(B\) if and only if the maximum index \(\ell\) for which \(w_\ell\) is a “yes”-instance of \(C\) is odd. Essentially, what is done in the proof is “splitting” the sequence \((w_1, \ldots, w_p)\) into two new sequences \((y_1, \ldots, y_m)\) and \((z_1, \ldots, z_m)\) of instances of \(C_1\) and \(C_2\), respectively. The first of the new sequences contains strings \(w_\ell\) with \(\ell\) odd, while the second one contains strings \(w_\ell\) with \(\ell\) even. Now, if the maximum index \(\ell\) for which \(w_\ell\) is a “yes”-instance of \(C\) is odd, then that particular instance is in the first of the new sequences, and hence the maximum index \(i\) for which \(y_i\) is a “yes”-instance of \(C_1\) is bigger than the maximum index \(j\) for which \(z_j\) is a “yes”-instance of \(C_2\) (because, in this case, the maximum \(j\) would equal \(i-1\)).

**Theorem 3.1.** Let \(B\) be a problem. Then, \(B \in \Theta_{k+1}^P\) if and only if there exist two (not necessarily distinct) problems \(C_1, C_2 \in \Sigma_k^P\) and two polynomial-time computable functions \(f_1\) and \(f_2\) such that, for all instances
Let us assume that $B \in \Theta_{k+1}^P$. By Lemma 2.1 there exist a problem $C \in \Sigma_k^P$ and a polynomial-time computable function $f$ such that, for all instances $x$ of $B$, $f(x) = (w_1, \ldots, w_p(\|x\|))$ is a sequence of $C$'s instances with $\chi_C(w_1) = \cdots = \chi_C(w_p(\|x\|))$, and $\chi_B(x) = 1 \iff \max \{\|i\| \leq i \leq p(\|x\|), \chi_C(x) = 1\}$ is odd.

Now, consider languages $C_1$ and $C_2$ such that $C_1 = C_2 = C$ (i.e., for any string $x$, $\chi_C(x) = \chi_C'(x)$). Clearly, $C \in \Sigma_k^P$; $C_1$ and $C_2$ belong to $\Sigma_k^P$ as well. Functions $f_1$ and $f_2$ are defined from $f$ as follows. Assume that for an instance $x$ of $B$, $f(x) = (w_1, \ldots, w_p(\|x\|))$. On the one hand, function $f_1$ is such that $f_1(x) = (y_1, \ldots, y_{p(\|x\|)})$, where $p_1(\|x\|) = \lfloor \frac{p(\|x\|)}{2} \rfloor$ and $y_i = w_{2i-1}$ for all $i$. On the other hand, function $f_2$ is such that $f_2(x) = (z_1, \ldots, z_{p(\|x\|)})$, where $p_2(\|x\|) = \lfloor \frac{p(\|x\|)}{2} \rfloor$ and $z_j = w_{2j}$ for all $j$. Intuitively, $f_1(x)$ produces the sequence of strings of $f(x)$ with odd index, while $f_2(x)$ produces the sequence of strings of $f(x)$ with even index. By their definition, $f_1$ and $f_2$ are polynomial-time computable, because $f$ is polynomial-time computable. Moreover, since $\chi_C(w_1) = \cdots = \chi_C(w_{p(\|x\|)})$, $\chi_C(y_1) = \cdots = \chi_C(y_{p(\|x\|)})$ and $\chi_C(z_1) = \cdots = \chi_C(z_{p(\|x\|)})$, as well. Now, observe that

$$\chi_B(x) = 1 \iff \max \{\|i\| \leq i \leq p(\|x\|), \chi_C(x) = 1\} \text{ is odd}$$

$$\iff \max \{i \leq i \leq p(\|x\|), \chi_C(x) = 1\} > \max \{j \leq j \leq p(\|x\|), \chi_C(z_j) = 1\}.$$
Σ_k^P-complete, there exist polynomial reductions h_1 and h_2 from E_1 to A_1 and from E_2 to A_2, respectively. Intuitively, the composition of f_1 with h_1 and of f_2 with h_2 provides the reduction sought.

**Theorem 3.2.** Let A_1 and A_2 be two (not necessarily distinct) Σ_k^P-complete problems. Then, for a given pair of sets X = \{x_1, \ldots, x_n\} and Y = \{y_1, \ldots, y_m\} of instances of A_1 and A_2, respectively, deciding whether 
\(|\{x_i : \chi_{A_1}(x_i) = 1\}| > |\{y_j : \chi_{A_2}(y_j) = 1\}|\) is \(\Theta_k^{P+1}\)-hard. The hardness holds even if sets X and Y are assumed to be such that \(\chi_{A_1}(x_1) \geq \cdots \geq \chi_{A_1}(x_n)\) and \(\chi_{A_2}(y_1) \geq \cdots \geq \chi_{A_2}(y_m)\), and \(n = m\).

**Proof.** Let us denote by CC the task of counting and comparing the number of "yes"-instances in the sets X and Y. We are going to prove the hardness of CC by showing that any problem D in \(\Theta_k^{P+1}\) is polynomially reducible to CC. Let D be any problem in \(\Theta_k^{P+1}\). By Theorem 3.1, there exist two problems E_1, E_2 ∈ Σ_k^P and two polynomial-time computable functions f_1 and f_2 such that, for all instances v of D, \(f_1(v) = (w_1, \ldots, w_{p_1(|v|)})\), \(f_2(v) = (z_1, \ldots, z_{p_2(|v|)})\) is a sequence of E_1’s instances with \(\chi_{E_1}(w_1) \geq \cdots \geq \chi_{E_1}(w_{p_1(|v|)})\), \(\chi_{E_1}(w_1) = 1\) > \(\max\{j \mid 1 \leq j \leq p_2(|v|)\}, \chi_{E_2}(z_j) = 1\})

Furthermore, since E_1, E_2 ∈ Σ_k^P, and A_1 and A_2 are Σ_k^P-complete, there exist polynomial reductions h_1 and h_2 such that \(\chi_{E_1}(w) = 1 \iff \chi_{A_1}(h_1(w)) = 1\) for all strings w, and \(\chi_{E_2}(z) = 1 \iff \chi_{A_2}(h_2(z)) = 1\) for all strings z. Consider a string v instance of D. From v we derive the sets X = \{x_1, \ldots, x_n\} and Y = \{y_1, \ldots, y_m\} (which constitute the instance of CC) as follows: \(x_i = h_1(w_i)\) for all i, and \(y_j = h_2(z_j)\) for all j. Observe that, from \(\chi_{E_1}(w_i) \geq \cdots \geq \chi_{E_1}(w_{p_1(|v|)})\), it follows that \(\chi_{A_1}(x_1) \geq \cdots \geq \chi_{A_1}(x_{p_1(|v|)})\), and from \(\chi_{E_2}(z_j) \geq \cdots \geq \chi_{E_2}(z_{p_2(|v|)})\), it follows that \(\chi_{A_2}(y_1) \geq \cdots \geq \chi_{A_2}(y_{p_2(|v|)})\). Since f_1 and f_2 are polynomial-time computable, sets X and Y are computable in polynomial time from v. Now, observe that

\[\chi_{D}(v) = 1 \iff (\max\{i \mid 1 \leq i \leq p_1(|v|), \chi_{E_1}(w_i) = 1\}) > (\max\{i \mid 1 \leq i \leq p_2(|v|), \chi_{E_2}(z_i) = 1\})\]

To conclude, we show that the result holds even if \(n = m\). In fact, assume w.l.o.g. that \(n < m\). Then, we can add to set X (with indices greater than \(n\) "no"-instances of A_1 until we have \(|X| = |Y|\). Clearly, this modification of X preserves the property that \(\chi_{A_1}(x_1) \geq \cdots \geq \chi_{A_1}(x_{p_1(|v|)})\), and does not alter the number of "yes"-instances of A_1 in X. Hence, again, \(\chi_{D}(v) = 1 \iff (\{x_i : \chi_{A_1}(x_i) = 1\}) > (\{y_j : \chi_{A_2}(y_j) = 1\})\). \(\square\)

Similarly, for two given sets X = \{x_1, \ldots, x_n\} and Y = \{y_1, \ldots, y_m\} of instances of two (not necessarily distinct) Π_k^P-complete problems A_1 and A_2, respectively, deciding whether the number of "yes"-instances of A_1 in X is greater than the number of "yes"-instances of A_2 in Y is \(\Theta_k^{P+1}\)-hard. Also in this case, the hardness holds even if it is assumed that \(\chi_{A_1}(x_1) \geq \cdots \geq \chi_{A_1}(x_n)\) and \(\chi_{A_2}(y_1) \geq \cdots \geq \chi_{A_2}(y_m)\), and \(n = m\). The proof is based on the fact that Π_k^P-complete problems are the complement of Σ_k^P-complete ones. Therefore, sets X and Y can be rearranged so that Theorem 3.2 can be used.

**Theorem 3.3.** Let A_1 and A_2 be two (not necessarily distinct) Π_k^P-complete problems. Then, for a given pair of sets X = \{x_1, \ldots, x_n\} and Y = \{y_1, \ldots, y_m\} of instances of A_1 and A_2, respectively, deciding whether 
\(|\{x_i : \chi_{A_1}(x_i) = 1\}| > |\{y_j : \chi_{A_2}(y_j) = 1\}|\) is \(\Theta_k^{P+1}\)-hard. Hardness holds even if the sets X and Y are assumed to be such that \(\chi_{A_1}(x_1) \geq \cdots \geq \chi_{A_1}(x_n)\) and \(\chi_{A_2}(y_1) \geq \cdots \geq \chi_{A_2}(y_m)\), and \(n = m\).

**Proof.** Let us assume \(n = m\), and remember that A_1 and A_2 denote the complement problems to A_1 and A_2, respectively. Clearly, A_1 and A_2 are Π_k^P-complete problems. Let us define, \(A'_1 = \overline{A_2}\), and \(A'_2 = \overline{A_1}\) (note the inversion of the subscripts). Let X’ = \{x_1’, \ldots, x_n’\} and Y’ = \{y_1’, \ldots, y_m’\} be two sets of instances of A’_1 and A’_2, respectively, such that \(\chi_{A'_1}(x'_i) \geq \cdots \geq \chi_{A'_1}(x'_n)\) and \(\chi_{A'_2}(y'_i) \geq \cdots \geq \chi_{A'_2}(y'_n)\). From Theorem 3.2, we know that deciding whether \(|\{x_i : \chi_{A'_1}(x_i) = 1\}| > |\{y_j : \chi_{A'_2}(y_j) = 1\}|\) is \(\Theta_k^{P+1}\)-hard.

Starting from X’ and Y’, we define sets X = \{x_1, \ldots, x_n\} and Y = \{y_1, \ldots, y_n\}, where \(x_i = x'_{n-i+1}\), and \(y_j = y'_{n-j+1}\). Intuitively, we put in X the elements of Y’ in inverted order, and we put in Y the elements of X’ in inverted order. By their definitions, X and Y are sets of instances of A_1 and A_2, respectively, and \(\chi_{A_1}(x_1) \geq \cdots \geq \chi_{A_1}(x_n)\) and \(\chi_{A_2}(y_1) \geq \cdots \geq \chi_{A_2}(y_n)\). Observe that

\[|\{y'_j : \chi_{A'_2}(y'_j) = 1\}| > |\{x'_i : \chi_{A'_1}(x'_i) = 1\}| \iff \{y'_j : \chi_{A'_2}(y'_j) = 0\} < \{x'_i : \chi_{A'_1}(x'_i) = 0\}\] (because \(n = m\))

\[|\{x'_i : \chi_{A'_1}(x'_i) = 1\}| > |\{y'_j : \chi_{A'_2}(y'_j) = 1\}| \iff \{x'_i : \chi_{A'_1}(x'_i) = 0\} < \{y'_j : \chi_{A'_2}(y'_j) = 0\}\]

\[|\{y'_j : \chi_{A'_2}(y'_j) = 0\} < \{x'_i : \chi_{A'_1}(x'_i) = 0\}\]

\[|\{x'_i : \chi_{A'_1}(x'_i) = 1\}| > |\{y'_j : \chi_{A'_2}(y'_j) = 1\}| \iff \{x'_i : \chi_{A'_1}(x'_i) = 1\} > |\{y'_j : \chi_{A'_2}(y'_j) = 1\}|\). \(\square\)
It is interesting to see that, for two given sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ of instances of problems $A_1$ and $A_2$, respectively, if $A_1$ is in $\Sigma^p_k$ and $A_2$ is in $\Pi^p_k$, then deciding whether $|\{x_i : \chi_{A_1}(x_i) = 1\}| > \{|y_j : \chi_{A_2}(y_j) = 1\}|$ is actually computationally easier than $\Theta^p_{k+1}$, in particular, it is in $\Sigma^p_k$.

The intuition behind this simplification of the problem is the following. If there are $p$ “yes”-instances in $X$, and $q$ “no”-instances in $Y$, and $p + q > |Y|$, then $|\{x_i : \chi_{A_1}(x_i) = 1\}| > \{|y_j : \chi_{A_2}(y_j) = 1\}|$. Therefore, if this is the case, it suffices to guess the $p$ “yes”-instances in $X$, and the $q$ “no”-instances in $Y$. Guessing the $p$ “yes”-instances in $X$ is feasible by an NP machine, and guessing the $q$ “no”-instances in $Y$ is feasible in NP as well (because $A_2$ is in a “complement” class). Essentially, if $A_1 \in \Sigma^p_k$ and $A_2 \in \Pi^p_k$, we do not need to actually count precisely the number of “yes”-instances in $X$ and $Y$.

**Theorem 3.4.** Let $A_1$ be a $\Sigma^p_k$-complete problem and $A_2$ be a $\Pi^p_k$-complete problem. Then, for a given pair of sets $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_m\}$ of instances of $A_1$ and $A_2$, respectively, deciding whether $|\{x_i : \chi_{A_1}(x_i) = 1\}| > \{|y_j : \chi_{A_2}(y_j) = 1\}|$ is $\Sigma^p_k$-complete.

**Proof.** Let $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$. We first show that $|\{x_i : \chi_{A_1}(x_i) = 1\}| > \{|y_j : \chi_{A_2}(y_j) = 1\}|$ if and only if there are $p$ “yes”-instances of $A_1$ in $X$ and $q$ “no”-instances of $A_2$ in $Y$ such that $p + q > |Y|$. Assume that $|\{x_i : \chi_{A_1}(x_i) = 1\}| > \{|y_j : \chi_{A_2}(y_j) = 1\}|$. Observe that $|Y| = \{|y_j : \chi_{A_2}(y_j) = 1\}|$ and $p + q > |Y|$. Conversely, assume that there are $p$ “yes”-instances of $A_1$ in $X$ and $q$ “no”-instances of $A_2$ in $Y$ such that $p + q > |Y|$. Observe that $p \leq \{|x_i : \chi_{A_1}(x_i) = 1\}|$ and $q \leq \{|y_j : \chi_{A_2}(y_j) = 1\}|$. By summing up these two inequalities, $p + q \geq \{|x_i : \chi_{A_1}(x_i) = 1\} + \{|y_j : \chi_{A_2}(y_j) = 1\}$. Therefore, $|\{x_i : \chi_{A_1}(x_i) = 1\}| > \{|y_j : \chi_{A_2}(y_j) = 1\}|$.

Thus, as for membership in $\Sigma^p_k$, to decide whether $|\{x_i : \chi_{A_1}(x_i) = 1\}| > \{|y_j : \chi_{A_2}(y_j) = 1\}|$, it suffices to guess a set of $p$ “yes”-instances of $A_1$ from $X$ along with their witnesses (feasible in NP); (2) check that all the guessed $p$ instances of $A_1$ are actually “yes”-instances, and all the guessed $q$ instances of $A_2$ are actually “no”-instances (feasible via polynomially many specific, $p + q$ calls to a suitable $\Pi^p_{k-1}$ oracle by passing the guessed witnesses); and (3) verify that $p + q > |Y|$ (feasible in P). Hence, this is overall feasible in $\Sigma^p_k$.

Hardness for $\Sigma^p_k$ is proven as follows. Consider an instance $x_1$ of $A_1$, and observe that, for sets $X = \{x_1\}$ and $Y = \{\}$, deciding whether the number of “yes”-instances of $A_1$ in $X$ is greater than the number of “yes”-instances of $A_2$ in $Y$ is equivalent to decide whether $x_1$ is a “yes”-instance of $A_1$ (which is $\Sigma^p_k$-hard, because $A_1$ is $\Sigma^p_k$-complete).

On the other hand, for two given sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ of instances of problems $A_1$ and $A_2$, respectively, if $A_1$ is in $\Pi^p_k$ and $A_2$ is in $\Sigma^p_k$, then deciding whether $|\{x_i : \chi_{A_1}(x_i) = 1\}| > \{|y_j : \chi_{A_2}(y_j) = 1\}|$ is in $\Pi^p_k$. The reason for the simplification of this task is similar to the previous. If there are $p$ “no”-instances in $X$, and $q$ “yes”-instances in $Y$, and $p' + q' \geq |X|$, then $|\{x_i : \chi_{A_1}(x_i) = 1\}| \leq \{|y_j : \chi_{A_2}(y_j) = 1\}|$ (note the inversion of the relationship between the two terms). Therefore, if this is the case, it suffices to guess the $p'$ “no”-instances in $X$, and the $q'$ “yes”-instances in $Y$. Guessing the $p'$ “no”-instances in $X$ is feasible by an NP machine (because $A_1$ is in a “complement” class), and guessing the $q'$ “yes”-instances in $Y$ is feasible in NP as well. Essentially, to answer “no”, if $A_1 \in \Pi^p_k$ and $A_2 \in \Sigma^p_k$, we do not need to actually count precisely the number of “yes”-instances in $X$ and $Y$.

**Theorem 3.5.** Let $A_1$ be a $\Pi^p_k$-complete problem, and $A_2$ be a $\Sigma^p_k$-complete problem. Then, for a given pair of sets $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_m\}$ of instances of $A_1$ and $A_2$, respectively, deciding whether $|\{x_i : \chi_{A_1}(x_i) = 1\}| > \{|y_j : \chi_{A_2}(y_j) = 1\}|$ is $\Pi^p_k$-complete.

**Proof.** Membership and hardness can be proven by a similar line of argumentation as in the proof of Theorem 3.4. We only have to prove, which is left to the reader, that $|\{x_i : \chi_{A_1}(x_i) = 1\} \leq \{|y_j : \chi_{A_2}(y_j) = 1\}|$ (i.e., if $p' + q' \geq |X|$). By exploiting this property, it can be shown that deciding whether $|\{x_i : \chi_{A_1}(x_i) = 1\}| > \{|y_j : \chi_{A_2}(y_j) = 1\}|$ is feasible in $\Pi^p_k$, and hence deciding $|\{x_i : \chi_{A_1}(x_i) = 1\}| > \{|y_j : \chi_{A_2}(y_j) = 1\}|$ is feasible in $\Pi^p_k$.

To conclude this part, we show the analogue of Lemma 2.2 for the new characterization of $\Theta^p_k$, which directly descends from Theorems 3.2 and 3.3 and provides a new sufficient condition for the $\Theta^p_{k+1}$-hardness of a problem. Intuitively, this states that, a problem $B$ is $\Theta^p_{k+1}$-hard, if there exists a reduction to $B$ from the problem of deciding, for a given pair of sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ of instances of two (not necessarily distinct) $\Sigma^p_k$-complete (or $\Pi^p_k$-complete) problems $A_1$ and $A_2$, respectively, whether the number of “yes”-instances of $A_1$ in $X$ is greater than the number of “yes”-instances of $A_2$ in $Y$. Similarly to Wagner’s result, this result holds even if it is assumed that $\chi_{A_1}(x_1) \geq \cdots \geq \chi_{A_1}(x_n)$ and $\chi_{A_2}(y_1) \geq \cdots \geq \chi_{A_2}(y_m)$ (and $n = m$).
Theorem 3.6. Let $A_1$ and $A_2$ be two $\Sigma^P_k$-complete (resp., $\Pi^P_k$-complete) problems, and let $B$ be a problem. Then, $B$ is $\Theta^P_{k+1}$-hard, if there exists a polynomial-time computable function $f$ such that, for all pair of sets $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_m\}$ of instances of $A_1$ and $A_2$, respectively, it holds that $|\{x_i : \chi_{A_1}(x_i) = 1\}| > |\{y_j : \chi_{A_2}(y_j) = 1\}| \Rightarrow \chi_B(f(X, Y)) = 1$. The $\Theta^P_{k+1}$-hardness of $B$ remains proven even if the sets $X$ and $Y$ are assumed to be such that $\chi_{A_1}(x_i) \geq \cdots \geq \chi_{A_2}(x_n)$ and $\chi_{A_2}(y_i) \geq \cdots \geq \chi_{A_2}(y_m)$, and $n = m$. 

Proof. From Theorems 3.2 and 3.3 deciding whether $|\{x : \chi_{A_1}(x) = 1\}| > |\{y : \chi_{A_2}(y) = 1\}|$ is $\Theta^P_{k+1}$-hard, both in the case that $A_1$ and $A_2$ are $\Sigma^P_k$-complete and in the case that $A_1$ and $A_2$ are $\Pi^P_k$-complete. Hardness holds even if $\chi_{A_1}(x_i) \geq \cdots \geq \chi_{A_2}(x_n)$ and $\chi_{A_2}(y_i) \geq \cdots \geq \chi_{A_2}(y_m)$, and $n = m$. Therefore, if there is a reduction (f) from the task of deciding $|\{x_i : \chi_{A_1}(x_i) = 1\}| > |\{y_j : \chi_{A_2}(y_j) = 1\}|$ to $B$, then $B$ is $\Theta^P_{k+1}$-hard as well. 

3.3 Complexity of $\text{Comp-Valid}_k$

We now prove $\text{Comp-Valid}_k \Theta^P_{k+1}$-complete. In the following, for a set $S$ of QBFs, $\#\text{val}(S)$ denotes the number of valid formulas in $S$. To show $\text{Comp-Valid}_k$’s membership in $\Theta^P_{k+1}$, we use two problems $\text{LBound-Valid}_{k,3}$ and $\text{UBound-Valid}_{k,\forall}$ defined as follows. A pair $(S, n)$, where $S$ is a set of QBFs with at most $k$ alternating quantifiers, whose outermost quantifier is $\exists$, and $0 \leq n \leq |S|$ is an integer, is a “yes”-instance of $\text{LBound-Valid}_{k,3}$ whenever $\#\text{val}(S) \geq n$. On the other hand, a pair $(S, n)$, where $S$ is a set of QBFs with at most $k$ alternating quantifiers, whose outermost quantifier is $\forall$, and $0 \leq n \leq |S|$ is an integer, is a “yes”-instance of $\text{UBound-Valid}_{k,\forall}$ whenever $\#\text{val}(S) \leq n$. We show that both these problems are feasible in $\Sigma^P_k$. Intuitively, this is the case, because, to decide $\text{LBound-Valid}_{k,3}$, it is sufficient to guess $n$ valid formulas in $S$ (feasible in NP), and then ask to a $\Pi^P_{k-1}$ oracle that the guessed formulas are actually valid. A similar intuition is also behind the complexity of $\text{UBound-Valid}_{k,\forall}$. Indeed, to decide $\text{UBound-Valid}_{k,\forall}$, it is sufficient to guess $|S| - n$ non-valid formulas in $S$ (feasible in NP), and then ask a $\Pi^P_{k-1}$ oracle that the guessed formulas are actually non-valid.

Lemma 3.7. $\text{LBound-Valid}_{k,3}$ and $\text{UBound-Valid}_{k,\forall}$ belong to $\Sigma^P_k$.

Proof. We first focus on $\text{LBound-Valid}_{k,3}$. Let $(S, n)$ be an instance of $\text{LBound-Valid}_{k,3}$. Since all the formulas $\Phi_i$ in $S$ are of the form $\Phi_i = (\exists X_1^i) \cdots (Q_P X^n_i)\phi_i(X_1^i, \ldots, X^n_p)$, with $p \leq k$, we can guess the indices $\{j_1, \ldots, j_m\}$ of a set of $n$ valid formulas, along with the complete assignments $\sigma_{X_1}$ over $X_1$, for each $\ell \in \{j_1, \ldots, j_m\}$, witnessing their validity. This guess is polynomial in size, and hence it can be carried out by an NP machine. Given such a guess, for each $\ell \in \{j_1, \ldots, j_m\}$, we can check the validity of $\Phi_i$ by checking the validity of $\Phi'_i = (\forall X_1^i) (\exists X^n_1) \cdots (Q_P X^n_i)\phi_i(X_1^i, \ldots, X^n_p)$ through a call to a $\Pi^P_{k-1}$ oracle (because $p - 1 \leq k - 1$). Clearly, the overall procedure is feasible in $\Sigma^P_k$.

For $\text{UBound-Valid}_{k,\forall}$, the proof is similar. Let $(S, n)$ be an instance of $\text{UBound-Valid}_{k,\forall}$. Since all the formulas $\Phi_i$ in $S$ are in the form $\Phi_i = (\forall X_1^i) \cdots (Q_P X^n_i)\phi_i(X_1^i, \ldots, X^n_p)$, we can guess the indices $\{j_1, \ldots, j_m\}$ of a set of $m = |S| - n$ non-valid formulas, along with the complete assignments $\sigma_{X_1}$ over $X_1$, for each $\ell \in \{j_1, \ldots, j_m\}$, witnessing the non-validity of the guessed formulas. Clearly, if there are $|S| - n$ non-valid formulas in $S$, then there at most $n$ valid formulas in $S$. Also this guess is polynomial in size, and hence it can be performed by an NP machine. Given such a guess, for each $\ell \in \{j_1, \ldots, j_m\}$, we can check that $\Phi_i$ is non-valid by checking that $\neg\Phi'_i = (\forall X_1^i) (\exists X^n_1) \cdots (Q_P X^n_i)\phi_i(X_1^i, \ldots, X^n_p)$ is valid through a call to a $\Pi^P_{k-1}$ oracle (because $p - 1 \leq k - 1$). Also in this case, the overall procedure is feasible in $\Sigma^P_k$.

We are now ready to show that $\text{Comp-Valid}_k$ is $\Theta^P_{k+1}$-complete. We first prove its membership in $\Theta^P_{k+1}$. Intuitively, $\text{Comp-Valid}_k$ is in $\Theta^P_{k+1}$ because, for its solution, it is sufficient to count the number of valid formulas in the sets $A$ and $B$. This can be done by a binary search exploiting two oracles for $\text{LBound-Valid}_{k,3}$ and $\text{UBound-Valid}_{k,\forall}$, respectively. The former is needed to count valid quantified formulas whose outermost quantifier is $\exists$, while the latter is used to count valid quantified formulas whose outermost quantifier is $\forall$.

Theorem 3.8. Let $A$ and $B$ be two sets of QBFs with at most $k$ alternating quantifiers. Then, deciding whether $\#\text{val}(A) > \#\text{val}(B)$ is feasible in $\Theta^P_{k+1}$.

Proof. Let $(A, B)$ be an instance of $\text{Comp-Valid}_k$. To decide whether $\#\text{val}(A) > \#\text{val}(B)$, we count the number of valid formulas in $A$ and in $B$, and then compare the numbers. Consider the set $A$. We can partition $A$ in two subsets $A_3$ and $A_4$, containing the formulas whose outermost quantifier is $\exists$ and $\forall$, respectively. Clearly, $|A| = |A_3| + |A_4|$. We can count the number of valid formulas belonging to $A_3$ and $A_4$, via a binary search in the range $[0, |A|]$, by a Turing machine in $P$ querying an oracle for $\text{LBound-Valid}_{k,3}$ and $\text{UBound-Valid}_{k,\forall}$, respectively. Observe that it is sufficient to devise a single oracle with a variable in the input to ask the oracle to decide either $\text{LBound-Valid}_{k,3}$ or $\text{UBound-Valid}_{k,\forall}$. The number of valid formulas of $B$ can be computed
similarly. Notice that the number of queries submitted to the oracle is logarithmic in the size of the input, and that LBOUND-VALID_{k, \exists} and UBOUND-VALID_{k, \forall} belong to \Sigma^p_{n} (see Lemma 3.7). Thus, the overall procedure is feasible in P^{\Sigma^p_{n}[O(\log n)]} = \Theta^p_{k+1}.

We next prove that COMP-VALID_{k} is \Theta^p_{k+1}-hard. The reduction to show the \Theta^p_{k+1}-hardness of COMP-VALID_{k} is a direct application of Theorems 3.2 and 3.3. In fact, counting and comparing the number of valid QBFs of two given sets is essentially counting and comparing the number of “yes”-instances in the two sets containing instances of the problems QBF^C_{k,3} (or QBF^DNF_{k,3}) Furthermore, we also prove that the hardness of COMP-VALID_{k} holds even if \( |A| = |B| \), all formulas in \( (A, B) \) are instances of QBF^C_{k,3} (resp., QBF^DNF_{k,3}), have the same number of clauses (resp., terms), and, for each \( 1 \leq d \leq k \), quantifiers \( Q_d \) of all formulas in \( (A, B) \) are defined on the very same set of variables. To prove that this restriction on the structure of the formulas does not influence the hardness of the problem, we show that a generic instance of COMP-VALID_{k} can always be rewritten in polynomial time in an instance fulfilling the required constraints. Note that, in the following proof showing the \( \Theta^p_{k+1} \)-completeness of COMP-VALID_{k}, for the hardness to hold, it is required that the number of formulas in the sets \( A \) and \( B \) with actually \( k \) alternating quantifiers is unbounded.

**Theorem 3.9.** Let \( A \) and \( B \) be two sets of QBFs with at least \( k \) alternating quantifiers. Then, deciding whether \#val(A) > \#val(B) is \( \Theta^p_{k+1} \)-hard. Hardness holds even if \( |A| = |B| \), all formulas in \( (A, B) \) are instances of QBF^C_{k,3} (resp., QBF^DNF_{k,3}), have the same number of clauses (resp., terms), and, for each \( 1 \leq d \leq k \), quantifiers \( Q_d \) of all formulas in \( (A, B) \) are defined on the very same set of variables.

**Proof.** We first show that hardness of COMP-VALID_{k} holds even over the class \( \mathcal{I} \) of instances \( (A, B) \) such that \( |A| = |B| \), and all formulas in \( A \) and \( B \) are instances of QBF^C_{k,3} (resp., QBF^DNF_{k,3}). In particular, for the class \( \mathcal{I} \), we neither restrict formulas of the instances in \( \mathcal{I} \) to have the same number of clauses (resp., terms), nor their quantifiers to be defined on the same set of variables. We prove hardness by applying Theorems 3.2 and 3.3.

QBF^C_{k,3} for odd \( k \) (resp., QBF^DNF_{k,3} for even \( k \)) is \( \Sigma^p_{k} \)-complete, and, given any two sets \( X = \{ x_1, \ldots, x_n \} \) and \( Y = \{ y_1, \ldots, y_m \} \) of instances of QBF^C_{k,3} for odd \( k \) (resp., QBF^DNF_{k,3} for even \( k \)), clearly the pair \( (X, Y) \) itself is a “yes”-instance of COMP-VALID_{k} if and only if \( \{ (x_i : \chi_{A_i}(x_i) = 1) \} \neq \{ (y_j : \chi_{A_j}(y_j) = 1) \} \).

Hence, Theorem 3.2 applies (with \( n = m \)), and thus COMP-VALID_{k} is \( \Theta^p_{k+1} \)-hard. Furthermore, QBF^C_{k,3} for odd \( k \) (resp., QBF^DNF_{k,3} for even \( k \)) is \( \Pi^p_{k} \)-complete, and the line of argumentation is similar, applying Theorem 3.3, instead of Theorem 3.2.

Consider now the special case in which formulas of the instances are restricted to have the same number of clauses (resp., terms), and their quantifiers are restricted to be defined over the same set of variables. We show that a generic instance of COMP-VALID_{k} in \( \mathcal{I} \) can be reduced in polynomial time to an instance satisfying these restrictions. The idea is to rewrite formulas considering first each quantifier in turn, from the outermost to the innermost, in order to have the quantifiers defined over the very same sets of variables. After the formulas are rewritten so that all quantifiers are defined over the same sets of variables, if necessary, the number of clauses (resp., terms) among the formulas is made equal. Note that there is no need to perform any “balancing” of the number of quantifiers”, since all formulas of the instances in \( \mathcal{I} \) are instances of QBF^C_{k,3} (resp., QBF^DNF_{k,3}), and hence all of them have the same number of nested quantifiers. In the following, we assume that all QBFs are instances of QBF^C_{k,3}, i.e., all QBFs are of the form \( \Phi = (Q_1 X_1') \cdot \ldots \cdot (Q_k X_k') \phi_k(X_1', \ldots, X_k') \) with \( Q_k = \exists \) and \( \phi_k \) being in CNF. The case that all QBFs are instances of QBF^DNF_{k,3} can be proven in a similar way.

First, let us consider the procedure to have the quantifiers defined over the same sets of variables. Quantifiers are processed from the outermost to the innermost, and exactly one quantifier is processed at a time. Assume that \( Q_d \) is the currently considered quantifier.

If necessary, we balance the number of variables in the sets \( X'_d \) among the various formulas \( \Phi_i \). If there are two formulas \( \Phi_i, \Phi_j \in A \cup B \) such that \( |X'_d| \neq |X'_d| \), then we compute max_{d} = max_{\Phi \in A \cup B} |X'_d|. Subsequently, we rewrite all formulas \( \Phi_i \) with \( |X'_d| < max_{d} \) by: (i) extending \( X'_d \) to \( X'_d \) by adding fresh variables, so that \( |X'_d| = max_{d} \); (ii) adding to \( \phi_i \), for each variable \( x \in X'_d \), dummy satisfiable clauses of three literals \( (x \lor \neg x \lor x) \), so that the new variables appear in the non-quantified part. Since \( (\exists x)(x \lor \neg x \lor x) \) and \( (\forall x)(x \lor \neg x \lor x) \) are both always valid, adding clauses like \( (x \lor \neg x \lor x) \) does not alter the validity of the formulas, irrespective of the considered quantifier \( Q_d \) being \( \exists \) or \( \forall \), or \( k \) being even or odd. At the end of this procedure, all quantifiers are defined over the same number of variables. Now, if necessary, we can rename the variables so that those sets contain the very same elements. We denote by \( \phi^d \) the new non-quantified part of the formulas obtained after the rewriting, while \( \Phi_i \) denotes the new quantified formulas after the rewriting.

In a second phase, if necessary, we balance the number of clauses in the various formulas. If there are two formulas \( \Phi_i, \Phi_j \in A \cup B \) such that the number of clauses in \( \phi^d \) is different from the number of clauses in \( \phi^d \), then we compute max_{i,j} = max_{\Phi \in A \cup B} |\text{number of clauses of } \phi_i^d| \). After this, we rewrite all formulas \( \Phi_i \) whose formula \( \phi^d \) has less clauses than max_{i,j} by adding dummy satisfiable clauses of three literals \( (x \lor \neg x \lor x) \), where
The selected problem is the Max voting scheme over value), I prefer a white wine (\(|\) defined over the very same set of variables, having the same number of clauses, and such that mentioned. With these nets, it is possible to show that deciding Max dominance is for two specific outcomes agents of \(\alpha\) and \(\beta\). Note that, although the features of the individual CP-nets are the same, the graph structure of the individual features which, in turn, have the same possible values. The "CP-nets, each agent votes whether an outcome is preferred to another, and different ways of collecting votes (i.e., groups of agents in CP-nets. CP-nets \[2\] are a graphical preference model, and they are among the most studied paper provides a powerful tool to prove \(\Theta_p^k\)-hardness of problems whose semantics is tightly linked to the one of counting and comparison. In particular, we select a problem taken from the area of computational social choice. The selected problem is the Max voting scheme over m-CP-nets, which are a tool to represent preferences in groups based on CP-nets. CP-nets \[2\] are a graphical preference model, and they are among the most studied preference models, as the vast literature on them demonstrates. In CP-nets, graph vertices represent features, and an edge from vertex \(A\) to vertex \(B\) models that \(A\)'s value influences the choice of \(B\)'s value. Intuitively, an outcome is a particular configuration of the features in the domain at hand, i.e., an outcome is an object assigning a value to every feature. For a CP-net \(N\), \(\beta \succ_N \alpha\) denotes that the outcome \(\beta\) is preferred to the outcome \(\alpha\) according to the preferences modeled in \(N\), and \(\beta \succ_N \alpha\) denotes that \(\beta\) and \(\alpha\) are incomparable in \(N\).

In mCP-nets, a set of CP-nets is used to model the preferences of each agent in a group. Preferences for groups of agents in m-CP-nets are defined through voting schemes. In particular, through their own individual CP-nets, each agent votes whether an outcome is preferred to another, and different ways of collecting votes (i.e., different voting schemes) give rise to different dominance semantics for m-CP-nets. Various voting schemes were proposed for m-CP-nets \[28\] \[24\], and here we focus on the Max voting scheme \[28\].

More precisely, an m-CP-net \(\mathcal{M}\) is a collection \((N_1, \ldots, N_m)\) of m-CP-nets defined over the same set of features which, in turn, have the same possible values. The "\(m\)" of an m-CP-net stands for "multiple" agents and also indicates that the preferences of \(m\) agents are modeled in the net, so a 3CP-net is an m-CP-net with \(m = 3\). Note that, although the features of the individual CP-nets are the same, the graph structure of the individual nets may be different, i.e., the links between the features in the various individual CP-nets may vary.

Let \(\mathcal{M} = (N_1, \ldots, N_m)\) be an m-CP-net, and let \(\alpha, \beta\) be two outcomes. We define the sets \(S^\mathcal{M}_\alpha(\alpha, \beta) = \{i \mid \alpha \succ_N i \succ \beta\}\), \(S^\mathcal{M}_\alpha(\beta, \alpha) = \{i \mid \beta \succ_N i \succ \alpha\}\), and \(S^\mathcal{M}_\alpha(\beta, \alpha) = \{i \mid \alpha \succ_N i \succ \beta\}\), which are the sets of the agents of \(\mathcal{M}\) preferring \(\alpha\) to \(\beta\), preferring \(\beta\) to \(\alpha\), and for which \(\alpha\) and \(\beta\) are incomparable, respectively. The Max voting is defined as follows: the outcome \(\beta\) maximizes the outcome \(\alpha\), denoted by \(\beta \succ^\mathcal{M} \alpha\), if the group of the agents of \(\mathcal{M}\) preferring \(\beta\) to \(\alpha\) is the biggest, i.e., \(|S^\mathcal{M}_\alpha(\beta, \alpha)| > \max(|S^\mathcal{M}_\alpha(\alpha, \beta)|, |S^\mathcal{M}_\alpha(\beta, \alpha)|)\).

For any given outcome \(\gamma\), it is possible to design a CP-net \(D(\gamma)\) for which \(\gamma\) is the optimum outcome \[25\]. For any 3CNF Boolean formula \(\phi\), it is possible to design two different CP-nets, \(F(\phi)\) and \(F(\phi)\), such that, for two specific outcomes \(\alpha\) and \(\beta\), \(\beta \succ^\mathcal{M} \alpha\) (resp., \(\alpha \succ^\mathcal{M} \beta\)) if and only if \(\phi\) is satisfiable, and \(\beta \succ^\mathcal{M} \alpha\) (resp., \(\alpha \succ^\mathcal{M} \beta\)) if and only if \(\phi\) is unsatisfiable \[25\]. Below, we will refer again to the outcomes \(\alpha\) and \(\beta\) just mentioned. With these nets, it is possible to show that deciding Max dominance is \(\Theta_p^k\)-hard.

Consider a generic instance \((A, B)\) of COMP-SAT, where \(A\) and \(B\) are two sets of 3CNF Boolean formulas defined over the very same set of variables, having the same number of clauses, and such that \(|A| = a\) and \(|B| = b\). From \((A, B)\), it is possible to build a \(3(a + b)\)-CP-net \(M_{\max}(\alpha, \beta)\) such that \(\beta \succ^\mathcal{M} \alpha\) if and only if \((\alpha, \beta)\) is a "yes"-instance of COMP-SAT. In particular, the agents of \(M_{\max}(\alpha, \beta)\) are:

- for each formula \(\phi\) in \(A\), there is an agent whose CP-net is \(N_{A, \phi} = F(\phi)\);
- for each formula \(\varphi\) in \(B\), there is an agent whose CP-net is \(N_{B, \varphi} = T(\varphi)\);
- there are \(a + b\) agents whose preferences are encoded by the CP-net \(D(\alpha)\); and
- there are \(a + b\) agents whose preferences are encoded by the CP-net \(D(\beta)\).

Given the above construction it is possible to show the following.

**Theorem 4.1 (25).** Let \(\mathcal{M}\) be an m-CP-net, and let \(\alpha\) and \(\beta\) be two outcomes. Deciding whether \(\beta \succ^\mathcal{M} \alpha\) is \(\Theta_p^k\)-hard.
5 Conclusion

In this paper, we have introduced a new characterization for the class $\Theta_2^P$ in general and for $\Theta_2^P$ in particular. We have shown that problems belonging to $\Theta_2^P$ are also those involving the task of counting and comparing the number of “yes”-instances of two sets $A$ and $B$ of $\Sigma_k^P$-complete (or $\Pi_k^P$-complete) problem instances. Moreover, we have also shown that this new characterization is sufficient to entail the $\Theta_{k+1}^P$-hardness of the problem at hand. In fact, if the problem involves the task of counting and comparing the number of “yes”-instances of two sets of $\Sigma_k^P$-complete (or $\Pi_k^P$-complete) problems instances, then the problem is $\Theta_{k+1}^P$-hard. On the other hand, we have proven that this problem becomes computationally easier when the instances of sets $A$ and $B$ are of a $\Sigma_k^P$-complete and of a $\Pi_k^P$-complete problem, respectively.

We have complemented this work by providing also the $\Theta_{k+1}^P$-complete problem $\text{COMP-VALID}_k$, which is deciding, given two sets $A$ and $B$ of quantified Boolean formulas with $k$ alternating quantifiers, whether the number of valid formulas in $A$ is greater than the number of valid formulas in $B$. The results shown here prove that its specialization $\text{COMP-VALID}_1$, when only existentially quantified CNF Boolean formulas are considered, ($= \text{COMP-SAT}$), is the very natural and intuitive problem of deciding whether the number of satisfiable CNF Boolean formulas of a set is bigger than the number of satisfiable CNF Boolean formulas of another set, is $\Theta_2^P$-complete. $\text{COMP-VALID}_k$ (resp., $\text{COMP-SAT}$) proves to be an ideal candidate when one needs a reduction to show $\Theta_{k+1}^P$-hardness (resp., $\Theta_2^P$-hardness) of a problem involving the task of counting and comparison. In fact, the $\Theta_2^P$-hardness of $\text{COMP-SAT}$ is an easy corollary of the $\Theta_{k+1}^P$-hardness of $\text{COMP-VALID}_k$, and it was successfully used to easily prove the $\Theta_2^P$-hardness of a voting problem in \cite{25}.

Acknowledgments. This work was supported by the UK EPSRC grants EP/J008346/1, EP/L012138/1, and EP/M025268/1, and by The Alan Turing Institute under the EPSRC grant EP/N510129/1. We thank Dominik Peters and the anonymous reviewers for their helpful comments on a preliminary version of the paper.

References

[1] T. Baker, J. Gill, and R. Solovay. “Relativizations of the $P = \text{NP}$ Question”. In: SIAM J. Comput. 4.4 (1975), pp. 431–442.
[2] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. “CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements”. In: J. Artif. Intell. Res. 21 (2004), pp. 135–191.
[3] S. R. Buss and L. Hay. “On truth-table reducibility to SAT and the difference hierarchy over NP”. In: Proc. of CoCo. 1988, pp. 224–233.
[4] S. R. Buss and L. Hay. “On truth-table reducibility to SAT”. In: Inform. Comput. 91.1 (1991), pp. 86–102.
[5] S. A. Cook. “The Complexity of Theorem-Proving Procedures”. In: Proc. of STOC. 1971, pp. 151–158.
[6] T. Eiter and G. Gottlob. “The complexity class $\Theta_2^P$: Recent results and applications in AI and modal logic”. In: Proc. of FCT. Vol. 1279. LNCS. Springer, 1997, pp. 1–18.
[7] T. Eiter and T. Lukasiewicz. “Default reasoning from conditional knowledge bases: Complexity and tractable cases”. In: Artif. Intell. 124.2 (2000), pp. 169–241.
[8] M. R. Garey and D. S. Johnson. Computers and Intractability. A Guide to the Theory of NP-Completeness. New York, NY, USA: W. H. Freeman and Company, 1979.
[9] G. Gottlob. “NP Trees and Carnap’s Modal Logic”. In: J. ACM 42.2 (1995), pp. 421–457.
[10] G. Gottlob. “Relativized Logspace and Generalized Quantifiers over Finite Ordered Structures”. In: J. Symb. Logic 62.2 (1997), pp. 545–574.
[11] G. Gottlob and E. Malizia. “Achieving New Upper Bounds for the Hypergraph Duality Problem through Logic”. In: Proc. of CSL-LICS. 2014, 43:1–43:10.
[12] G. Gottlob and E. Malizia. Achieving New Upper Bounds for the Hypergraph Duality Problem through Logic. Tech. rep. arXiv:1407.2912. Computing Research Repository (CoRR), 2014.
[13] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. “Non-Transferable Utility Coalition Games via Mixed-Integer Linear Constraints”. In: J. Artif. Intell. Res. 38 (2010), pp. 633–685.
[14] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. “On the complexity of core, kernel, and bargaining set”. In: Artif. Intell. 175.12–13 (2011), pp. 1877–1910.
