A novel characterization of the complexity class Θ_k^{P} 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 Θ_2^{P}, 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 Θ_k^{P} to the k-th level of the polynomial hierarchy. We show that problems in Θ_k^{P} are also those whose solution involves deciding, for two given sets A and B of instances of two $\Sigma_{k-1}^{\mathrm{P}}$-complete (or Π_{k-1}^{P}-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 Θ_k^{P}-hardness. We also define the general problem COMP-VALID$_k$, which is proven here $\Theta_{k+1}^{\mathrm{P}}$-complete. 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 COMP-SAT of deciding whether a set contains more satisfiable Boolean formulas than another set, which is a particular case of COMP-VALID$_1$, demonstrates itself as a very intuitive Θ_2^{P}-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, COMP-VALID$_k$ is among the most suitable tools to prove Θ_k^{P}-hardness of problems involving the counting and comparison of the number of "yes"-instances in two sets. We support this by showing that the Θ_2^{P}-hardness of the Max voting scheme over mCP-nets is easily obtained via the new characterization of Θ_k^{P} 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 Δ_k^{P}, Σ_k^{P}, and Π_k^{P}, for $k \geq 0$, constitute the hierarchy: $\Delta_0^{\mathrm{P}} = \Sigma_0^{\mathrm{P}} = \Pi_0^{\mathrm{P}} = \mathrm{P}$, and, for $k \geq 1$, $\Delta_{k+1}^{\mathrm{P}} = \mathrm{P}^{\Sigma_k^{\mathrm{P}}}$, $\Sigma_{k+1}^{\mathrm{P}} = \mathrm{NP}^{\Sigma_k^{\mathrm{P}}}$, and $\Pi_{k+1}^{\mathrm{P}} = \text{co-}\Sigma_{k+1}^{\mathrm{P}}$ (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 ODD-CLIQUE [35], which is the problem of deciding whether the size of the largest cliques of a graph is odd. Clearly, ODD-CLIQUE is in Δ_2^{P}, because a P machine can decide ODD-CLIQUE via a binary search aided by an oracle in NP. However, ODD-CLIQUE is not Δ_2^{P}-hard. In fact, only logarithmically many queries need to be issued to the oracle, and this could be a clue for ODD-CLIQUE not being among the hardest problems of Δ_2^{P}, which, instead, for their solution require P machines performing polynomially many calls to their NP oracles. Therefore, to precisely

 The formal publication of this manuscript is available via the DOI: 10.1016/j.tcs.2017.06.023.

[1]A catalogue of problems complete for various levels of PH can be found in [30, 31] and its updated revision [32].

capture this kind of problems, the class of languages that can be recognized by a P machine issuing at most logarithmic many calls to an NP oracle, also denoted $\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ (where "$[O(\log n)]$" denotes the logarithmic restriction on the maximum number of allowed oracle calls), was defined [27, 23, 21, 37, 4].

Unlike the nondeterministic levels $\Sigma_k^{\mathrm{P}} = \mathrm{NP}^{\Sigma_{k-1}^{\mathrm{P}}}$ 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., $\mathrm{NP}^{\Sigma_{k-1}^{\mathrm{P}}[O(1)]} = \mathrm{NP}^{\Sigma_{k-1}^{\mathrm{P}}[O(\log n)]} = \mathrm{NP}^{\Sigma_{k-1}^{\mathrm{P}}}$ [37], for the deterministic levels of the hierarchy, this does not seem to be the case. In fact, clearly, $\mathrm{P}^{\Sigma_{k-1}^{\mathrm{P}}[O(1)]} \subseteq \mathrm{P}^{\Sigma_{k-1}^{\mathrm{P}}[O(\log n)]} \subseteq \mathrm{P}^{\Sigma_{k-1}^{\mathrm{P}}}$, however, it is currently unknown whether the inclusion is strict. On the other hand, Odd-Clique was proven to be complete for the class $\Theta_2^{\mathrm{P}} = \mathrm{P}^{\mathrm{NP}[O(\log n)]}$ [35]. This supported the widely accepted conjectures that $\Theta_2^{\mathrm{P}} \neq \Delta_2^{\mathrm{P}}$ and that, for all $k \geq 2$, $\Theta_k^{\mathrm{P}} \neq \Delta_k^{\mathrm{P}}$ [37], where $\Theta_k^{\mathrm{P}} = \mathrm{P}^{\Sigma_{k-1}^{\mathrm{P}}[O(\log n)]}$, which suggested to include the classes Θ_k^{P} as constitutional components of PH [36, 37].

In recent years, many natural problems have been shown to be complete for Θ_2^{P} (see, e.g., [35, 23, 21, 37, 6, 7, 33] and references therein).[2] These Θ_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 Θ_2^{P}-complete (resp., Θ_k^{P}-complete) problem is based on directed trees whose nodes are parametric queries to an NP (resp., $\Sigma_{k-1}^{\mathrm{P}}$) oracle, where the directed tree encodes the dependence structure among the oracle queries [9]. Furthermore, Θ_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 Θ_2^{P} [29, 19, 18, 25]. These complexity results come as no surprise to us, since, as we will show in this paper, Θ_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).[3] A result suggesting that problems in Θ_2^{P} can be characterized in this way appeared only in [33]. However, to our knowledge, in the literature, 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 Θ_k^{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 $\mathcal{G} = \{\langle G_1, p_1\rangle, \ldots, \langle G_n, p_n\rangle\}$ and $\mathcal{H} = \{\langle H_1, q_1\rangle, \ldots, \langle H_m, q_m\rangle\}$ of pairs $\langle \mathit{graph}, \mathit{integer}\rangle$, 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 Θ_k^{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 $\Sigma_{k-1}^{\mathrm{P}}$-complete or Π_{k-1}^{P}-complete languages, are Θ_k^{P}-hard. To our knowledge, this characterization of Θ_k^{P} is new in the literature, and also its specialization to Θ_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 Θ_k^{P}-hard. Interestingly, we also show that problems requiring the comparison of the number of "yes"-instances of a set containing instances of a $\Sigma_{k-1}^{\mathrm{P}}$-complete problem and a set containing instances of a Π_{k-1}^{P}-complete problem are computationally easier, since they can be solved in subclasses of Θ_k^{P} (more specifically, in $\Sigma_{k-1}^{\mathrm{P}}$ or Π_{k-1}^{P}).

Furthermore, by exploiting the characterization given in this paper, we also define the following general $\Theta_{k+1}^{\mathrm{P}}$-complete problem Comp-Valid$_k$ (observe the different subscripts, $k+1$ and k, respectively): given a pair $\langle A, B\rangle$ 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 $\Theta_{k+1}^{\mathrm{P}}$-complete.

In addition, Comp-Sat, which is a particular case of Comp-Valid$_1$, is a very intuitive Θ_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 Θ_2^{P}-hardness of problems involving the comparison of the number of "yes"-instances of two sets containing instances of NP-complete problems. Comp-Sat is the first Θ_2^{P}-complete problem of this kind. In fact, it was successfully used to prove the Θ_2^{P}-hardness of a voting problem

[2] Note that in the literature, Θ_2^{P} was proven to be equivalently characterized by different definitions (see, e.g., [6, 37]), and thus Θ_2^{P}-complete problems are often shown to be complete for apparently different (but actually identical) classes.

[3] Interestingly, several problems in computational game theory are complete for various classes of PH and Δ_2^{P} in particular. Among them, there are a number of different tasks related to the solution concepts of coalitional games (see, e.g., [13, 14, 15, 16]).

in [25], and, given its adherence to the counting-and-comparison semantics, this was fairly simple. In this respect, we actually believe that COMP-VALID$_k$ is the ideal problem when a reduction is needed for a $\Theta^{\mathrm{P}}_{k+1}$-hard problem involving counting and comparison. In this paper, the Θ^{P}_2-completeness of COMP-SAT comes as an easy corollary of the $\Theta^{\mathrm{P}}_{k+1}$-completeness of COMP-VALID$_k$, and we do not need a tailored reduction as in [25].

The rest of this paper is organized as follows. Section 2 provides some preliminaries on complexity theory. In Section 3, after an overview of this paper's results, we analyze the new characterization of Θ^{P}_k, and we also prove that COMP-VALID$_k$ is $\Theta^{\mathrm{P}}_{k+1}$-complete. In Section 4, we show how the results presented here can be easily applied to prove the Θ^{P}_2-hardness of the Max voting scheme over mCP-nets. Finally, Section 5 is devoted to conclusions.

2 Preliminaries

In this section, we briefly recall some basics from complexity theory on decision problems, the complexity classes of the polynomial hierarchy (PH), and their prototypical hard problems. For more on this, the reader is referred to any standard textbook on the topic, such as [26], or the survey [20].

2.1 Decision problems and complexity classes

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, χ_L denotes the *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 $\chi_L(s)=1$. *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 *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 $\mathcal{C}$, co-$\mathcal{C}$ denotes the class of languages whose complements are in $\mathcal{C}$. With a slight abuse of terminology, we say that a Turing machine M belongs to a complexity class $\mathcal{C}$, if M is of the sort and uses the amount of computational resources characterizing the class $\mathcal{C}$.

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 *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 $X=\{x_1,\dots,x_n\}$ is satisfiable, i.e., whether there exists some truth assignment to these variables making ϕ `true`, is a well-known problem in NP; in fact, any satisfying truth assignment for ϕ is clearly a certificate that ϕ is a "yes"-instance, i.e., that ϕ is satisfiable. On the other hand, the problem of deciding whether a Boolean formula ϕ is *not* satisfiable is in co-NP. Clearly, the class P is contained in both NP and co-NP, i.e., $\mathrm{P}\subseteq\mathrm{NP}\cap\text{co-NP}$.

We will also refer to a type of computation called computation with *oracles*. Intuitively, oracles are subroutines that are supposed to have unit cost. A Turing machine $M^?$ 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 $M^?$ is independent from its oracle, and the symbol "?" indicates that different oracles for different languages can be "attached" to M [26]. If A is a language, by M^A we denote that the oracle attached to $M^?$ decides A. If $\mathcal{C}$ is a (deterministic or a nondeterministic) time complexity class, and A is a language, $\mathcal{C}^A$ denotes the class of languages that can be decided by Turing machines of the sort and the time bound of $\mathcal{C}$ that can moreover query an oracle for A.[4] By extension, for a time complexity class $\mathcal{C}$, and a generic complexity class $\mathcal{D}$, $\mathcal{C}^{\mathcal{D}}$ denotes

[4] If $\mathcal{C}$ is a space complexity class, it is more difficult to define computation with oracles [26, 17].

the class of languages that can be decided by Turing machines of the sort and the time bound of $\mathcal{C}$ that can moreover query an oracle for a language in $\mathcal{D}$. In the following, when we say that a Turing machine M queries an oracle in $\mathcal{D}$, or a $\mathcal{D}$ oracle, we mean that M queries an oracle for a language in $\mathcal{D}$.

The classes Σ_k^{P}, Π_k^{P}, and Δ_k^{P}, forming the *polynomial hierarchy (PH)* [34], are defined as follows: $\Sigma_0^{\mathrm{P}} = \Pi_0^{\mathrm{P}} = \Delta_0^{\mathrm{P}} = \mathrm{P}$, and, for all $k \geq 1$, $\Sigma_k^{\mathrm{P}} = \mathrm{NP}^{\Sigma_{k-1}^{\mathrm{P}}}$, $\Delta_k^{\mathrm{P}} = \mathrm{P}^{\Sigma_{k-1}^{\mathrm{P}}}$, and $\Pi_k^{\mathrm{P}} = \text{co-}\Sigma_k^{\mathrm{P}}$. Here, Σ_k^{P} (resp., Δ_k^{P}) 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 $\Sigma_{k-1}^{\mathrm{P}}$. Note that $\Sigma_1^{\mathrm{P}} = \mathrm{NP}^{\Sigma_0^{\mathrm{P}}} = \mathrm{NP}^{\mathrm{P}} = \mathrm{NP}$, $\Pi_1^{\mathrm{P}} = \text{co-}\Sigma_0^{\mathrm{P}} = \text{co-NP}$, and $\Delta_1^{\mathrm{P}} = \mathrm{P}^{\Sigma_0^{\mathrm{P}}} = \mathrm{P}^{\mathrm{P}} = \mathrm{P}$. Sometimes, a bound is imposed on the number of the allowed oracle calls, highlighted in brackets besides the oracle class. For example, $\mathrm{P}^{\Sigma_{k-1}^{\mathrm{P}}[O(\log n)]}$ denotes the class of languages recognizable by a deterministic polynomial-time Turing machine that is allowed to query a $\Sigma_{k-1}^{\mathrm{P}}$ oracle at most logarithmic many times (in the size of the input). In particular, classes $\Theta_k^{\mathrm{P}} = \mathrm{P}^{\Sigma_{k-1}^{\mathrm{P}}[O(\log n)]}$ were proposed to be included in the standard definition of the PH as well [36, 37]. To this end, we pose $\Theta_0^{\mathrm{P}} = \mathrm{P}$, and observe that $\Theta_1^{\mathrm{P}} = \mathrm{P}$. Note that Θ_k^{P} and Δ_k^{P} are deterministic classes, as the machine calling the oracle is deterministic. This implies that Θ_k^{P} and Δ_k^{P} are closed under complement, i.e., $\Theta_k^{\mathrm{P}} = \text{co-}\Theta_k^{\mathrm{P}}$ and $\Delta_k^{\mathrm{P}} = \text{co-}\Delta_k^{\mathrm{P}}$. Given their definitions, for all $k \geq 1$, the relationships among the mentioned classes are as follows (see, e.g., [36, 37]): $\Sigma_k^{\mathrm{P}} \cup \Pi_k^{\mathrm{P}} \subseteq \Theta_{k+1}^{\mathrm{P}} \subseteq \Delta_{k+1}^{\mathrm{P}} \subseteq \Sigma_{k+1}^{\mathrm{P}} \cap \Pi_{k+1}^{\mathrm{P}}$.

2.2 Prototypical hard problems

We now recall the notion of reducibility among decision problems. A decision problem L_1 is *(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 *polynomially (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 *hard* for a class $\mathcal{C}$ of the PH at any level $k \geq 1$, i.e., beyond P, or $\mathcal{C}$-hard, if every problem in $\mathcal{C}$ is polynomially reducible to L; if L is hard for $\mathcal{C}$ and belongs to $\mathcal{C}$, then L is *complete* for $\mathcal{C}$, or $\mathcal{C}$-complete. Thus, problems that are complete for $\mathcal{C}$ are the most difficult problems in $\mathcal{C}$. 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 $f\colon \{\mathtt{true}, \mathtt{false}\}^n \mapsto \{\mathtt{true}, \mathtt{false}\}$ from the n-dimensional Boolean space to a Boolean value. A way to represent n-ary Boolean functions is through *Boolean formulas* $\phi(X)$ over the set of Boolean variables $X = \{x_1, \dots, x_n\}$. Boolean formulas are inductively constructed from Boolean variables via the unary Boolean operator $\neg$ and the binary Boolean operators $\wedge$ and $\vee$. Boolean variables $x_1, \dots, x_n$ and their negations $\neg x_1, \dots, \neg x_n$ are called *literals*. A *clause* and a *term* are a disjunction and a conjunction of literals, respectively. A Boolean formula is in *conjunctive normal form* (or *CNF*), if it is a conjunction of clauses, while it is in *disjunctive normal form* (or *DNF*), if it is a disjunction of terms. For example, $\gamma_1(x_1, x_2, x_3, x_4) = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$ is a CNF formula, while $\gamma_2(x_1, x_2, x_3, x_4) = (x_1 \wedge \neg x_2 \wedge x_3) \vee (x_2 \wedge \neg x_3 \wedge \neg x_4)$ is a DNF formula.[5] A Boolean formula is in 3*CNF* (resp., 3*DNF*), if the number of literals of each clause (resp., term) is exactly three.

Deciding the satisfiability of Boolean formulas is the prototypical NP-complete problem, which remains NP-hard even if only 3CNF formulas are considered [8, 22]; we denote this problem by Sat. The complementary problem Unsat of deciding whether a given Boolean formula is *not* satisfiable is co-NP-complete. It remains co-NP-hard even if only 3CNF formulas are considered, and it is the equivalent to the problem Taut of deciding whether a 3DNF formula is a tautology.

We next define the prototypical Σ_k^{P}- and Π_k^{P}-complete $\mathrm{QBF}_{Q_1,k}$ problems as follows: given a quantified Boolean formula (QBF) $\Phi = (Q_1 X_1)(Q_2 X_2) \dots (Q_k X_k)\phi(X_1, X_2, \dots, X_k)$, where

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

decide whether Φ is valid. The problem $\mathrm{QBF}_{\exists,k}$ is Σ_k^{P}-complete [34, 38], while $\mathrm{QBF}_{\forall,k}$ is Π_k^{P}-complete [34, 38]. These problems remain hard for their respective classes even if $\phi(X_1, X_2, \dots, X_k)$ is in 3CNF, when $Q_k = \exists$, and if $\phi(X_1, X_2, \dots, X_k)$ is in 3DNF, when $Q_k = \forall$ [34, 38].

[5] 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 equivalent. Nevertheless, Boolean formulas in CNF and DNF are actually linked, and they can be transformed from one form to the other. In particular, translating a positive CNF formula into an equivalent minimal DNF one (or vice versa) involves a process called *dualization*, which is currently unknown to be feasible in output-polynomial time (for more on this, see, e.g., [11, 12] and references therein).

We denote by $\mathrm{QBF}_{k,\exists}^{CNF}$ (resp., $\mathrm{QBF}_{k,\forall}^{DNF}$)[6] the problem of deciding the validity of formulas $\Phi = (Q_1 X_1) \dots (Q_k X_k)\phi(X_1, \dots, X_k)$, where Q_k is $\exists$ (resp., $\forall$), and $\phi(X_1, \dots, X_k)$ is in 3CNF (resp., 3DNF). For odd k, $\mathrm{QBF}_{k,\exists}^{CNF}$ (resp., $\mathrm{QBF}_{k,\forall}^{DNF}$) is complete for Σ_k^{P} (resp., Π_k^{P}), while, for even k, $\mathrm{QBF}_{k,\exists}^{CNF}$ (resp., $\mathrm{QBF}_{k,\forall}^{DNF}$) is complete for Π_k^{P} (resp., Σ_k^{P}). Observe that $\mathrm{QBF}_{1,\exists}^{CNF}$ (resp., $\mathrm{QBF}_{1,\forall}^{DNF}$) is equivalent to Sat (resp., Taut).

2.3 A previous characterization of Θ_k^{P}

Wagner [35, 37] analyzed extensively the properties of Θ_2^{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 Θ_k^{P}.

A first result intuitively states that, for any language $B \in \Theta_{k+1}^{\mathrm{P}}$, the task of deciding B can be faithfully transformed into the task of deciding, for a suitable language $C \in \Sigma_k^{\mathrm{P}}$ and a suitable sequence $\langle y_1, \dots, y_n\rangle$ of C's instances with $\chi_C(y_1) \geq \dots \chi_C(y_n)$, 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, $B \in \Theta_{k+1}^{\mathrm{P}}$ if and only if there exist a problem $C \in \Sigma_k^{\mathrm{P}}$ and a polynomial-time computable function f such that, for all instances x of B, $f(x) = \langle y_1, \dots, y_{p(\|x\|)}\rangle$ is a sequence of C's instances with $\chi_C(y_1) \geq \dots \geq \chi_C(y_{p(\|x\|)})$, 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 previous lemma, $p(\|x\|)$ is a polynomial, because the function f is polynomial-time computable.

A second result gives a sufficient condition for the $\Theta_{k+1}^{\mathrm{P}}$-hardness of a problem B. Intuitively, this result states that B is $\Theta_{k+1}^{\mathrm{P}}$-hard, if there exists a reduction to B from the problem of deciding, for a given set $\{x_1, \dots, x_n\}$ of instances of a Σ_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 $\chi_A(x_1) \geq \dots \geq \chi_A(x_n)$.

Lemma 2.2 ([35, Theorem 5.2],[37, Theorem 7.1])**.** *Let A be a Σ_k^{P}-complete problem, and let B be a problem. Then, B is $\Theta_{k+1}^{\mathrm{P}}$-hard, if there exists a polynomial-time computable function f such that, for all sets $X = \{x_1, \dots, x_n\}$ of instances of A, $|\{x_i \colon \chi_A(x_i) = 1\}|$ is odd $\Leftrightarrow \chi_B(f(X)) = 1$. The $\Theta_{k+1}^{\mathrm{P}}$-hardness of B remains proven even if sets X are assumed to be such that $\chi_A(x_1) \geq \dots \geq \chi_A(x_n)$.*

3 A new characterization of Θ_k^{P} and its hard problems

In this section, we provide a new characterization of Θ_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 Θ_k^{P}.

In particular, on the one hand, we show that, for any language $B \in \Theta_{k+1}^{\mathrm{P}}$, the task of deciding B can be faithfully transformed into the task of deciding, for two suitable languages $C_1, C_2 \in \Sigma_k^{\mathrm{P}}$ and two suitable sequences $\langle y_1, \dots, y_n\rangle$ and $\langle z_1, \dots, z_m\rangle$ of C_1's and C_2's instances, respectively, with $\chi_{C_1}(y_1) \geq \dots \geq \chi_{C_1}(y_n)$ and $\chi_{C_2}(z_1) \geq \dots \geq \chi_{C_2}(z_n)$, 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.

On the other hand, we show also that, a problem B is $\Theta_{k+1}^{\mathrm{P}}$-hard, if there exists a reduction to B from the problem of deciding, for a given pair of sets $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$ of instances of two (not necessarily distinct) problems A_1 and A_2, respectively, being both of them either Σ_k^{P}-complete or Π_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 $\chi_{A_1}(x_1) \geq \dots \geq \chi_{A_1}(x_n)$ and $\chi_{A_2}(y_1) \geq \dots \geq \chi_{A_2}(y_m)$.

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

[6]Note the difference in the subscripts of the notations $\mathrm{QBF}_{Q_1,k}$ and $\mathrm{QBF}_{k,Q_k}^{CNF}$ (resp., $\mathrm{QBF}_{k,Q_k}^{DNF}$). In the former notation, Q_1 is the *first* quantifier of the sequence, and, for notational convenience, we place "Q_1" before "k" in the subscript. On the other hand, in the latter notation, Q_k is the *last* quantifier of the sequence, and, for notational convenience, we place "Q_k" after "k" in the subscript.

than the number of "yes"-instances of A_2 in Y is $\Theta^{\mathrm{P}}_{k+1}$-hard. Clearly, this problem is also in $\Theta^{\mathrm{P}}_{k+1}$, and hence $\Theta^{\mathrm{P}}_{k+1}$-complete. Therefore, problems in $\Theta^{\mathrm{P}}_{k+1}$ 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 Σ^{P}_k and the other in Π^{P}_k, then comparing the number of "yes"-instances of the two sets is actually a problem belonging to subclasses of $\Theta^{\mathrm{P}}_{k+1}$. In particular, if A_1 is in Σ^{P}_k, and A_2 is in Π^{P}_k, then the comparison can be done in Σ^{P}_k. Symmetrically, if A_1 is in Π^{P}_k, and A_2 is in Σ^{P}_k, then the comparison can be done in Π^{P}_k.

After these general results, we define the problem COMP-VALID$_k$ that is based on the idea of counting and comparison, and we prove it $\Theta^{\mathrm{P}}_{k+1}$-complete. COMP-VALID$_k$ is defined as follows. Given a pair $\langle A, B\rangle$ 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 COMP-VALID$_k$ is defined and shown to be $\Theta^{\mathrm{P}}_{k+1}$-complete. Furthermore, we also prove that the hardness of COMP-VALID$_k$ holds even if $|A| = |B|$, all formulas in $\langle A, B\rangle$ are instances of $\mathrm{QBF}^{CNF}_{k,\exists}$ (resp., $\mathrm{QBF}^{DNF}_{k,\forall}$), have the same number of clauses (resp., terms), and, for each $1 \le d \le k$, quantifiers Q_d of all formulas in $\langle A, B\rangle$ are defined on the very same set of variables. As it will emerge from the proof, the $\Theta^{\mathrm{P}}_{k+1}$-hardness of 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 COMP-VALID$_k$, according to the various types of formulas contained in A and B.

	k is odd	Formulas in B: $\mathrm{QBF}^{CNF}_{k,\exists}$	Formulas in B: $\mathrm{QBF}^{DNF}_{k,\forall}$
Formulas in A	$\mathrm{QBF}^{CNF}_{k,\exists}$	$\Theta^{\mathrm{P}}_{k+1}$	Σ^{P}_k
	$\mathrm{QBF}^{DNF}_{k,\forall}$	Π^{P}_k	$\Theta^{\mathrm{P}}_{k+1}$

	k is even	Formulas in B: $\mathrm{QBF}^{CNF}_{k,\forall}$	Formulas in B: $\mathrm{QBF}^{DNF}_{k,\exists}$
Formulas in A	$\mathrm{QBF}^{CNF}_{k,\forall}$	$\Theta^{\mathrm{P}}_{k+1}$	Π^{P}_k
	$\mathrm{QBF}^{DNF}_{k,\exists}$	Σ^{P}_k	$\Theta^{\mathrm{P}}_{k+1}$

Figure 1: Summary of the complexity results for COMP-VALID$_k$ when specific types of formulas are in sets A and B. COMP-VALID$_k$ is actually complete for the respective complexity classes shown in the tables.

The problem COMP-VALID$_1$, when all formulas are furthermore restricted to be instances of $\mathrm{QBF}^{CNF}_{1,\exists}$ (i.e., SAT), is equivalent to the problem COMP-SAT introduced in [25], which is: Given a pair $\langle A, B\rangle$ 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, COMP-SAT is Θ^{P}_2-complete, and it is Θ^{P}_2-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 Θ^{P}_k-hard if, for its solution, it is required to count the number of "yes"-instances of two sets A and B of instances of Σ^{P}_k-complete, or Π^{P}_k-complete, languages, and compare the computed numbers.

The following result is the analogue of Lemma 2.1 for the new characterization of Θ^{P}_k. Intuitively, it states that, for any language $B \in \Theta^{\mathrm{P}}_{k+1}$, the task of deciding B can be faithfully transformed into the task of deciding, for two suitable languages $C_1, C_2 \in \Sigma^{\mathrm{P}}_k$ and two suitable sequences $\langle y_1, \ldots, y_n\rangle$ and $\langle z_1, \ldots, z_m\rangle$ of C_1's and C_2's instances, respectively, with $\chi_{C_1}(y_1) \ge \cdots \ge \chi_{C_1}(y_n)$ and $\chi_{C_2}(z_1) \ge \cdots \ge \chi_{C_2}(z_n)$, 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 $B \in \Theta^{\mathrm{P}}_{k+1}$, there is a language $C \in \Sigma^{\mathrm{P}}_k$ and a polynomial-time computable function f such that, for any instance x of B, $f(x)$ is a sequence $\langle w_1, \ldots, w_p\rangle$ of instances of C with $\chi_C(w_1) \ge \cdots \ge \chi_C(w_p)$, and x is a "yes"-instance of B if and only if the maximum index ℓ for which w_ℓ is a "yes"-instance of C is odd. Essentially, what is done in the proof is "splitting" the sequence $\langle w_1, \ldots, w_p\rangle$ into two new sequences $\langle y_1, \ldots, y_n\rangle$ and $\langle z_1, \ldots, z_m\rangle$ of instances of C_1 and C_2, respectively. The first of the new sequences contains strings w_ℓ with ℓ odd, while the second one contains strings w_ℓ with ℓ even. Now, if the maximum index ℓ for which w_ℓ 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, $B \in \Theta^{\mathrm{P}}_{k+1}$ if and only if there exist two (not necessarily distinct) problems $C_1, C_2 \in \Sigma^{\mathrm{P}}_k$ and two polynomial-time computable functions f_1 and f_2 such that, for all instances*

x of B, $f_1(x) = \langle y_1, \ldots, y_{p_1(\|x\|)} \rangle$ is a sequence of C_1's instances with $\chi_{C_1}(y_1) \geq \cdots \geq \chi_{C_1}(y_{p_1(\|x\|)})$, $f_2(x) = \langle z_1, \ldots, z_{p_2(\|x\|)} \rangle$ is a sequence of C_2's instances with $\chi_{C_2}(z_1) \geq \cdots \geq \chi_{C_2}(z_{p_2(\|x\|)})$, and $\chi_B(x) = 1 \Leftrightarrow (\max\{i \mid 1 \leq i \leq p_1(\|x\|),\, \chi_{C_1}(y_i) = 1\} > \max\{j \mid 1 \leq j \leq p_2(\|x\|),\, \chi_{C_2}(z_j) = 1\})$.

Proof.

($\Rightarrow$) Let us assume that $B \in \Theta^{\mathrm{P}}_{k+1}$. By Lemma 2.1, there exist a problem $C \in \Sigma^{\mathrm{P}}_k$ and a polynomial-time computable function f such that, for all instances x of B, $f(x) = \langle w_1, \ldots, w_{p(\|x\|)} \rangle$ is a sequence of C's instances with $\chi_C(w_1) \geq \cdots \geq \chi_C(w_{p(\|x\|)})$, and $\chi_B(x) = 1 \Leftrightarrow \max\{\ell \mid 1 \leq i \leq p(\|x\|),\, \chi_C(w_\ell) = 1\}$ is odd.

Now, consider languages C_1 and C_2 such that $C_1 = C_2 = C$ (i.e., for any string x, $\chi_{C_1}(x) = \chi_{C_2}(x) = \chi_C(x)$). Clearly, since $C \in \Sigma^{\mathrm{P}}_k$, C_1 and C_2 belong to Σ^{P}_k as well. Functions f_1 and f_2 are defined from f as follows. Assume that for an instance x of B, $f(x) = \langle w_1, \ldots, w_{p(\|x\|)} \rangle$. On the one hand, function f_1 is such that $f_1(x) = \langle y_1, \ldots, y_{p_1(\|x\|)} \rangle$, where $p_1(\|x\|) = \left\lceil \frac{p(\|x\|)}{2} \right\rceil$ and $y_i = w_{2i-1}$ for all i. On the other hand, function f_2 is such that $f_2(x) = \langle z_1, \ldots, z_{p_2(\|x\|)} \rangle$, where $p_2(\|x\|) = \left\lfloor \frac{p(\|x\|)}{2} \right\rfloor$ 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 $\chi_C(w_1) \geq \cdots \geq \chi_C(w_{p(\|x\|)})$, $\chi_{C_1}(y_1) \geq \cdots \geq \chi_{C_1}(y_{p_1(\|x\|)})$ and $\chi_{C_2}(z_1) \geq \cdots \geq \chi_{C_2}(z_{p_2(\|x\|)})$, as well. Now, observe that

$$\begin{aligned}\chi_B(x) = 1 &\Leftrightarrow \max\{\ell \mid 1 \leq \ell \leq p(\|x\|),\, \chi_C(w_\ell) = 1\} \text{ is odd}\\ &\Leftrightarrow (\max\{i \mid 1 \leq i \leq p_1(\|x\|),\, \chi_{C_1}(y_i) = 1\} > \max\{j \mid 1 \leq j \leq p_2(\|x\|),\, \chi_{C_2}(z_j) = 1\}).\end{aligned}$$

($\Leftarrow$) Let us now assume that there exist two problems $C_1, C_2 \in \Sigma^{\mathrm{P}}_k$ and two polynomial-time computable functions f_1 and f_2 such that, for any instance x of B, $f_1(x) = \langle y_1, \ldots, y_{p_1(\|x\|)} \rangle$ is a sequence of C_1's instances with $\chi_{C_1}(y_1) \geq \cdots \geq \chi_{C_1}(y_{p_1(\|x\|)})$, $f_2(x) = \langle z_1, \ldots, z_{p_2(\|x\|)} \rangle$ is a sequence of C_2's instances with $\chi_{C_2}(z_1) \geq \cdots \geq \chi_{C_2}(z_{p_2(\|x\|)})$, and $\chi_B(x) = 1 \Leftrightarrow (\max\{i \mid 1 \leq i \leq p_1(\|x\|),\, \chi_{C_1}(y_i) = 1\} > \max\{j \mid 1 \leq j \leq p_2(\|x\|),\, \chi_{C_2}(z_j) = 1\})$. We are going to show that $B \in \Theta^{\mathrm{P}}_{k+1}$ by exhibiting a Turing machine in P that can decide B by querying logarithmic many times an oracle in Σ^{P}_k.

Consider a deterministic polynomial-time Turing machine M that can query an oracle in Σ^{P}_k for C_1 and C_2. Observe that it is sufficient to devise a single oracle receiving in input also a variable which tells the oracle to decide either C_1 or C_2. Given the existence of those specific problems, C_1 and C_2, and functions, f_1 and f_2, in order for M to decide whether an input string x is a "yes"-instance of B or not, it is sufficient to compute the max values max_i and max_j of i and j such that $\chi_{C_1}(y_i) = 1$ and $\chi_{C_2}(z_j) = 1$, respectively, and compare them. M can compute max_i as follows. First, M computes $f_1(x) = \langle y_1, \ldots, y_{p_1(\|x\|)} \rangle$. M can do so because f_1 is polynomial-time computable. Next, since $\chi_{C_1}(y_1) \geq \cdots \geq \chi_{C_1}(y_{p_1(\|x\|)})$ (i.e., all the "yes"-instances of C_1 are at the beginning of the sequence), max_i can be computed via a binary search. In fact, by asking to the oracle whether the various y_i are "yes"-instances of C_1, M can perform such a binary search in the range $[1, p_1(\|x\|)]$ and compute max_i. Observe that M needs to issue only a logarithmic number of calls to its oracle. Similarly, M can compute max_j. Clearly, the overall procedure is feasible in $\mathrm{P}^{\Sigma^{\mathrm{P}}_k[O(\log n)]} = \Theta^{\mathrm{P}}_{k+1}$. □

Note that, also in this case, in the statement of the previous lemma, $p_1(\|x\|)$ and $p_2(\|x\|)$ are polynomials, because functions f_1 and f_2 are polynomial-time computable.

From the theorem above, the next theorem follows, stating that, for two given sets $X = \{x_1, \ldots, x_n\}$ and $Y = \{y_1, \ldots, y_m\}$ of instances of two (not necessarily distinct) Σ^{P}_k-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 $\Theta^{\mathrm{P}}_{k+1}$-hard. Interestingly, 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 main idea behind the proof is the following. We know that, to prove the $\Theta^{\mathrm{P}}_{k+1}$-hardness of a problem, we have to show that there exists a polynomial reduction from any problem $D \in \Theta^{\mathrm{P}}_{k+1}$. If D is a problem belonging to $\Theta^{\mathrm{P}}_{k+1}$, by Theorem 3.1, there exist two problems $E_1, E_2 \in \Sigma^{\mathrm{P}}_k$ and two polynomial-time computable functions f_1, f_2 such that, for all instances v of D, $f_1(v) = \langle w_1, \ldots, w_{p_1} \rangle$ is a sequence of E_1's instances with $\chi_{E_1}(w_1) \geq \cdots \geq \chi_{E_1}(w_{p_1})$, $f_2(v) = \langle z_1, \ldots, z_{p_2} \rangle$ is a sequence of E_2's instances with $\chi_{E_2}(z_1) \geq \cdots \geq \chi_{E_2}(z_{p_2})$, and v is a "yes"-instance of B if and only if max_i, that is the maximum index i for which w_i is a "yes"-instance of E_1, is bigger than max_j, that is the maximum index j for which z_j is a "yes"-instance of E_2. However, since $\chi_{E_1}(w_1) \geq \cdots \geq \chi_{E_1}(w_{p_1})$ and $\chi_{E_2}(z_1) \geq \cdots \geq \chi_{E_2}(z_{p_2})$, max_i and max_j are actually the number of "yes"-instances of E_1 and E_2 in $\langle w_1, \ldots, w_{p_1} \rangle$ and $\langle z_1, \ldots, z_{p_2} \rangle$, respectively. Furthermore, by A_1 and A_2 being

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 $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$ of instances of A_1 and A_2, respectively, deciding whether $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$ is $\Theta_{k+1}^{\mathrm{P}}$-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 $\Theta_{k+1}^{\mathrm{P}}$ is polynomially reducible to CC. Let D be any problem in $\Theta_{k+1}^{\mathrm{P}}$. By Theorem 3.1, there exist two problems $E_1, E_2 \in \Sigma_k^{\mathrm{P}}$ and two polynomial-time computable function f_1 and f_2 such that, for all instances v of D, $f_1(v) = \langle w_1, \dots, w_{p_1(\|v\|)} \rangle$ is a sequence of E_1's instances with $\chi_{E_1}(w_1) \geq \cdots \geq \chi_{E_1}(w_{p_1(\|v\|)})$, $f_2(v) = \langle z_1, \dots, z_{p_2(\|v\|)} \rangle$ is a sequence of E_2's instances with $\chi_{E_2}(z_1) \geq \cdots \geq \chi_{E_2}(z_{p_2(\|v\|)})$, and $\chi_D(v) = 1 \Leftrightarrow (\max\{i \mid 1 \leq i \leq p_1(\|v\|),\ \chi_{E_1}(w_i) = 1\} > \max\{j \mid 1 \leq j \leq p_2(\|v\|),\ \chi_{E_2}(z_j) = 1\})$.

Furthermore, since $E_1, E_2 \in \Sigma_k^{\mathrm{P}}$, and A_1 and A_2 are Σ_k^{P}-complete, there exist polynomial reductions h_1 and h_2 such that $\chi_{E_1}(w) = 1 \Leftrightarrow \chi_{A_1}(h_1(w)) = 1$ for all strings w, and $\chi_{E_2}(z) = 1 \Leftrightarrow \chi_{A_2}(h_2(z)) = 1$ for all strings z.

Consider a string v instance of D. From v we derive the sets $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$ (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 $\chi_{E_1}(w_1) \geq \cdots \geq \chi_{E_1}(w_{p_1(\|v\|)})$, it follows that $\chi_{A_1}(x_1) \geq \cdots \geq \chi_{A_1}(x_{p_1(\|v\|)})$, and from $\chi_{E_2}(z_1) \geq \cdots \geq \chi_{E_2}(z_{p_2(\|v\|)})$, 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

$$\begin{aligned}
\chi_D(v) = 1 &\Leftrightarrow (\max\{i \mid 1 \leq i \leq p_1(\|v\|),\ \chi_{E_1}(w_i) = 1\} > \max\{j \mid 1 \leq j \leq p_2(\|v\|),\ \chi_{E_2}(z_j) = 1\}) \\
&\Leftrightarrow (\max\{i \mid 1 \leq i \leq p_1(\|v\|),\ \chi_{A_1}(h_1(w_i)) = 1\} > \max\{j \mid 1 \leq j \leq p_2(\|v\|),\ \chi_{A_2}(h_2(z_j)) = 1\}) \\
&\Leftrightarrow (\max\{i \mid 1 \leq i \leq p_1(\|v\|),\ \chi_{A_1}(x_i) = 1\} > \max\{j \mid 1 \leq j \leq p_2(\|v\|),\ \chi_{A_2}(y_j) = 1\}) \\
&\Leftrightarrow (|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|).
\end{aligned}$$

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, $\chi_D(v) = 1 \Leftrightarrow (|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|)$. ☐

Similarly, for two given sets $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$ 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 $\Theta_{k+1}^{\mathrm{P}}$-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 $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$ of instances of A_1 and A_2, respectively, deciding whether $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$ is $\Theta_{k+1}^{\mathrm{P}}$-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 $\overline{A_1}$ and $\overline{A_2}$ denote the complement problems to A_1 and A_2, respectively. Clearly, $\overline{A_1}$ and $\overline{A_2}$ are Σ_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 $X' = \{x_1', \dots, x_n'\}$ and $Y' = \{y_1', \dots, y_n'\}$ be two sets of instances of A_1' and A_2', respectively, such that $\chi_{A_1'}(x_1') \geq \cdots \geq \chi_{A_1'}(x_n')$ and $\chi_{A_2'}(y_1') \geq \cdots \geq \chi_{A_2'}(y_n')$. From Theorem 3.2, we know that deciding whether $|\{x_i \colon \chi_{A_1'}(x_i') = 1\}| > |\{y_j \colon \chi_{A_2'}(y_j') = 1\}|$ is $\Theta_{k+1}^{\mathrm{P}}$-hard.

Starting from X' and Y', we define sets $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_n\}$, where $x_i = y_{n-i+1}'$, and $y_j = x_{n-j+1}'$. 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

$$\begin{aligned}
(|\{x_i' \colon \chi_{A_1'}(x_i') = 1\}| > |\{y_j' \colon \chi_{A_2'}(y_j') = 1\}|) &\Leftrightarrow (|\{x_i' \colon \chi_{A_1'}(x_i') = 0\}| < |\{y_j' \colon \chi_{A_2'}(y_j') = 0\}|) (\text{because } n = m) \\
&\Leftrightarrow (|\{x_i' \colon \chi_{\overline{A_2}}(x_i') = 0\}| < |\{y_j' \colon \chi_{\overline{A_1}}(y_j') = 0\}|) \\
&\Leftrightarrow (|\{y_j \colon \chi_{\overline{A_2}}(y_j) = 0\}| < |\{x_i \colon \chi_{\overline{A_1}}(x_i) = 0\}|) \\
&\Leftrightarrow (|\{x_i \colon \chi_{\overline{A_1}}(x_i) = 0\}| > |\{y_j \colon \chi_{\overline{A_2}}(y_j) = 0\}|) \\
&\Leftrightarrow (|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|).
\end{aligned}$$
☐

It is interesting to see that, for two given sets $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$ of instances of problems A_1 and A_2, respectively, if A_1 is in Σ_k^{P}, and A_2 is in Π_k^{P}, then deciding whether $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$ is actually computationally easier than $\Theta_{k+1}^{\mathrm{P}}$, in particular, it is in Σ_k^{P}.

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 $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$. 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 $A_1 \in \Sigma_k^{\mathrm{P}}$ and $A_2 \in \Pi_k^{\mathrm{P}}$, 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 Σ_k^{P}-complete problem and A_2 be a Π_k^{P}-complete problem. Then, for a given pair of sets $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_m\}$ of instances of A_1 and A_2, respectively, deciding whether $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$ is Σ_k^{P}-complete.*

Proof. Let $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$. We first show that $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$ 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 $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$. Observe that $|Y| = |\{y_j \colon \chi_{A_2}(y_j) = 1\}| + |\{y_j \colon \chi_{A_2}(y_j) = 0\}| < |\{x_i \colon \chi_{A_1}(x_i) = 1\}| + |\{y_j \colon \chi_{A_2}(y_j) = 0\}|$. By choosing $p = |\{x_i \colon \chi_{A_1}(x_i) = 1\}|$ and $q = |\{y_j \colon \chi_{A_2}(y_j) = 0\}|$, it follows that $|Y| < p + q$. 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 $p \le |\{x_i \colon \chi_{A_1}(x_i) = 1\}|$ and $q \le |\{y_j \colon \chi_{A_2}(y_j) = 0\}| = |Y| - |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$. By summing up these two inequalities, $p + q \le |\{x_i \colon \chi_{A_1}(x_i) = 1\}| - |\{y_j \colon \chi_{A_2}(y_j) = 1\}| + |Y|$, which, along with $|Y| < p + q$, implies that $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| - |\{y_j \colon \chi_{A_2}(y_j) = 1\}| > 0$. Therefore, $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$.

Thus, as for membership in Σ_k^{P}, to decide whether $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$, it suffices to (1) guess a set of p "yes"-instances of A_1 from X along with their witnesses, and a set of q "no"-instances of A_2 from Y 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 (specifically, $p + q$) calls to a suitable Π_{k-1}^{P} oracle by passing the guessed witnesses); and (3) verify that $p + q > |Y|$ (feasible in P). Hence, this is overall feasible in Σ_k^{P}.

Hardness for Σ_k^{P} 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 Σ_k^{P}-hard, because A_1 is Σ_k^{P}-complete). ☐

On the other hand, for two given sets $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$ of instances of problems A_1 and A_2, respectively, if A_1 is in Π_k^{P} and A_2 is in Σ_k^{P}, then deciding whether $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$ is in Π_k^{P}. 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 $p' + q' \ge |X|$, then $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| \le |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$ (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 $A_1 \in \Pi_k^{\mathrm{P}}$ and $A_2 \in \Sigma_k^{\mathrm{P}}$, 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 Π_k^{P}-complete problem, and A_2 be a Σ_k^{P}-complete problem. Then, for a given pair of sets $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_m\}$ of instances of A_1 and A_2, respectively, deciding whether $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$ is Π_k^{P}-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 $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| \le |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$ (i.e., $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_m\}$ are a "no"-instance) if and only if there are p' "no"-instances of A_1 in $\{x_1, \dots, x_n\}$ and q' "yes"-instances of A_2 in $\{y_1, \dots, y_m\}$ such that $p' + q' \ge |X|$. By exploiting this property, it can be shown that deciding whether $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| \le |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$ is feasible in Σ_k^{P}, and hence deciding $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$ is feasible in Π_k^{P}. ☐

To conclude this part, we show the analogue of Lemma 2.2 for the new characterization of Θ_k^{P}, which directly descends from Theorems 3.2 and 3.3, and provides a new sufficient condition for the $\Theta_{k+1}^{\mathrm{P}}$-hardness of a problem. Intuitively, it states that, a problem B is $\Theta_{k+1}^{\mathrm{P}}$-hard, if there exists a reduction to B from the problem of deciding, for a given pair of sets $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$ of instances of two (not necessarily distinct) Σ_k^{P}-complete (or, Π_k^{P}-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 $\chi_{A_1}(x_1) \ge \dots \ge \chi_{A_1}(x_n)$ and $\chi_{A_2}(y_1) \ge \dots \ge \chi_{A_2}(y_m)$ (and $n = m$).

Theorem 3.6. *Let A_1 and A_2 be two Σ_k^{P}-complete (resp., Π_k^{P}-complete) problems, and let B be a problem. Then, B is $\Theta_{k+1}^{\mathrm{P}}$-hard, if there exists a polynomial-time computable function f such that, for all pair of sets $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$ of instances of A_1 and A_2, respectively, it holds that $(|\{x_i\colon \chi_{A_1}(x_i) = 1\}| > |\{y_j\colon \chi_{A_2}(y_j) = 1\}|) \Leftrightarrow \chi_B(f(X,Y)) = 1$. The $\Theta_{k+1}^{\mathrm{P}}$-hardness of B remains proven even if sets X and Y are assumed to be such that $\chi_{A_1}(x_1) \geq \dots \geq \chi_{A_1}(x_n)$ and $\chi_{A_2}(y_1) \geq \dots \geq \chi_{A_2}(y_m)$, and $n = m$.*

Proof. From Theorems 3.2 and 3.3, deciding whether $|\{x_i\colon \chi_{A_1}(x_i) = 1\}| > |\{y_j\colon \chi_{A_2}(y_j) = 1\}|$ is $\Theta_{k+1}^{\mathrm{P}}$-hard, both in the case that A_1 and A_2 are Σ_k^{P}-complete and in the case that A_1 and A_2 are Π_k^{P}-complete. Hardness holds even if $\chi_{A_1}(x_1) \geq \dots \geq \chi_{A_1}(x_n)$ and $\chi_{A_2}(y_1) \geq \dots \geq \chi_{A_2}(y_m)$, and $n = m$. Therefore, if there is a reduction (f) from the task of deciding $|\{x_i\colon \chi_{A_1}(x_i) = 1\}| > |\{y_j\colon \chi_{A_2}(y_j) = 1\}|$ to B, then B is $\Theta_{k+1}^{\mathrm{P}}$-hard as well. ☐

3.3 Complexity of Comp-Valid$_k$

We now prove Comp-Valid$_k$ $\Theta_{k+1}^{\mathrm{P}}$-complete. In the following, for a set S of QBFs, $\#val(S)$ denotes the number of valid formulas in S. To show Comp-Valid$_k$'s membership in $\Theta_{k+1}^{\mathrm{P}}$, we use two problems LBound-Valid$_{k,\exists}$ and UBound-Valid$_{k,\forall}$ defined as follows. A pair $\langle S, n\rangle$, 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 LBound-Valid$_{k,\exists}$ whenever $\#val(S) \geq n$. On the other hand, a pair $\langle S, n\rangle$, 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 UBound-Valid$_{k,\forall}$ whenever $\#val(S) \leq n$. We show that both these problems are feasible in Σ_k^{P}. Intuitively, this is the case, because, to decide LBound-Valid$_{k,\exists}$, it is sufficient to guess n valid formulas in S (feasible in NP), and then ask to a Π_{k-1}^{P} oracle that the guessed formulas are actually valid. A similar intuition is also behind the complexity of UBound-Valid$_{k,\forall}$. Indeed, to decide UBound-Valid$_{k,\forall}$, it is sufficient to guess $|S| - n$ non-valid formulas in S (feasible in NP), and then ask a Π_{k-1}^{P} oracle that the guessed formulas are actually non-valid.

Lemma 3.7. LBound-Valid$_{k,\exists}$ *and* UBound-Valid$_{k,\forall}$ *belong to* Σ_k^{P}.

Proof. We first focus on LBound-Valid$_{k,\exists}$. Let $\langle S, n\rangle$ be an instance of LBound-Valid$_{k,\exists}$. Since all the formulas Φ_i in S are of the form $\Phi_i = (\exists X_1^i) \dots (Q_p X_p^i)\phi_i(X_1^i, \dots, X_p^i)$, with $p \leq k$, we can guess the indices $\{j_1, \dots, j_n\}$ of a set of n valid formulas, along with the complete assignments $\sigma_{X_1^\ell}$ over X_1^ℓ, for each $\ell \in \{j_1, \dots, j_n\}$, 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 $\ell \in \{j_1, \dots, j_n\}$, we can check the validity of Φ_ℓ by checking the validity of $\Phi_\ell{}' = (\forall X_2^\ell)(\exists X_3^\ell) \dots (Q_p X_p^\ell)\phi_\ell(X_1^\ell/\sigma_{X_1^\ell}, X_2^\ell, \dots, X_p^\ell)$ through a call to a Π_{k-1}^{P} oracle (because $p - 1 \leq k - 1$). Clearly, the overall procedure is feasible in Σ_k^{P}.

For UBound-Valid$_{k,\forall}$ the proof is similar. Let $\langle S, n\rangle$ be an instance of UBound-Valid$_{k,\forall}$. Since all the formulas Φ_i in S are in the form $\Phi_i = (\forall X_1^i) \dots (Q_p X_p^i)\phi_i(X_1^i, \dots, X_p^i)$, we can guess the indices $\{j_1, \dots, j_m\}$ of a set of $m = |S| - n$ *non*-valid formulas, along with the complete assignments $\sigma_{X_1^\ell}$ over X_1^ℓ, for each $\ell \in \{j_1, \dots, j_m\}$, 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 $\ell \in \{j_1, \dots, j_m\}$, we can check that Φ_ℓ is non-valid by checking that $\neg\Phi_\ell{}' = (\forall X_2^\ell)(\exists X_3^\ell) \dots (Q_p X_p^\ell)\neg\phi_\ell(X_1^\ell/\sigma_{X_1^\ell}, X_2^\ell, \dots, X_p^\ell)$ is valid through a call to a Π_{k-1}^{P} oracle (because $p - 1 \leq k - 1$). Also in this case, the overall procedure is feasible in Σ_k^{P}. ☐

We are now ready to show that Comp-Valid$_k$ is $\Theta_{k+1}^{\mathrm{P}}$-complete. We first prove its membership in $\Theta_{k+1}^{\mathrm{P}}$. Intuitively, Comp-Valid$_k$ is in $\Theta_{k+1}^{\mathrm{P}}$ 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 LBound-Valid$_{k,\exists}$ and 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 $\#val(A) > \#val(B)$ is feasible in $\Theta_{k+1}^{\mathrm{P}}$.*

Proof. Let $\langle A, B\rangle$ be an instance of Comp-Valid$_k$. To decide whether $\#val(A) > \#val(B)$, 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 $A_\exists$ and $A_\forall$, containing the formulas whose outermost quantifier is $\exists$ and $\forall$, respectively. Clearly, $|A| = |A_\exists| + |A_\forall|$. We can count the number of valid formulas belonging to $A_\exists$ and $A_\forall$, via a binary search in the range $[0, |A|]$, by a Turing machine in P querying an oracle for LBound-Valid$_{k,\exists}$ and 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 LBound-Valid$_{k,\exists}$ or 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 Σ_k^{P} (see Lemma 3.7). Thus, the overall procedure is feasible in $\mathrm{P}^{\Sigma_k^{\mathrm{P}}[O(\log n)]} = \Theta_{k+1}^{\mathrm{P}}$. $\square$

We next prove that Comp-Valid$_k$ is $\Theta_{k+1}^{\mathrm{P}}$-hard. The reduction to show the $\Theta_{k+1}^{\mathrm{P}}$-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 $\mathrm{QBF}_{k,\exists}^{CNF}$ (or $\mathrm{QBF}_{k,\forall}^{DNF}$). Furthermore, we also prove that the hardness of Comp-Valid$_k$ holds even if $|A| = |B|$, all formulas in $\langle A, B\rangle$ are instances of $\mathrm{QBF}_{k,\exists}^{CNF}$ (resp., $\mathrm{QBF}_{k,\forall}^{DNF}$), have the same number of clauses (resp., terms), and, for each $1 \leq d \leq k$, quantifiers Q_d of all formulas in $\langle A, B\rangle$ 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 $\Theta_{k+1}^{\mathrm{P}}$-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 $\Theta_{k+1}^{\mathrm{P}}$-hard. Hardness holds even if $|A| = |B|$, all formulas in $\langle A, B\rangle$ are instances of $\mathrm{QBF}_{k,\exists}^{CNF}$ (resp., $\mathrm{QBF}_{k,\forall}^{DNF}$), have the same number of clauses (resp., terms), and, for each $1 \leq d \leq k$, quantifiers Q_d of all formulas in $\langle A, B\rangle$ 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 $\langle A, B\rangle$ such that $|A| = |B|$, and all formulas in A and B are instances of $\mathrm{QBF}_{k,\exists}^{CNF}$ (resp., $\mathrm{QBF}_{k,\forall}^{DNF}$). 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.

$\mathrm{QBF}_{k,\exists}^{CNF}$ for odd k (resp., $\mathrm{QBF}_{k,\forall}^{DNF}$ for even k) is Σ_k^{P}-complete, and, given any two sets $X = \{x_1, \dots, x_n\}$ and $Y = \{y_1, \dots, y_m\}$ of instances of $\mathrm{QBF}_{k,\exists}^{CNF}$ for odd k (resp., $\mathrm{QBF}_{k,\forall}^{DNF}$ for even k), clearly the pair $\langle X, Y\rangle$ itself is a "yes"-instance of Comp-Valid$_k$ if and only if $|\{x_i \colon \chi_{A_1}(x_i) = 1\}| > |\{y_j \colon \chi_{A_2}(y_j) = 1\}|$. Hence, Theorem 3.2 applies (with $n = m$), and thus Comp-Valid$_k$ is $\Theta_{k+1}^{\mathrm{P}}$-hard. Furthermore, $\mathrm{QBF}_{k,\exists}^{CNF}$ for even k (resp., $\mathrm{QBF}_{k,\forall}^{DNF}$ for odd k) is Π_k^{P}-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 on 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 $\mathrm{QBF}_{k,\exists}^{CNF}$ (resp., $\mathrm{QBF}_{k,\forall}^{DNF}$), and hence all of them have the very same number of nested quantifiers. In the following, we assume that all QBFs are instances of $\mathrm{QBF}_{k,\exists}^{CNF}$, i.e., all QBFs are of the form $\Phi_\ell = (Q_1 X_1^\ell) \dots (Q_k X_k^\ell)\phi_\ell(X_1^\ell, \dots, X_k^\ell)$ with $Q_k = \exists$ and ϕ_ℓ being in CNF. The case that all QBFs are instances of $\mathrm{QBF}_{k,\forall}^{DNF}$ 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^ℓ among the various formulas Φ_ℓ. If there are two formulas $\Phi_i, \Phi_j \in A \cup B$ such that $|X_d^i| \neq |X_d^j|$, then we compute $max_d = \max_{\Phi_\ell \in A\cup B}\{|X_d^\ell|\}$. Subsequently, we rewrite all formulas Φ_ℓ with $|X_d^\ell| < max_d$ by: (*i*) extending X_d^ℓ to $\tilde{X}_d^\ell$ by adding fresh variables, so that $|\tilde{X}_d^\ell| = max_d$; (*ii*) adding to ϕ_ℓ, for each variable $x \in \tilde{X}_d^\ell \setminus X_d^\ell$, dummy satisfiable clauses of three literals $(x \vee \neg x \vee x)$, so that the new variables appear in the non-quantified part. Since $(\exists x)(x \vee \neg x \vee x)$ and $(\forall x)(x \vee \neg x \vee x)$ are both always valid, adding clauses like $(x \vee \neg x \vee x)$ 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 $\tilde{\phi}_\ell$ the new non-quantified part of the formulas obtained after the rewriting, while $\tilde{\Phi}_\ell$ 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 $\tilde{\Phi}_i, \tilde{\Phi}_j \in A \cup B$ such that the number of clauses in $\tilde{\phi}_i$ is different from the number of clauses in $\tilde{\phi}_j$, then we compute $max_{cl} = \max_{\tilde{\Phi}_\ell \in A\cup B}\{\text{number of clauses of } \tilde{\phi}_\ell\}$. After this, we rewrite all formulas $\tilde{\Phi}_\ell$ whose formula $\tilde{\phi}_\ell$ has less clauses than max_{cl} by adding dummy satisfiable clauses of three literals $(x \vee \neg x \vee x)$, where

x is any variable of the formula. Again, note that adding clauses like $(x \vee \neg x \vee x)$ 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.

The just described rewriting of formulas in A and B is feasible in polynomial time. $\square$

It is interesting to note that, given the particular statements of Theorems 3.2 and 3.3, COMP-VALID$_k$ remains $\Theta_{k+1}^{\mathrm{P}}$-hard even in the case in which the sets $X = \{\Phi_1, \ldots, \Phi_n\}$ and $Y = \{\Psi_1, \ldots, \Psi_m\}$ of the quantified formulas are assumed to be such that $\Phi_1 \Leftarrow \cdots \Leftarrow \Phi_n$ and $\Psi_1 \Leftarrow \cdots \Leftarrow \Psi_m$, i.e., all the valid formulas are at the beginning of the lists (have the smallest indices in the sets).

4 Applications of the new characterization

In this section, we show that the new characterization for Θ_k^{P} (and, more specifically, for Θ_2^{P}) introduced in this paper provides a powerful tool to prove Θ_k^{P}-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 mCP-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, this model captures preferences like "given that the rest of the dinner does not change, with a fish dish (A's value), I prefer a white wine (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 β is preferred to the outcome α according to the preferences modeled in N, and $\beta \bowtie_N \alpha$ denotes that β and α 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 mCP-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 mCP-nets. Various voting schemes were proposed for mCP-nets [28, 24], and here we focus on the Max voting scheme [28].

More precisely, an mCP-net $\mathcal{M}$ is a collection $\langle N_1, \ldots, N_m \rangle$ of m CP-nets defined over the same set of features which, in turn, have the same possible values. The "m" of an mCP-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 mCP-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 $\mathcal{M} = \langle N_1, \ldots, N_m \rangle$ be an mCP-net, and let α, β be two outcomes. We define the sets $S_{\mathcal{M}}^{\succ}(\alpha, \beta) = \{i \mid \alpha \succ_{N_i} \beta\}$, $S_{\mathcal{M}}^{\prec}(\alpha, \beta) = \{i \mid \alpha \prec_{N_i} \beta\}$, and $S_{\mathcal{M}}^{\bowtie}(\alpha, \beta) = \{i \mid \alpha \bowtie_{N_i} \beta\}$, which are the sets of the agents of $\mathcal{M}$ preferring α to β, preferring β to α, and for which α and β are incomparable, respectively. The Max voting is defined as follows: the outcome β *max dominates* the outcome α, denoted by $\beta \succ_{\mathcal{M}}^{max} \alpha$, if the group of the agents of $\mathcal{M}$ preferring β to α is the *biggest*, i.e., $|S_{\mathcal{M}}^{\succ}(\beta, \alpha)| > \max(|S_{\mathcal{M}}^{\prec}(\beta, \alpha)|, |S_{\mathcal{M}}^{\bowtie}(\beta, \alpha)|)$.

For any given outcome γ, it is possible to design a CP-net $D(\gamma)$ for which γ is the optimum outcome [25]. For any 3CNF Boolean formula ϕ, it is possible to design two different CP-nets, $F(\phi)$ and $\overline{F}(\phi)$, such that, for two specific outcomes α and β, $\beta \succ_{F(\phi)} \alpha$ (resp., $\alpha \succ_{\overline{F}(\phi)} \beta$) if and only if ϕ is satisfiable, and $\beta \bowtie_{F(\phi)} \alpha$ (resp., $\alpha \bowtie_{\overline{F}(\phi)} \beta$) if and only if ϕ is unsatisfiable [25]. Below, we will refer again to the outcomes α and β just mentioned. With these nets, it is possible to show that deciding Max dominance is Θ_2^{P}-hard.

Consider a generic instance $\langle A, B \rangle$ 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 $\langle A, B \rangle$, it is possible to build a $3(a+b)$CP-net $\mathcal{M}_{\max}(\langle A, B \rangle)$ such that $\beta \succ_{\mathcal{M}_{\max}(\langle A, B \rangle)}^{max} \alpha$ if and only if $\langle A, B \rangle$ is a "yes"-instance of COMP-SAT. In particular, the agents of $\mathcal{M}_{\max}(\langle A, B \rangle)$ are:

- for each formula $\phi_i \in A$, there is an agent whose CP-net is $N_{A,i} = F(\phi_i)$;
- for each formula $\varphi_j \in B$, there is an agent whose CP-net is $N_{B,j} = \overline{F}(\varphi_j)$;
- 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 mCP-net, and let α and β be two outcomes. Deciding whether $\beta \succ_{\mathcal{M}}^{max} \alpha$ is Θ_2^{P}-hard.*

5 Conclusion

In this paper, we have introduced a new characterization for the class Θ_k^{P} in general and for Θ_2^{P} in particular. We have shown that problems belonging to $\Theta_{k+1}^{\mathrm{P}}$ are also those involving the task of counting and comparing the number of "yes"-instances of two sets A and B of Σ_k^{P}-complete (or Π_k^{P}-complete) problem instances. Moreover, we have also shown that this new characterization is sufficient to entail the $\Theta_{k+1}^{\mathrm{P}}$-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 Σ_k^{P}-complete (or Π_k^{P}-complete) problems instances, then the problem is $\Theta_{k+1}^{\mathrm{P}}$-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 Σ_k^{P}-complete and of a Π_k^{P}-complete problem, respectively.

We have complemented this work by providing also the $\Theta_{k+1}^{\mathrm{P}}$-complete problem 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 the valid formulas in B. The results shown here prove that its specialization COMP-VALID$_1$, when only existentially quantified CNF Boolean formulas are considered, (= COMP-SAT), which 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 Θ_2^{P}-complete. COMP-VALID$_k$ (resp., COMP-SAT) proves to be an ideal candidate when one needs a reduction to show $\Theta_{k+1}^{\mathrm{P}}$-hardness (resp., Θ_2^{P}-hardness) of a problem involving the task of counting and comparison. In fact, the Θ_2^{P}-hardness of COMP-SAT is an easy corollary of the $\Theta_{k+1}^{\mathrm{P}}$-hardness of COMP-VALID$_k$, and it was successfully used to easily prove the Θ_2^{P}-hardness of a voting problem in [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 $\mathcal{P} =?\mathcal{NP}$ 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 Θ_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 Coalitional 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.

[15] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. “On the Complexity of the Core over Coalition Structures”. In: *Proc. of IJCAI*. 2011, pp. 216–221.

[16] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. “The Complexity of the Nucleolus in Compact Games”. In: *ACM TOCT* 7.1 (2014), 3:1–3:52.

[17] J. Hartmanis. “Structural Complexity Column: Some Observations About Relativization of Space Bounded Computations”. In: *Bull. EATCS* 35 (1988), pp. 82–91.

[18] E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. “Exact analysis of Dodgson elections: Lewis Carroll’s 1876 voting system is complete for parallel access to NP”. In: *J. ACM* 44.6 (1997), pp. 806–825.

[19] E. Hemaspaandraa, H. Spakowski, and J. Vogel. “The complexity of Kemeny elections”. In: *Theor. Comput. Sci.* 349.3 (2005), pp. 382–391.

[20] D. S. Johnson. “A catalog of complexity classes”. In: *Handbook of Theoretical Computer Science (vol. A)*. Ed. by J. van Leeuwen. Amsterdam, The Netherlands: Elsevier Science Publishers B.V., 1990, pp. 67–161.

[21] J. Kadin. “$\mathrm{P}^{\mathrm{NP}[O(\log n)]}$ and sparse Turing-complete sets for NP”. In: *J. Comput. Syst. Sci.* 39.3 (1989), pp. 282–298.

[22] R. M. Karp. “Reducibility among combinatorial problems”. In: *Proc. of Complexity of Computer Computations*. Plenum Press, 1972, pp. 85–103.

[23] M. W. Krentel. “The Complexity of Optimization Problems”. In: *J. Comput. Syst. Sci.* 36.3 (1988), pp. 490–509.

[24] M. Li, Q. B. Vo, and R. Kowalczyk. “Aggregating multi-valued CP-nets: a CSP-based approach”. In: *J. Heuristics* 21.1 (2015), pp. 107–140.

[25] T. Lukasiewicz and E. Malizia. “On the Complexity of mCP-Nets”. In: *Proc. of AAAI*. 2016, pp. 558–564.

[26] C. H. Papadimitriou. *Computational Complexity*. Reading, MA, USA: Addison Wesley, 1994.

[27] C. H. Papadimitriou and S. K. Zachos. “Two remarks on the power of counting”. In: *Proc. of Theoretical Computer Science*. Vol. 145. LNCS. Springer, 1982, pp. 269–275.

[28] F. Rossi, K. B. Venable, and T. Walsh. “mCP Nets: representing and reasoning with preferences of multiple agents”. In: *Proc. of AAAI*. 2004, pp. 729–734.

[29] J. Rothe, H. Spakowski, and J. Vogel. “Exact Complexity of the Winner Problem for Young Elections”. In: *Theor. Comput. Syst.* 36.4 (2003), pp. 375–386.

[30] M. Schaefer and C. Umans. “Completeness in the Polynomial-Time Hierarchy: Part I: A Compendium”. In: *ACM SIGACT News* 33.3 (2002). Guest Column in L. A. Hemaspaandra, SIGACT News Complexity Theory Column 37, pp. 32–49.

[31] M. Schaefer and C. Umans. “Completeness in the Polynomial-Time Hierarchy: Part II: A Compendium”. In: *ACM SIGACT News* 33.4 (2002). Guest Column in L. A. Hemaspaandra, SIGACT News Complexity Theory Column 38, pp. 22–36.

[32] M. Schaefer and C. Umans. *Completeness in the Polynomial-Time Hierarchy: A Compendium*. Tech. rep. 2008.

[33] H. Spakowski and J. Vogel. “Θ_2^p-Completeness: A Classical Approach for New Results”. In: *Proc. of FST&TCS*. Vol. 1974. LNCS. Springer, 2000, pp. 348–360.

[34] L. J. Stockmeyer. “The polynomial-time hierarchy”. In: *Theor. Comput. Sci.* 3.1 (1976), pp. 1–22.

[35] K. W. Wagner. “More complicated questions about maxima and minima, and some closures of NP”. In: *Theor. Comput. Sci.* 51.1–2 (1987), pp. 53–80.

[36] K. W. Wagner. “Bounded query computations”. In: *Proc. of CoCo*. 1988, pp. 260–277.

[37] K. W. Wagner. “Bounded Query Classes”. In: *SIAM J. Comput.* 19.5 (1990), pp. 833–846.

[38] C. Wrathall. “Complete sets and the polynomial-time hierarchy”. In: *Theor. Comput. Sci.* 3.1 (1976), pp. 23–33.