Algorithmica (2013) 65:817–827
DOI 10.1007/s00453-012-9648-0

On the Exact Complexity of Evaluating Quantified k-CNF

Chris Calabro · Russell Impagliazzo · Ramamohan Paturi

Received: 30 June 2011 / Accepted: 31 March 2012 / Published online: 24 April 2012

Abstract We relate the exponential complexities $2^{s(k)n}$ of k-SAT and the exponential complexity $2^{s(\text{EVAL}(\Pi_2 3\text{-CNF}))n}$ of EVAL($\Pi_2$3-CNF) (the problem of evaluating quantified formulas of the form $\forall\vec{x}\exists\vec{y}\text{F}(\vec{x},\vec{y})$ where F is a 3-CNF in $\vec{x}$ variables and $\vec{y}$ variables) and show that $s(\infty)$ (the limit of $s(k)$ as $k\to\infty$) is at most $s(\text{EVAL}(\Pi_2 3\text{-CNF}))$. Therefore, if we assume the Strong Exponential-Time Hypothesis, then there is no algorithm for EVAL($\Pi_2$3-CNF) running in time 2^{cn} with $c<1$. On the other hand, a nontrivial exponential-time algorithm for EVAL($\Pi_2$3-CNF) would provide a k-SAT solver with better exponent than all current algorithms for sufficiently large k. We also show several syntactic restrictions of the evaluation problem EVAL($\Pi_2$3-CNF) have nontrivial algorithms, and provide strong evidence that the hardest cases of EVAL($\Pi_2$3-CNF) must have a mixture of clauses of two types: one universally quantified literal and two existentially quantified literals, or only existentially quantified literals. Moreover, the hardest cases must have at least $n-o(n)$ universally quantified variables, and hence only $o(n)$ existentially quantified variables. Our proofs involve the construction of efficient minimally unsatisfiable k-CNFs and the application of the Sparsification lemma.

R. Impagliazzo research was supported by the Simonyi Fund, the Bell Company Fellowship and the Fund for Math, and NSF grants DMS-083573, CNS-0716790 and CCF-0832797.
R. Paturi research was supported by NSF grant CCF-0947262 from the Division of Computing and Communication Foundations. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

C. Calabro
Google Inc., Mountain View, CA 94043, USA

R. Impagliazzo · R. Paturi (✉)
Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA 92093-0404, USA
e-mail: paturi@cs.ucsd.edu

Keywords Quantified Boolean formulas · Satisfiability

1 Introduction

From the viewpoint of exponential complexity of **NP**-complete problems, the most studied and best understood problems are probably the restricted versions of the formula satisfiability problem (SAT), in particular, k-SAT, the restriction of SAT to k-CNFs, and CNF-SAT, the restriction to general CNFs. There has been a sequence of highly nontrivial and interesting algorithmic approaches to these problems [7, 8, 11, 13–17], where the best known constant factor improvements in the exponent are of the form $1 - 1/\Theta(k)$ for k-SAT and $1 - 1/\Theta(\lg c)$ for CNF-SAT with at most cn clauses. More recently, the algorithmic approaches have been generalized to obtain improved exponential time algorithms for bounded-depth linear size circuits [6]. Also, a sequence of papers [4, 5, 9, 10] has shown many nontrivial relationships between the exponential complexities of these problems, and helped characterize their hardest instances (under the assumption that they are indeed exponentially hard).

In this paper, we consider the complexity of evaluating quantified formulas (quantified k-CNFs and k-DNFs). A quantified formula ϕ is an expression of the following form, $Q_1\vec{x}_1 \cdots Q_i\vec{x}_i$ F where F is either a k-CNF or a k-DNF, each $\vec{x}_j$ is a tuple of Boolean variables, and $Q_1, \ldots, Q_i$ is a sequence of alternating quantifiers. Similarly, we define a quantified circuit to be a quantified expression $Q_1\vec{x}_1 \cdots Q_i\vec{x}_i$ C where C is a Boolean circuit. For $k = 2$, the problem of evaluating quantified k-CNF or quantified k-DNF can be solved in linear time [1], but for $k \geq 3$, it is PSPACE-complete. Note that the exponential complexity of evaluating arbitrary quantified circuits C, which is at least as large as that of evaluating quantified 3-CNF, is at most $2^n \text{poly}(|C|)$: evaluation of a circuit can be carried out in time polynomial in its size, and backtracking need explore at most 2^n paths, each of length at most n, where n is the number of variables of the input (and this only requires polynomial space as well). We ask whether there exist nontrivial exponential-time algorithms for evaluating quantified k-CNF and quantified k-DNF for $k \geq 3$.

Since we cannot hope for a nontrivial, unconditional result about the exponential complexity of quantified formulas, we will relate the exponential complexity of evaluating quantified formulas to the exponential complexity of k-SAT. As in [9], let $s(k) = \inf\{c | \exists$ a randomized algorithm for k-SAT with time complexity $\text{poly}(m)2^{cn}$ for k-CNF formulas of size m over n variables$\}$. More generally, we define the exponential complexity of an evaluation problem Φ to be $s(\Phi) = \inf\{c | \exists$ a randomized algorithm for evaluating an instance ϕ of Φ with time complexity $\text{poly}(m)2^{cn}$ where m is the size of ϕ and n is the number of variables$\}$. While [9] did not explicitly use the words 'randomized algorithm' in its definition of $s(k)$ (it uses the word 'algorithm' without qualification), it meant to employ a broader class of algorithms (randomized algorithms) so as to define the constants $s(k)$ robustly.

Let **ETH** denote the Exponential-Time Hypothesis that $s(3) > 0$, i.e., 3-SAT requires 2^{cn} time for some $c > 0$. Let $s(\infty)$ denote the limit of the sequence $\{s(k)\}$. [9] proposed the open question whether $s(\infty) = 1$, which we will call the *Strong Exponential-Time Hypothesis* (**SETH**). Since the best known upper bounds for $s(k)$ are all of the form $1 - 1/\Theta(k)$, **SETH** is plausible.

As with CNF-SAT, we will consider various syntactic restrictions of quantified k-SAT to arrive at a *minimally complex* set of formulas which are as hard as the standard existentially quantified 3-CNFs as far as exponential complexity is concerned. However, merely bounding k does not seem to be enough. Our main restriction is to bound the number of alternations of quantifiers. Let $\Pi_i k$-CNF ($\Sigma_i k$-CNF) denote the set of quantified k-CNF of the form $Q_1\vec{x_1}\cdots Q_i\vec{x_i}$ F where F $\in k$-CNF, each $\vec{x_j}$ is a tuple of Boolean variables, and $Q_1,\ldots,Q_i$ is a sequence of i alternating quantifiers starting with a universal (existential) quantifier. Similarly, we define the formulas $\Sigma_i k$-DNF and $\Pi_i k$-DNF. Define EVAL(Φ) to be the problem of evaluating a quantified formula in Φ, where Φ is one of $\Pi_i k$-CNF, $\Pi_i k$-DNF, $\Sigma_i d$-CNF, and $\Sigma_i k$-DNF. When i is odd (even), EVAL($\Pi_i k$-CNF) (EVAL($\Sigma_i k$-CNF)) can be reduced in polynomial time to EVAL($\Pi_{i-1} k$-CNF) (EVAL($\Sigma_{i-1} k$-CNF)). Similarly, when i is even (odd), EVAL($\Pi_i k$-DNF) (EVAL($\Sigma_i k$-DNF)) can be reduced in polynomial time to EVAL($\Pi_{i-1} k$-DNF) (EVAL($\Sigma_{i-1} k$-DNF)). Using the earlier notation, we use s(EVAL(Φ)) to denote the exponential complexity of the evaluation problem Φ, where Φ is one of EVAL($\Pi_i k$-CNF), EVAL($\Pi_i k$-DNF), EVAL($\Sigma_i d$-CNF), and EVAL($\Sigma_i k$-DNF). Note that k-SAT is the same problem as EVAL($\Sigma_1 k$-CNF) (also, $s(k) = s$(EVAL($\Sigma_1 k$-CNF))) and that EVAL($\Pi_1 k$-CNF) is the problem of evaluating universally quantified k-CNF, which can be done in polynomial time.

Our main result is $s(\infty) \leq s$(EVAL($\Pi_2$3-CNF)). Thus **SETH** would imply that for EVAL($\Pi_2$3-CNF), we should not expect algorithms with time complexity of the form $2^{\delta n}$ for $\delta < 1$. Since EVAL($\Pi_i k$-CNF) is at least as hard when $i \geq 2$, $k \geq 3$, the exponential complexity of problems in the polynomial hierarchy would seem to top off rather early. Of course, **SETH** is a very strong assumption, so this evidence should be considered weak. Conversely, a single nontrivial algorithm for EVAL($\Pi_2$3-CNF) would provide a k-SAT solver better than all current algorithms for sufficiently large k.

We also show several syntactic restrictions of EVAL($\Pi_2$3-CNF) that have nontrivial algorithms, concluding that the hardest cases of EVAL($\Pi_2$3-CNF) must have a mixture of clauses of two types under the assumption **SETH**: one universally quantified literal and two existentially quantified literals, or only existentially quantified literals. Algorithmic design may benefit by concentrating on the hard cases, in much the same way that much of the progress in k-SAT solvers has been driven by focusing on the hard cases—when there are a linear number of clauses, only one solution, expanding variable-dependence graph, etc.

Lastly, we relate the exponential complexities s(EVAL(Φ)) in terms of each other for various Φ when the clause density is not too high, although it may well be that these are all simply 1. In particular, we show that the evaluation problem EVAL($\Pi_i k$-CNF) (EVAL($\Sigma_i k$-CNF)) for formulas with at most m clauses can be reduced to EVAL($\Pi_{i+1} k'$-DNF) (EVAL($\Sigma_{i+1} k'$-DNF)) formulas in polynomial time as long as $k' \geq 3$ and $m \in o(n^{k'-1})$. Similar statements hold for the problems EVAL($\Pi_i k$-DNF) and EVAL($\Sigma_i k$-DNF).

Williams [18] explored similar problems, namely exponential upper bounds for evaluating quantified CNF and k-CNF formulas with at most cn clauses and with no bound on the number of quantifier alternations. We will call such quantified formulas QCNF and k-QCNF respectively. Among other results, Williams [18] presents

an algorithm for EVAL(QCNF) that runs in time $O(2^{.773cn})$ time and space. He also presents an algorithm that runs in time $O(2^{.695n})$ and polynomial space for evaluating 3-QCNF for the special case of $c = 1$. Each of these algorithms outperforms exhaustive search only when c is very small, certainly no more than 2. This is probably not an accident: our results suggest that finding a substantially improved algorithm for moderate c, even for a small number of quantifier alternations, is unlikely.

The main step of our lower bound proof is to encode a k-CNF as a $\Pi_2 3$-CNF using only $o(n)$ additional variables. For this encoding, we construct a minimally unsatisfiable CNF formula with many clauses while using few additional variables. Another key technique in our proofs is the Sparsification lemma [5, 10]:

Lemma 1 (Sparsification lemma) *There is an algorithm A with the following property. For all $k \geq 2$, $\varepsilon \in (0, 1]$, and $\mathrm{F} \in k$-CNF with n variables, $A(k, \varepsilon, \phi)$ outputs a sequence $\mathrm{F}_1, \ldots, \mathrm{F}_s \in k$-CNF with $n^{O(k)}$ delay per formula such that*

1. $s \leq 2^{\epsilon n}$,
2. F *is satisfiable if and only if* $\bigvee_i \mathrm{F}_i$ *is satisfiable, and*
3. *for all* i, $|\mathrm{F}_i| \leq O(\frac{k}{\epsilon})^{3k} n$.

1.1 Notation

We assume that the readers are familiar with the standard vocabulary dealing with Boolean formulas such as conjunctive normal form (CNF), disjunctive normal form (DNF), k-CNF, and k-DNF. We use roman letters F, G, and H to denote such unquantified formulas. If F is a formula, then var(F) denotes set of variables occurring in F. If a is a partial assignment to the variables of F, F$|a$ denotes the restriction of F to a. |F| denotes the number of clauses in the CNF F. A CNF (DNF) F is regarded as a Boolean formula at times in which case it is a conjunction (disjunction) of clauses (terms) and as a set of clauses (terms) at other times. Intended meaning should be clear from the context. For example, for CNFs F and G, F $\subseteq$ G indicates that the CNFs F and G should be viewed as sets of clauses.

For $i \geq 1$ and $k \geq 1$, $\Pi_i k$-CNF, $\Pi_i k$-DNF, $\Sigma_i k$-CNF, and $\Sigma_i k$-DNF denote quantified formulas as defined earlier. We use greek letters ϕ and ψ to denote quantified formulas. If ϕ is a quantified formula where the variable x is existentially (universally) quantified, then we say that the corresponding literals are existentially (universally) quantified.

For convenience, we use the same symbol to denote the type of a formula as well as the set of such formulas. For example, k-CNF stands for the set of all k-CNFs.

We use SAT, CNF-SAT, and k-SAT to denote the standard satisfiability problems. For quantified formula classes Φ, we let EVAL(Φ) to denote the corresponding evaluation problem. For example, EVAL($\Pi_2 3$-CNF) is the problem of evaluating $\Pi_2 3$-CNF formulas. For each satisfiability problem Φ, Φ also stands for the set of satisfiable formulas of the appropriate type. For example, SAT stands for the set of all satisfiable Boolean formulas. Similarly, for each evaluation problem Φ, Φ stands for the set of true formulas of the appropriate type.

2 Lower Bound for EVAL($\Pi_2$3-CNF)

Our proof uses a construction similar to [3], Proposition 3.2, that allows us to reduce the clause width of a k-CNF using a small number of new quantified variables. For this, we need for all k' a "minimally unsatisfiable" k'-CNF which includes a large number of $k'-1$ clauses. Moreover, we would like to employ as few additional variables as possible, certainly no more than $o(n)$ additional variables. The following construction suffices for our purposes. An almost identical construction is independently obtained by Choongbum Lee [12]. However, it is an open question whether there are other constructions that employ fewer variables. Our construction differs from [3] primarily in its efficiency, since we are more concerned with exponential complexity rather than hardness for a class closed under polynomial time reductions, such as NP or coNP.

Say a pair (F, G) of CNFs is *minimally unsatisfiable* if and only if $\forall F' \subseteq F$ $((F' \wedge G) \notin \text{SAT} \iff F' = F)$. In other words, the pair (F, G) is minimally unsatisfiable if and only if $F \wedge G$ is unsatisfiable, but for any proper subset $F' \subsetneq F$, the formula $F' \wedge G$ is satisfiable. Say that $H \in \text{CNF}$ is *combinable* with (F, G) if and only if $|H| \leq |F|$ and $\text{var}(H) \cap \text{var}(F \wedge G) = \emptyset$. In this case, letting $f : H \to F$ be an arbitrary injection, we define

$$\text{combine}(H, F, G) = G \cup \{\{\bar{l}\} \cup f(C) \mid l \in C \in H\} \cup (F - \text{range}(f)),$$

i.e., starting with G, for each clause C of H and literal l of C, we add a clause meaning "l implies the clause of F corresponding to C"; and any clauses of F that are left over get added as-is, without adjoining a literal.

Lemma 2 *Let* H *be combinable with minimally unsatisfiable* (F, G) *and* $H' = \text{combine}(H, F, G)$. *For every assignment to the variables of* H, H|*a evaluates to true if and only if* $H'|a$ *is unsatisfiable*.

Proof This is simply because in H', after assigning to var(H), each clause $f(C)$ appears in the resulting formula if and only if some literal $l \in C$ is assigned true. Note that this does not depend on the specific injection $f : H \to F$. □

Lemma 3 *For all integers* $k \geq 3$ *and* $m \geq 1$, *one can construct in time* poly(m), $F \in (k-1)$-CNF *and* $G \in k$-CNF *such that*

- (F, G) *is minimally unsatisfiable*
- $|F| \geq m$
- $|\text{var}(F \wedge G)| \leq \frac{(k-1)^2}{k-2} \lceil m^{1/(k-1)} \rceil$.

Proof Let $l = \lceil m^{1/(k-1)} \rceil$. Consider a $(k-1) \times l$ matrix A of Boolean variables $x_{i,j}$ and the contradictory statements

1. "every row of A contains a 1",
2. "every choice of one entry from each row contains a 0".

We can express 1. by using at most $\frac{k-1}{k-2}l$ new variables $y_{i,j}$ and k-clauses G as follows. For each i, partition $x_{i,1},\ldots,x_{i,l}$ into $b=\lceil\frac{l}{k-2}\rceil$ blocks of size at most $k-2$. Add to G k-clauses expressing that $y_{i,1}$ is the OR of the variables of the first block; $y_{i,j}$ is the OR of $y_{i,j-1}$ and the variables of the jth block, for $j\in[2,b-1]$; and that the OR of $y_{i,b-1}$ and the variables of the bth block is true. Note that $y=z_1\vee z_2\vee\cdots\vee z_l$ can be expressed as an $(l+1)$-CNF since every Boolean function in n variables can be expressed as a CNF where each clause has width at most n. We can express 2. by using a set F of $(k-1)$-clauses where $|\mathrm{F}|=l^{k-1}\geq m$. Each clause in F is a disjunction of the negations of the variables in the matrix where each row contributes exactly one variable. F and G have the desired properties. □

Corollary 1 *For all $k,k'\geq 3$, given $\mathrm{H}\in k$-CNF with n variables $\vec{x}$ and m clauses, one can construct in time* poly(n), $\mathrm{H}'\in k'$-CNF *with n variables $\vec{x}$ and $\frac{(k'-1)^2}{k'-2}\lceil m^{1/(k'-1)}\rceil$ variables $\vec{y}$ such that $\forall a\in\{0,1\}^{\mathrm{var}(\mathrm{H})}$ $(\mathrm{H}(a)=1\leftrightarrow \mathrm{H}'|a\notin \mathrm{SAT})$. In particular, $\mathrm{H}\in$ SAT if and only if $\forall\vec{x}\exists\vec{y}\ \mathrm{H}'$ is false.*

Proof Use Lemma 3 to construct minimally unsatisfiable (F, G) with $\mathrm{F}\in(k'-1)$-CNF, $\mathrm{G}\in k'$-CNF, $|\mathrm{F}|=m$, $|\mathrm{var}(\mathrm{F}\wedge\mathrm{G})|\leq\frac{(k'-1)^2}{k'-2}\lceil m^{1/(k'-1)}\rceil$, and $\mathrm{var}(\mathrm{H})\cap\mathrm{var}(\mathrm{F}\wedge\mathrm{G})=\emptyset$. Let $\mathrm{H}'=\mathrm{combine}(\mathrm{H},\mathrm{F},\mathrm{G})$ and apply Lemma 2. □

We are now ready to prove our main theorem which shows that for all k, the exponential complexity $s(k)$ is at most $s(\mathrm{EVAL}(\Pi_2 3\text{-CNF}))$.

Theorem 1 *$s(\infty)\leq s(\mathrm{EVAL}(\Pi_2 3\text{-CNF}))$, where $s(k)$ is the limit of the nondecreasing sequence $s(k)$.*

Proof Fix k and $\epsilon>0$. We will show that $s(k)\leq s(\mathrm{EVAL}(\Pi_2 3\text{-CNF}))+4\epsilon$. Let F be a k-CNF with n variables. Use Lemma 1 to sparsify F into at most $2^{\epsilon n}$ many subformulas F_i so that each has at most $m=\lceil\frac{\epsilon n}{4}\rceil^2$ clauses for all sufficiently large n. By Corollary 1, with $k'=3$, for each F_i, we can construct in time poly(n) a $\phi_i\in\Pi_2 3$-CNF with $(1+\epsilon)n$ variables such that $\mathrm{F}_i\in$ SAT if and only if ϕ_i is false. So $\mathrm{F}\in$ SAT if and only if ϕ_i is false for some i. Evaluating each ϕ_i with some $\mathrm{EVAL}(\Pi_2 3\text{-CNF})$ solver with exponential complexity at most $s(\mathrm{EVAL}(\Pi_2 3\text{-CNF}))+\epsilon$, we get that

$$s(k)\leq(s(\mathrm{EVAL}(\Pi_2 3\text{-CNF}))+\epsilon)(1+\epsilon)+\epsilon\leq s(\mathrm{EVAL}(\Pi_2 3\text{-CNF}))+4\epsilon.$$ □

Remark 1 Our construction in the preceding proof can be done using $O(\sqrt{n})$ new variables rather than ϵn new variables. For the current proof, we did not need the advantage of fewer variables and hence we used a more relaxed bound on the number of clauses of the sparsified formulas.

The previous result puts us in an interesting situation. [9] showed that **ETH** implies $s(k)$ increases infinitely often as a function of k. Consider the following much weaker analogue.

Conjecture 1 *If* **ETH** *is true, then there exist $i\geq 2, k\geq 3$ such that $s(\mathrm{EVAL}(\Pi_i k\text{-CNF}))<s(\mathrm{EVAL}(\Pi_{i+1}(k+1)\text{-CNF}))$.*

Conjecture 1, together with Theorem 1, would imply that $s(\infty) < 1$. To see this, assume that for some i and k, $s(\text{EVAL}(\Pi_i k\text{-CNF})) < s(\text{EVAL}(\Pi_{i+1}(k+1)\text{-CNF}))$. Since $s(\text{EVAL}(\Pi_{i+1}(k+1)\text{-CNF})) \leq 1$, it follows that $s(\text{EVAL}(\Pi_2 3\text{-CNF})) \leq s(\text{EVAL}(\Pi_i k\text{-CNF})) < 1$, which implies $s(\infty) < 1$ by Theorem 1.

3 Algorithms for Two Special Cases

In this section, we show that quantified CNF-SAT has a poly time algorithm if each clause has at most one existentially quantified literal and that $\text{EVAL}(\Pi_2 3\text{-CNF})$ has a nontrivial algorithm if each clause has at least one universally quantified literal. The purpose of such theorems is to find where the hard cases of $\text{EVAL}(\Pi_i k\text{-CNF})$ lie.

In a quantified CNF, we say that two clauses A, B *disagree* on variable x if and only if one of A, B contains x and the other contains $\neg x$. A, B are *resolvable* if and only if they disagree on exactly one variable x, and the *resolvent* is $\text{resolve}(A, B) = A \cup B - \{x, \neg x\}$. Also, define $\text{eliminate}(A)$ to be A after removing any universally quantified literal l of A for which no existentially quantified literal l' of A occurs after l in the order of quantification. A, B are *Q-resolvable* if and only if they are resolvable on an existential variable, and the *Q-resolution* operation is $\text{qresolve}(A, B) = \text{eliminate}(\text{resolve}(A, B))$. Büning, Karpinski and Flögel [2] introduced the Q-resolution proof system and showed that it is sound and complete, in the sense that a quantified CNF is false if and only if the empty clause can be derived by first replacing each clause C by $\text{eliminate}(C)$ and then repeatedly applying qresolve to generate new clauses.

Theorem 2 *Let $\phi = \forall \vec{x}_1\, \exists \vec{x}_2\, \ldots \forall \vec{x}_n$ F be a quantified Boolean formula where each clause of* F $\in$ CNF *has at most one existentially quantified literal. Then ϕ can be evaluated in polynomial time.*

Proof Since each clause contains at most one existentially quantified literal, any application of the Q-resolution operator will produce the empty clause. Letting ϕ' be ϕ after replacing each clause C in F by $\text{eliminate}(C)$, ϕ is true if and only if ϕ' does not contain the empty clause and no two clauses are Q-resolvable in ϕ'. □

Theorem 3 *Let $\phi = \forall \vec{x}\, \exists \vec{y}$ F, where* F $\in$ 3-CNF. *If each clause of* F *has at least one universally quantified literal, then ϕ can be evaluated in time 2^{cn} for some $c < 1$.*

Proof Let $G = (V, E)$ where the vertex set V consists of the existentially quantified literals of ϕ and there is an edge between from vertex a to vertex b with label c if $(\bar{a} \vee b \vee \bar{c})$ is a clause of F, a, b are existentially quantified literals and c is a universally quantified literal.

While we define the graph G assuming that each clause has exactly two existentially quantified literals and one universally quantified literal, other cases can be handled as well. For example, if $(a \vee \bar{b} \vee \bar{c})$ is a clause with a as the sole existentially quantified variable and the remaining two literals $\bar{b}$ and $\bar{c}$ are universally quantified, we introduce the edge $(\bar{a}, a)$ with the label $b \wedge c$. 2-clauses can be treated similarly.

Branching rules used in [18] can also be applied to reduce the formulas with these special types of clauses (clauses with more than one universally quantified literal or 2-clauses) to formulas with clauses containing exactly one universally quantified literal and two existentially quantified literals.

A *consistent path* in G is a path such that no two edge labels disagree, a *consistent cycle* is a consistent path that is a cycle. Consistency of a path can be checked in polynomial time since all labels are products of literals. Then ϕ is false if and only if there is an existentially quantified variable z such that there is a consistent cycle in G containing nodes $z, \bar{z}$.

Notice that a simple cycle containing nodes $z, \bar{z}$ cannot use more edge labels than $4\epsilon n$, where ϵn is the number of existentially quantified variables. So we can test whether there is a cycle in G containing $z, \bar{z}$ in time at most $\text{poly}(|\phi|)\binom{2n}{4\epsilon n} \leq \text{poly}(|\phi|)2^{2h(2\epsilon)n}$ where h is the binary entropy function.

If ϵ is large, we can exhaustively search over all settings of the universally quantified variables and then use a k-SAT[3] algorithm on the rest. If our k-SAT[3] algorithm runs in time $2^{s(3)n}$, then the combined algorithm runs in time at most $\min\{2^{(1-\epsilon)n+\epsilon s(3)n}, 2^{2h(2\epsilon)n}\}$, which is maximized at about $\epsilon = .0526$ for $s(3) = .387$ and yields a run time of at most $2^{.964n}$. If $s(3)$ is smaller, this value would be even smaller. Note that there is already an algorithm for k-SAT[3] with $s(3) < .387$ [8]. □

It is unlikely that we can find a polynomial time algorithm for the variant where each clause has at least one universally quantified literal because of the following.

Theorem 4 *The language of true $\Pi_2$3-CNF formulas where each clause has at least one universally quantified literal is* coNP*-complete.*

Proof The language is in coNP because a witness for falsehood is a consistent cycle containing some existential variable z and its negation $\bar{z}$. To show coNP-hardness, we reduce from 3-UNSAT. Let $\text{F} \in$ 3-CNF have n variables $\vec{x}$ and m clauses. Assume without loss of generality that m is even, otherwise, just repeat some clause twice. Let $n' = \frac{m}{2}$, $\vec{y} = (y_1, \ldots, y_{n'})$ be new variables, and consider the following contradictory 2-CNF:

$$G = \{(y_i \to y_{i+1}), (y_{i+1} \to y_i) \mid i \in [n' - 1]\} \cup \{(y_{n'} \to \bar{y}_1), (\bar{y}_1 \to y_{n'})\}.$$

Then the pair of formulas $(\text{G}, \emptyset)$ is minimally unsatisfiable and $|G| = m$. By Lemma 2, $\phi = \forall\vec{x}\exists\vec{y}$ combine$(\text{F}, \text{G}, \emptyset)$ is a $\Pi_2$3-CNF with at least one universally quantified variable in each clause and such that $\text{F} \in$ SAT if and only if ϕ is false. □

The coNP-hardness of Theorem 4 also follows from [3], Proposition 3.2, if one observes that the number of universal variables in each constraint constructed there is 1.

Remark 2 As pointed out by one of the referees, the construction in Theorem 4 also shows that if **ETH** is true, then the exponential complexity of EVAL($\Pi_2$3-CNF) with at least one universally quantified literal per clause is greater than 0.

4 Reduction to a Canonical Form

Theorems 1 and 3 together suggest that clauses containing only existentially quantified literals and clauses with a mixture of universally quantified literals and existentially quantified literals are both essential to make an instance of EVAL($\Pi_2$3-CNF) hard. Below we present further evidence of this.

Let $\Pi_2$3-CNF$'$ be the special case of $\Pi_2$3-CNF where each clause is one of just two types:

(1) contains one universally quantified literal and two existentially quantified literals
(2) contains only existentially quantified literals.

We then obtain the following theorem, which uses the same idea as in [18] and follows from the algorithm therein.

Theorem 5 $s(\text{EVAL}(\Pi_2\text{3-CNF}')) < 1 \implies s(\text{EVAL}(\Pi_2\text{3-CNF})) < 1.$

Proof Let A be an algorithm with run time $\leq O(2^{cn})$ with $c < 1$ for EVAL($\Pi_2$3-CNF$'$). Given a $\Pi_2$3-CNF, do this: while there is a clause other than types (1) or (2), i.e., a clause C with at least one universally quantified variable and at most one existentially quantified variable, reject if there is no existentially quantified variable in C, but otherwise branch on the universally quantified variables. One of the branches will force the existentially quantified variable. Each formula at a leaf has only clauses of type (1) or (2), so apply algorithm A. To simplify the calculations, since our goal is only to show that $s(\text{EVAL}(\Pi_2\text{3-CNF})) < 1$, we will consider a simpler algorithm at the expense of the constant in the exponent: while there is a clause C with at least one universally quantified variable and at least one existentially quantified variable, spawn seven subproblems, one for each setting of the three variables in the clause except for the setting that makes the clause false. This algorithm solves the general case in time at most $O(2^{dn})$ where $d = \max\{c, \frac{\lg 7}{3}\} < 1$. □

We may also assume, without loss of generality, that in a $\Pi_2 k$-CNF the number of existentially quantified variables is $o(n)$, since otherwise we could branch on every possible setting of the universal variables and then invoke a k-SAT solver to obtain a nontrivial algorithm for EVAL($\Pi_2 k$-CNF).

5 Parameter trade-off at higher levels of the hierarchy

In the next theorem, we show that when confronted with evaluating a $\Pi_i k$-CNF instance, one may be able to reduce k to $k' < k$ if one is willing to increase i by 1, and the input clause density is not too high. For a function m, let $\Sigma_i k\text{-CNF}_m$ ($\Pi_i k\text{-CNF}_m$) be $\Sigma_i k$-CNF ($\Pi_i k$-CNF) but where the number of clauses is promised to be at most $m(n)$. Similar notation is used for the quantified k-DNFs.

Theorem 6 *Let* $k, k' \geq 3, m \in o(n^{k'-1})$. *For all* $i \geq 1$,

$$s(\text{EVAL}(\Pi_i k\text{-CNF}_m)) \leq s(\text{EVAL}(\Pi_{i+1} k'\text{-DNF}))$$
$$s(\text{EVAL}(\Sigma_i k\text{-CNF}_m)) \leq s(\text{EVAL}(\Pi_{i+1} k'\text{-DNF})).$$

Remark 3 The first (second) inequality of the theorem is trivial for odd (even) i since EVAL($\Pi_i k$-CNF) (EVAL($\Sigma_i k$-CNF)) can be reduced in polynomial time to EVAL($\Pi_{i-1} k$-CNF) (EVAL($\Sigma_{i-1} k$-CNF)).

Remark 4 Inequalities dual to those in Theorem 6 also hold for quantified k-DNFs.

Proof Consider a quantified formula $\phi = Q_1 \vec{x_1} \cdots Q_i \vec{x_i}$ F where $Q_1, \ldots, Q_i$ are alternating quantifiers, each $\vec{x_j}$ is a tuple of Boolean variables, and F is a k-CNF with n variables $\vec{x}$ and at most $m(n)$ clauses. Using Corollary 1, $\forall \epsilon > 0$ and for sufficiently large n, we can construct in polynomial time a k'-CNF F$'$ such that

- F$'$ has $(1+\epsilon)n$ variables: $\vec{x}$ and ϵn new variables $\vec{y}$.
- For each assignment a to $\vec{x}$, F$(a) = 1$ if and only if F$'|a \notin$ SAT.

So

$$\begin{aligned} \phi = Q_1 \vec{x_1} \cdots Q_i \vec{x_i}\ \mathrm{F} \quad &\Longleftrightarrow \quad Q_1 \vec{x_1} \cdots Q_i \vec{x_i}\ \neg \exists \vec{y}\ \mathrm{F}' \\ &\Longleftrightarrow \quad Q_1 \vec{x_1} \cdots Q_i \vec{x_i} \forall \vec{y}\ \neg \mathrm{F}' = \phi' \end{aligned}$$

If $\phi \in \Pi_i k$-CNF ($\Sigma_i k$-CNF), then $\phi' \in \Pi_{i+1} k'$-CNF ($\Sigma_{i+1} k'$-CNF). Moreover, ϕ' has at most $(1+\epsilon)n$ variables where ϵ can be chosen to be arbitrarily small for all sufficiently large n. Theorem follows from the definition of the exponential complexity. □

It would be nice to eliminate the requirement that ϕ have at most $o(n^{k'-1})$ clauses. One might try to do this by first sparsifying ϕ, but the possibly complex quantification of the variables prevents this.

Acknowledgements We thank the reviewers for their helpful comments and corrections.

References

1. Aspvall, B., Plass, M.F., Tarjan, R.E.: A linear-time algorithm for testing the truth of certain quantified boolean formulas. Inf. Process. Lett. **8**(3), 121–123 (1979)
2. Kleine Büning, H., Karpinski, M., Flögel, A.: Resolution for quantified boolean formulas. Inf. Comput. **117**(1), 12–18 (1995)
3. Chen, H.: Existentially restricted quantified constraint satisfaction. Inf. Comput. **207**(3), 369–388 (2009)
4. Calabro, C., Impagliazzo, R., Kabanets, V., Paturi, R.: The complexity of unique k-SAT: an isolation lemma for k-CNFs. In: Proceedings of the Eighteenth IEEE Conference on Computational Complexity, 386–393 May 2008. Journal of Computer and Systems Sciences, vol. 74, pp. 135–144 (2003). Preliminary version
5. Calabro, C., Impagliazzo, R., Paturi, R.: A duality between clause width and clause density for SAT. In: CCC '06: Proceedings of the 21st Annual IEEE Conference on Computational Complexity, pp. 252–260. IEEE Computer Society, Washington (2006)
6. Calabro, C., Impagliazzo, R., Paturi, R.: The complexity of satisfiability of small depth circuits. In: Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10–11, 2009. Revised Selected Papers, pp. 75–85. Springer, Berlin (2009)
7. Dantsin, E., Wolpert, A.: An improved upper bound for SAT. In: Bacchus, F. (ed.) Lecture Notes in Computer Science, vol. 3569, pp. 400–407. Springer, Berlin (2005)

8. Timon, H.: 3-sat faster and simpler - unique-sat bounds for PPSZ hold in general. In: Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science, pp. 277–284 (2011)
9. Impagliazzo, R., Paturi, R.: The complexity of k-SAT. In: IEEE Conference on Computational Complexity, vol. 14, pp. 237–240 (1999)
10. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexit? J. Comput. Syst. Sci. **63**, 512–530 (1998). Preliminary version In: 39th Annual IEEE Symposium on Foundations of Computer Science, pp. 653–662 (1998)
11. Iwama, K., Seto, K., Takai, T., Tamaki, S.: Improved randomized algorithms for 3-sat. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) Algorithms and Computation. Lecture Notes in Computer Science, vol. 6506, pp. 73–84. Springer, Berlin (2010)
12. Lee, C.: On the size of minimal unsatisfiable formulas. ArXiv E-prints (2008)
13. Paturi, R., Pudlák, P., Saks, M.E., Zane, F.: An improved exponential-time algorithm for k-SAT. J. ACM **52**(3), 337–364 (2005). Preliminary version In: 39th Annual IEEE Symposium on Foundations of Computer Science, pp. 628–637 (1998)
14. Paturi, R., Pudlák, P., Zane, F.: Satisfiability coding lemma. In: Preliminary version in 38th Annual Symposium on Foundations of Computer Science, vol. 566 (1999). 1997
15. Daniel, R.: Improved bound for the PPSZ/Schöning-algorithm for 3-sat. J. Satisf. Boolean Model. Comput. **1**(2), 111–122 (2006)
16. Schöning, U.: A probabilistic algorithm for k-sat based on limited local search and restart. Algorithmica **32**(4), 615–623 (2002)
17. Schuler, R.: An algorithm for the satisfiability problem of formulas in conjunctive normal form. J. Algorithms **54**(1), 40–44 (2005)
18. Williams, R.: Algorithms for quantified boolean formulas. In: Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 299–307 (2002)