The Complexity of AND-Decomposition of Boolean Formulas

[a,b]Pavel Emelyanov, [a,c]Denis Ponomaryov

[a)]Institute of Informatics Systems and [b)]Novosibirsk State University, Novosibirsk, Russia
[c)]Institute of Artificial Intelligence, University of Ulm, Ulm, Germany
emelyanov@iis.nsk.su, ponom@iis.nsk.su

Abstract

Decomposition of boolean functions is an important research topic having a long history and a wide range of applications in the logic circuit synthesis, (hyper)graph/game theory, and combinatorial optimization. AND-decomposition means representing a boolean function as a conjunction of two (or several) functions sharing a given subset of variables. If the shared subset is empty, the decomposition is called disjoint. Though numerous heuristics and approaches to computing more general forms of decompositions have been developed, the algorithmic complexity of AND-decomposition remained unknown. The renewed interest to this question has been motivated by the recent research on decomposition of theories in expressive logics and modern developments in the analysis of complex systems, combinatorial optimization, and circuit design. In this paper, we study the complexity of AND-decomposition for various representations of boolean functions (CNF/DNF/Full DNF/ANF), prove a number of complexity bounds, and identify tractable cases. The positive complexity results of this paper follow from tractability of multilinear polynomial factorization over the finite field of order 2, for which we provide a polytime factorization algorithm based on identity testing for partial derivatives of multilinear polynomials.

1 Introduction

Decomposition of boolean functions is an important research topic having a long history and a wide range of applications. Among other application fields such as game and graph theory, it has attracted the most attention in the logic circuit synthesis. Decomposition is related to the algorithmic complexity and practical issues of implementation of electronic circuits, their size, time delay, and power consumption. The report [16] contains an extensive survey of decomposition methods till the mid–1990's. The results of the next fifteen years of research are presented in [13, 23, 11, 3, 6, 9, 5]. To position our paper in this broad research landscape, we describe some key questions related to decomposition and indicate the specifics of our work.

Representations of Boolean Functions. In general, the known approaches to decomposition differ in representations of boolean functions they are applicable to. The form in which a boolean function is given directly influences the computational complexity of operations on the function, as studied in the field of Knowledge Compilation [7]. The representation of an input function is also related to applicability of the existing methods of circuit design. In the paper, we consider several well-known normal forms for boolean functions, CNF, DNF, Full DNF, and ANF, but omit questions of translation between different representations.

Inputs in the above mentioned normal forms are widely used in the logic circuit synthesis: [16, 19, 20, 13, 11]. An input to SAT solvers is usually given in CNF which is the "Product of Sums" (PoS) representation of boolean functions. Due to the efficiency of SAT solvers, there has been developed quite a number of SAT-based approaches to decomposition of functions given in CNF, see e.g. [9, Ch. 5, 6], [5]. Another universal way to define a boolean function is DNF which is "Sum of Products" (SoP). Approaches to logic synthesis based on SoP inputs are one of the best elaborated from the point of view mapping to architectures. In this connection, decomposition methods for DNF are widely presented in literature.

Both, CNF and DNF can be easily constructed from zeros and units of a boolean function. One more advantage of these forms in the scope of decomposition is that both logic and algebraic approaches can be applied. From 1950s and till the invention of BDDs, these PoS and SoP forms have been used as the principle underlying data structures in the computer–aided logic design. They are still widely used, in particular, because of their generic nature allowing the user to apply optimization and decomposition simultaneously without additional expenses on transformation of representation. Decomposition of positive boolean formulas (known also as monotone formulas) given in CNF/DNF attracted particular attention in game and reliability theory (see the introduction in [3] for a summary of literature). These appear also as a natural abstraction of read-once formulas, in which every variable occurs only once and thus, the polarity of variables does not matter for certain operations on such formulas [8]. Boolean functions in Full DNF (i.e. given by explicit enumeration of satisfying vectors) have been considered in the logic synthesis in the scope of lookup tables (LUTs), which are a space consuming representation, but allow for very efficient operations on their content [11].

Another well-known SoP–like form is the Algebraic Normal Form (ANF, Zhegalkin polynomial or Reed–Muller canonical form). From the algebraic point of view, ANF is a linear multivariate polynomial over the finite field of order 2. In comparison to DNF, the advantages of ANF that recently made this form popular again are a more natural and compact representation of some classes of boolean functions (for instance, arithmetical functions, coders/cyphers, etc; some researchers conjectured that this basis is more economical in general), a more natural mapping to some circuit technologies (FPGA–based and nanostructure–based electronics), and good testability properties.

Proposed about thirty years ago [4], Binary Decisions Diagrams (BDDs) are considered as the most significant breakthrough in the logic design. BDD-based techniques are very popular among decomposition methods for boolean functions, see [14, 1, 23] and [9, Ch. 3]. BDDs provide a canonical form of boolean functions and allow for fast manipulations with them, however a wrong node ordering may cause "combinatorial explosion". Finding an optimal ordering is a NP-hard problem, but there are heuristics which are known to provide good results.

Nevertheless some authors argued [5], [9, Ch. 5], [12] that BDDs are not suitable enough for design of circuits over real-world boolean functions, in particular, due to the combinatorial explosion problem. It turns out that in some cases, BDDs may not be appropriate representation for computing decomposition [13]. Contrary to BDD-approaches to decomposition, in [5], the authors argued an approach based on Quantified Boolean Formulas in which the underlying data structure is given by formulas in the prenex normal form. Although the fact of using QBFs might seem to bring computational difficulties, the approach gives a number of interesting results from the point of view of logic design. Quantification allows for a succinct representation, while good performance and quality of decomposition are achieved by solving the Minimally Unsatisfiable Subformula Problem and by using the Craig interpolation theorem. These results belong to the "SAT-based" class of decomposition algorithms (see also [9, Ch. 6]).

In this paper, we consider the PoS/SoP–representations of boolean functions: CNF, DNF, Full DNF, and ANF. We specially consider positive CNF and DNF due to the above mentioned applications. Interestingly, positive DNF has a number of set theoretic and hypergraph interpretations which makes this form particular interesting from the point of view of combinatorics. In particular, our solution to disjoint decomposition of positive formulas in DNF given in this paper shows tractability of a variant of the well-known NP-complete Hypergraph 2-Coloring Problem. There are other forms which are similar to PoS/SoP. For instance, there is a SoP–like form based on "additive" XOR and "multiplicative" OR[1]. We note that our results on AND–decomposition are applicable to this basis as well.

Types of Decomposition. Typically one is interested in decompositions of the form $F = F_1 \odot \ldots \odot F_k$ where $\odot \in \{\text{OR, AND, XOR}\}$. Bi-decomposition is the most important case of decomposition of boolean functions. Even though it may not be stated explicitly, this case is considered in many papers: [14, 20, 1, 13, 6, 3, 5], [9, Ch. 3–6]. Bi-decomposition has the form: $F(X) = \pi(F_1(\Sigma_1, \Delta), F_2(\Sigma_2, \Delta))$, where $\pi \in \{\text{OR, AND, XOR}\}$, $\Delta \subseteq X$, and $\{\Sigma_1, \Sigma_2\}$ is a partition of the variables $X \setminus \Delta$. As a rule, a decomposition into more than two components can be obtained by iterative computation of bi-decomposition. The well–known examples of bi-decomposition are Boole's or Shannon's Expansions: $F = xF_{x=1} \vee \neg xF_{x=0} = (x \vee F_{x=0})(\neg x \vee F_{x=1})$. They can be classified as OR/AND–decompositions respectively and play an important role in the boolean function analysis and switching circuit design. In some sense, a complete and self-contained solution to bi-decomposition of arbitrary functions is described in series of papers by Steinbach et al. [23]. It allows to verify whether a given boolean function is decomposable wrt a given variable partition and to compute its components. The solution however implies a number of steps which may be intractable.

If $\Delta = \emptyset$ then decomposition is called disjoint and considered as optimal for many reasons. Sometimes requirements to decompositions are strengthened by additional optimality criteria, with the most popular being balancedness of the variable partition. It should be noticed that these additional requirements usually imply solving computationally hard problems.

In [2, 3], Bioch studies computational properties of modular decompositions based on a generalization of Shannon's Expansion. A set of variables A is called modular set of a boolean function $F(X)$ if F can be represented as $F(X) = H(G(A), B)$, where $\{A, B\}$ is a partition of X and H, G are some boolean functions. The function $G(A)$ is called component of F and a modular decomposition is obtained from iterative decomposition into such components. It is shown that in general it is coNP-complete to decide whether a subset of variables is modular, however for monotone functions in DNF this problem is tractable. The complexity of finding a modular tree representing all modular sets of a monotone boolean function is $O(n^5N)$ where n is the number of variables and N is the number of products in DNF.

We note that a function may have a modular or bi-decomposition, but may not be AND-decomposable, since this form of decomposition requires representation of a function strictly as a conjunction. Thus, AND–decomposition can be viewed as a special case of modular and bi-decomposition. Our results demonstrate that deciding even this special case of decomposability is coNP-complete for formulas given in CNF and DNF. On the other hand, we show tractability of computing AND-decompositions of formulas given in the forms: positive CNF and DNF, Full DNF, and ANF. It is not obvious, whether the technique used by Bioch for positive DNF is applicable to the case of AND-decomposition. We note however that in our Lemma 2, the idea of computing decomposition components resembles the final step of constructing components in [3, Sect. 2.9].

How to Partition the Variables. This is the principal problem in decomposition of boolean functions. Constructing modular sets [2, 3] is one of possible ways of solving this problem. As already mentioned, it becomes feasible for monotone functions in DNF. Once a modular set is given, the corresponding component of decomposition can be easily computed.

Usually, if a partition of the variables is given, then a decomposition algorithm either finds components corresponding to this partition, or says that decomposition does not exist. If a partition is not given, then either the variable sets $\Delta, \Sigma_1, \Sigma_2$ are guessed before the algorithm is run (BDD– and PoS/SoP–based approaches [14, 20, 13]) or they evolve during decomposition (SAT–based approaches

[1]Of course, this form is slightly exotic (in particular, this basis is not complete).

[9, Ch. 5, 6] and [5]). In [1], a heuristic based method for variable partitioning is described and proved to be efficient in practice.

In [6], the authors propose a graph–theoretical approach. To partition the variable set, the authors describe a procedure to build an undirected "Blocking Edge Graph" and argue its efficiency. However, the procedure essentially relies on massive checking whether boolean functions constructed by some (simple) manipulations are equal to zero or not. Obviously, the efficiency of this step strongly depends on representation of boolean functions; for some of them this problem is NP-complete. Then the mentioned procedure detects a minimum vertex cut. Notice that the complexity of this problem wrt some optimality criteria (balancedness-like) increases and the problem can easily become NP-complete.

In this paper, we note the important fact that every AND–decomposable function given in one of the considered forms (CNF, DNF or ANF) uniquely defines the finest partition of its variables. For formulas in CNF/DNF, this follows from a property of a large class of logical calculi shown in [17]. For formulas in ANF, a similar result follows from the fact that the ring of (multivariate) polynomials over the finite field of order 2 is a unique factorization domain. One of the main contributions of the paper is that we identify cases when computing AND-decomposition is tractable even if a variable partition is not given.

Logic vs Algebraic Decomposition.

Approaches to decomposition of boolean functions can be classified into logic and algebraic. The first are based on equivalent transformations of formulas in propositional logic. The second ones consider boolean functions as algebraic objects with the corresponding transformation rules. The most elaborated representation is polynomials, usually over finite fields, among which $\mathbb{F}_2$ (the Galois field of order 2) is the best known. Then AND–decomposition corresponds to factorization of multivariate polynomials over $\mathbb{F}_2$. It is known (Theorem 1.6 in [21]) that a polynomial $F(x_1, \ldots, x_m)$ of the total degree n over all its variables can be factored over a finite field of order p^r in time that is polynomial in n^m, r, and p. The state of the research on this problem is described in [24]. Shpilka and Volkovich [22] noted the strong relation between polynomial factorization and polynomial identity testing (i.e. testing equality to the zero polynomial). It follows from their results that a multilinear polynomial over $\mathbb{F}_2$ can be factored in time that is cubic in the size of the polynomial (given as a symbol sequence). We provide a factorization algorithm for multilinear polynomials over $\mathbb{F}_2$ which runs in cubic time and is based on identity testing for partial derivatives of a product of polynomials obtained from the input one. The product is computed only once in contrast to the approach of [22]. Moreover, we show that the algorithm can be implemented without computing the product of polynomials explicitly, thus contributing to efficiency of factorization of large input polynomials.

In general, logic-based approaches to decomposition are more powerful and achieve better results than algebraic ones: a boolean function can be decomposable logically, but not algebraically, since boolean divisors of a boolean function can differ from its algebraic factors [9, Ch. 4]. In our work, we follow the logic approach to decomposition, but show that tractability of multilinear polynomial factorization over $\mathbb{F}_2$ gives polytime decomposition algorithms for boolean functions in positive DNF and Full DNF.

2 Preliminaries

2.1 Basic Facts about AND-Decomposability

Let us introduce some conventions and notations. For a boolean formula φ, we denote the set of its variables by $\texttt{var}\,(\varphi)$. If Σ is a set of propositional variables and $\texttt{var}\,(\varphi) \subseteq \Sigma$, then we say that the formula φ is *over variables* Σ (or *over* Σ, for short); $\texttt{taut}(\Sigma)$ denotes a valid formula over Σ. A literal is either a variable (positive literal) or negation of a variable (negative literal). We call φ *positive* if it does not contain negative literals. If ξ and ξ' are clauses (or conjuncts, respectively), then the notation $\xi' \subseteq \xi$ means that ξ' is a subclause (subconjunct) of ξ, i.e. ξ' is given by a non-empty subset of literals from ξ. If φ is in CNF (DNF, respectively), then a clause (conjunct) ξ of φ is called *redundant* in φ if there exists another clause (conjunct) ξ' of φ such that $\xi' \subseteq \xi$.

We now define the main property of boolean formulas studied in this paper, the definition is adopted from [17], where it is given in a general form.

Definition 1 (Decomposability) *A boolean formula φ is called AND–decomposable wrt a (possibly empty) subset of variables $\Delta \subseteq \texttt{var}(\varphi)$ (or Δ–decomposable, for short) if it is equivalent to the conjunction $\psi_1 \wedge \psi_2$ of some formulas ψ_1 and ψ_2 such that:*

1. $\texttt{var}(\psi_1) \cup \texttt{var}(\psi_2) = \texttt{var}(\varphi)$;
2. $\texttt{var}(\psi_1) \cap \texttt{var}(\psi_2) \subseteq \Delta$;
3. $\texttt{var}(\psi_i) \setminus \Delta \neq \varnothing$, *for* $i = 1, 2$.

The formulas ψ_1 and ψ_2 are called Δ–decomposition components of φ. Note that ψ_1 and ψ_2 partition the set of variables $\texttt{var}(\varphi) \setminus \Delta$ and may share the variables only from Δ. We say that φ is Δ–decomposable with a variable partition $\{\Sigma_1, \Sigma_2\}$ if φ has some Δ-decomposition components ψ_1 and ψ_2 over the variables $\Sigma_1 \cup \Delta$ and $\Sigma_2 \cup \Delta$, respectively.

Note that a similar definition could be given for OR–decomposability, i.e. for decomposition into the disjunction of ψ_1 and ψ_2. Clearly, a formula φ is AND–decomposable wrt $\Delta \subseteq \texttt{var}(\varphi)$ iff $\neg\varphi$ is OR–decomposable wrt Δ.

Example 1 *The formula in CNF $(x \vee \neg d) \wedge (u \vee \neg x \vee d) \wedge (u \vee x \vee d)$ is equivalent to the conjunction of $x \vee \neg d$ and $u \vee d$ and hence, $\{d\}$–decomposable (with the variable partition $\{\{x\}, \{u\}\}$).*

The positive formula in DNF $(x \wedge u) \vee (x \wedge v) \vee (y \wedge u) \vee (y \wedge v)$ is $\varnothing$–decomposable into the components $x \vee y$ and $u \vee v$.

Note that Definition 1 is formulated with the two components ψ_1 and ψ_2, which in turn can be Δ–decomposable formulas. Since at each decomposition step, the variable sets of the components must be proper subsets of the variables of the original formula φ, the decomposition process necessarily stops and gives formulas which are non-decomposable. The obtained formulas define some partition of $\texttt{var}(\varphi) \setminus \Delta$ and the fact below (which follows from a property of a large class of logical calculi shown in [17]) says that this variable partition is unique.

Fact 1 (Uniqueness of Decompositions - Corollary of Thm. 1 in [17])
Let φ be a boolean formula and $\Delta \subseteq \texttt{var}(\varphi)$ be a subset of its variables. If φ is Δ-decomposable, then there is a unique partition $\{\pi_1, \ldots, \pi_n\}$ of $\texttt{var}(\varphi) \setminus \Delta$, $2 \leqslant n$, such that φ is equivalent to $\bigwedge\{\psi_i \mid \texttt{var}(\psi_i) = \pi_i \cup \Delta,\ i = 1, \ldots, n\}$, where each formula ψ_i is not Δ-decomposable.

This means that *any* possible algorithm[2] for decomposing a formula into components could be applied iteratively to obtain from a given φ some formulas ψ_i, $i = 1, \ldots, n$, which are non-decomposable and *uniquely* define a partition of the variables of φ "modulo" Δ.

[2]Existence and complexity of decomposition algorithms in various logics have been studied in [15, 10, 18, 17].

2.2 The Computational Problems Considered in the Paper

In the text, we omit subtleties related to efficient encoding of input sets of variables, boolean formulas (given in CNF, DNF, or ANF), and polynomials, assuming their representation as symbol sequences. The complexity of each computational problem below will be defined wrt the size of the input formula/polynomial given in this way.

$\boxed{\Delta\texttt{Dec}}$ *For a given boolean formula φ and a subset $\Delta \subseteq \texttt{var}(\varphi)$, decide whether φ is Δ–decomposable.*

$\boxed{\Delta\texttt{DecPart}}$ *For a given boolean formula φ, a subset $\Delta \subseteq \texttt{var}(\varphi)$, and a partition $\{\Sigma_1, \Sigma_2\}$ of $\texttt{var}(\varphi) \setminus \Delta$, decide whether φ is Δ–decomposable with this partition.*

We also consider the variants of these problems for $\Delta = \varnothing$, which are denoted as $\varnothing\texttt{Dec}$ and $\varnothing\texttt{DecPart}$, respectively.

It turns out that the problem $\varnothing\texttt{Dec}$ for formulas in DNF is closely related to the problem of multilinear polynomial factorization ($\texttt{Dec}\mathbb{F}_2$) which we formulate below. The connection is in particular due to the fact that taking a conjunction of two formulas in DNF is quite similar to taking a product of two multivariate polynomials. We recall that a multivariate polynomial F is *linear* (*multilinear*) if the degree of each variable in F is 1. We denote a finite field of order 2 by $\mathbb{F}_2$ and say that a polynomial is *over the field* $\mathbb{F}_2$ if it has coefficients from $\mathbb{F}_2$. A polynomial F is called *factorable* over $\mathbb{F}_2$ if $F = G_1 \cdot G_2$, where G_1 and G_2 are non-constant polynomials over $\mathbb{F}_2$.

Example 2 *Consider the multilinear polynomial $F = xu + xv + yu + yv$ (cf. the positive DNF from Example 1). We have $F = (x + y) \cdot (u + v)$, thus F is factorable.*

The following important observation shows further connection between polynomial factorization and the problem $\varnothing\texttt{Dec}$:

Fact 2 (Factoring over $\mathbb{F}_2$) If a multilinear polynomial F is factorable over $\mathbb{F}_2$, then its factors do not have variables in common.

Clearly, if some factors G_1 and G_2 of F have a common variable then the polynomial $G_1 \cdot G_2$ is not linear and thus, is not equal to F in the ring of polynomials over $\mathbb{F}_2$.

$\boxed{\texttt{Dec}\mathbb{F}_2}$ *Given a non-constant multilinear polynomial F over $\mathbb{F}_2$, decide whether F is factorable over $\mathbb{F}_2$.*

Since every monomial can be viewed as a set of variables and the whole polynomial as a family of such sets, the problem $\texttt{Dec}\mathbb{F}_2$ can be reformulated as a variant of the well-known NP-complete Set Splitting Problem[3] :

$\boxed{\texttt{Cartesian Splitting Problem}}$ *Given a family $\mathcal{F}$ of subsets of a set S, decide whether there exists a partition $\{\Sigma_1, \Sigma_2\}$ of S such that $\mathcal{F} = \{S_1 \cup S_2 \mid S_i \in \mathcal{F}_i,\ i = 1, 2\}$, where each $\mathcal{F}_i$ is a family of sets over elements from Σ_i.*

It turns out that the above requirement of splitting into a cartesian union introduces enough structure into the Set Splitting Problem to obtain tractability.

[3]known also as the Hypergraph 2-Coloring Problem

3 Main Results

First, we formulate complexity results on decomposition of formulas given in the Conjunctive Normal Form and then proceed to formulas in DNF and ANF. Observe that decomposition itself is conceptually closer to the CNF representation, since it gives a conjunction of formulas. In particular, proving tractability of decomposition for positive CNF appears to be easier than for positive DNF. The situation with positive DNF and full DNF is more complicated, because decomposable formulas in DNF have a cartesian structure which can be recognized in polytime, but the proof of this fact relies on polynomial factorization over $\mathbb{F}_2$.

Theorem 1 (Complexity for CNF) *For boolean formulas given in CNF,*

1. *the problems* $\varnothing$`DecPart` *and* Δ`DecPart` *are coNP–complete.*
2. *the problem* $\varnothing$`Dec` *is coNP–hard and is in* P^{NP}*;*

Proof Sketch. We prove coNP-hardness in point 1 by showing that the set of formulas in CNF which are valid or unsatisfiable (denote it by Ω) is Karp-reducible to the set of $\varnothing$–decomposable formulas in CNF. For a given formula φ in CNF we consider the following formula constructed for φ, where $p, q \notin \mathtt{var}(\varphi)$ are "fresh" variables:

$$\psi = (\varphi \vee p) \wedge \bigwedge_{x \in \mathtt{var}(\varphi)} (q \vee x)$$

Clearly, ψ can be converted into CNF in linear time (in the size of φ). In the proof, we show the following equivalences: ψ is $\varnothing$–decomposable $\Leftrightarrow$ ψ is $\varnothing$–decomposable with the variable partition $\{\{p\}, \mathtt{var}(\varphi) \cup \{q\}\}$ $\Leftrightarrow$ $\varphi \in \Omega$. This shows coNP–hardness of the problems $\varnothing$`Dec`, $\varnothing$`DecPart` and hence, of Δ`DecPart` as well.

The containment of Δ`DecPart` in coNP is shown by a Karp-reduction to the set of valid boolean formulas. Let φ be a boolean formula, $\Delta \subseteq \mathtt{var}(\varphi)$ and $\{\Sigma_1, \Sigma_2\}$ be an arbitrary partition of $\mathtt{var}(\varphi) \setminus \Delta$. We consider the formulas $\varphi^*_{\Sigma_1}$ and $\varphi^*_{\Sigma_2}$ such that $\mathtt{var}(\varphi^*_{\Sigma_1}) \cap \mathtt{var}(\varphi^*_{\Sigma_2}) = \Delta$ and for $i = 1, 2$, each formula $\varphi^*_{\Sigma_i}$ is obtained from φ by renaming the variables from Σ_{3-i} into "fresh" ones, not present in φ. Then we show that φ is Δ–decomposable with the partition $\{\Sigma_1, \Sigma_2\}$ iff the formula $\varphi^*_{\Sigma_1} \wedge \varphi^*_{\Sigma_2} \to \varphi$ is valid.

For the proof of point 2 of the Theorem we provide a P^{NP}-algorithm for solving the problem $\varnothing$`Dec`. We show that if a formula φ in CNF is $\varnothing$–decomposable and contains a clause ξ with variables from both decomposition components, then ξ must contain a subclause $\xi' \subset \xi$ such that φ entails ξ'. This property gives an algorithm for deciding $\varnothing$`Dec`, which tries to "eliminate" literals one-by-one from clauses of a given formula in CNF (by quering an NP-oracle) and gives an equivalent formula, for which $\varnothing$–decomposability can be decided (and the corresponding components can be computed) in polytime. $\square$

We now formulate the result on tractability of Δ-decomposition for positive CNF. There are two key properties of positive formulas in CNF (and DNF) in proving tractability which can be informally described as follows. A positive formula does not have "too many equivalent reformulations" and entailment of positive formulas is tractable. This allows to easily compute a "canonical form" of a positive formula φ in CNF as a set of clauses, which uniquely defines decomposition components of φ (iff φ is Δ–decomposable).

Theorem 2 (Complexity for Positive CNF) *For positive boolean formulas in CNF, the problem* Δ`Dec` *is in* P*. Moreover,* Δ*-decomposition components can be computed in polynomial time (if decomposition exists).*

Let us proceed to results on decomposition of formulas given in DNF and ANF. We recall that the Algebraic Normal Form of a boolean formula (ANF) can be viewed as a multilinear polynomial over $\mathbb{F}_2$. Due to Fact 2, the notion of $\varnothing$–decomposability for formulas in ANF can be defined in terms of polynomial factorability over $\mathbb{F}_2$. For this reason, we use the terminology of polynomials when talking about algebraic results further in this section. We start with the complexity of decomposition for formulas in Full DNF (i.e. formulas given by the set of their satisfying assignments) and then formulate results on positive DNF and polynomial factorization over $\mathbb{F}_2$. Interestingly, the latter problem is related also to $\varnothing$–decomposition of formulas in Full DNF, even though such formulas contain negative literals. We show that negative literals can be encoded as "fresh" variables giving a positive DNF.

Theorem 3 (Complexity for Full DNF) *For boolean formulas in Full DNF,*

1. *the problem* $\Delta\texttt{DecPart}$ *is in P;*
2. *the problem* $\varnothing\texttt{Dec}$ *is reducible to* $\texttt{Dec}\mathbb{F}_2$ *and hence is in P.*

In each of the cases, the corresponding decomposition components can be computed in polynomial time.

Proof Sketch. The proof is based on the following important characterization:

Lemma 1 (Semantic Criterion of Decomposability) *Let φ be a boolean formula, V be the set of its satisfying assignments, and let $\Delta \subseteq \texttt{var}(\varphi)$ be a subset of variables. The formula φ is Δ–decomposable with a variable partition $\{\Sigma_1, \Sigma_2\}$ iff $V = V|_{\Sigma_1\cup\Delta} \uplus V|_{\Sigma_2\cup\Delta}$, where for $i = 1, 2$, $V|_{\Sigma_i\cup\Delta}$ is the restriction of V onto the variables from $\Sigma_i \cup \Delta$ and $V|_{\Sigma_1\cup\Delta} \uplus V|_{\Sigma_2\cup\Delta}$ is the set of all assignments v such that the restriction of v onto $\Sigma_i \cup \Delta$ belongs to $V|_{\Sigma_i\cup\Delta}$.*

To prove point 1 of Theorem 3, let φ be a formula in Full DNF and $\{\Sigma_1, \Sigma_2\}$ be a partition of $\texttt{var}(\varphi)$. Since Full DNF is the explicit representation of all satisfying assignments, one can compute the sets $V|_{\Sigma_i\cup\Delta}$, for $i = 1, 2$ and check whether the condition in Lemma 1 holds in polynomial time. If this is the case, then the the formulas $\psi_i = \bigvee_{\xi\in V|_{\Sigma_i\cup\Delta}} \xi$ in DNF, for $i = 1, 2$ are the required Δ–decomposition components (for brevity, in the definition of ψ_i we abuse the notion of satisfying assignment assuming that ξ is a conjunction of literals).

To prove point 2, for a given formula φ, we consider a positive formula φ' obtained by injective substitution of negative literals by "fresh" variables, not occurring in φ. Then we construct a multilinear polynomial F as a sum of monomials which correspond to conjuncts of φ' and show that φ is $\varnothing$-decomposable iff F is factorable over $\mathbb{F}_2$. $\square$

We show that for a positive formula φ in DNF without redundant conjuncts, $\varnothing$–decomposability is equivalent to factorability over $\mathbb{F}_2$ of the multilinear polynomial corresponding to φ. The polynomial is obtained as the sum of monomials (products of variables) corresponding to the conjuncts of φ. Observe that the positive formula $\varphi = x \vee (x \wedge y) \vee z$ with the redundant conjunct $x \wedge y$ is equivalent to $(x \vee z) \wedge \texttt{taut}(\{y\})$ and thus, $\varnothing$–decomposable. However, the polynomial $x + xy + z$ corresponding to φ is non-factorable. Also note that if a polynomial has a factor with the constant monomial, e.g. $xy + y = (x + 1) \cdot y$, then the corresponding boolean formula in DNF contains a redundant conjunct.

Theorem 4 (Decomposition of Positive DNF and Factorization) *For positive boolean formulas in DNF without redundant conjuncts, the problem* $\varnothing\texttt{Dec}$ *is equivalent to* $\texttt{Dec}\mathbb{F}_2$*.*

We formulate the main result on formulas given in DNF in the following corollary which is a consequence of Theorems 4, 5, and the constructions mentioned in the proof sketch to Theorem 1.

Corollary 1 (Complexity for DNF)

1. *For formulas in DNF, the problems* $\varnothing$`DecPart` *and* Δ`DecPart` *are coNP-complete;*
2. *for positive boolean formulas in DNF, the problem* $\varnothing$`Dec` *is in P and the corresponding decomposition components can be computed in polynomial time.*

Note that the formula ψ given in the proof sketch to point 1 of Theorem 1 can be converted into DNF in polynomial time, if the input formula φ is given in DNF, which shows coNP-hardness for the first point of the Corollary. The containment in coNP can be shown by reduction to the validity of boolean formulas as in the proof sketch to Theorem 1, since the construction there does not depend on the normal form in which the formula φ is given. The second point of the Corollary follows from Theorems 4 and 5, and the fact that redundant conjuncts can be found and eliminated efficiently from an input formula.

We now turn to tractability of the problem `Dec`$\mathbb{F}_2$, to which the decomposition problems in Theorem 3 and Corollary 1 are reduced. Originally, tractability of `Dec`$\mathbb{F}_2$ is a consequence of the results from [22], where the authors provide two solutions to polynomial decomposition over an arbitrary finite field F. The first one is a decomposition algorithm, which has a subroutine for computing a justification assignment for an input polynomial, and relies on a procedure for identity testing in F. It is proved that the complexity of this algorithm is $O(n^3 \cdot d \cdot IT)$, where n is the number of variables, d is the maximal individual degree of variables in the input polynomial, and IT is the complexity of identity testing in F. It follows that for multilinear polynomials over the field $\mathbb{F}_2$ this gives a factorization algorithm of quartic complexity (assuming that polynomials are given as symbol sequences, as noted in Section 2.2). The second solution proposed by the authors is a decomposition algorithm which constructs for every variable of an input polynomial f, a combination $f \cdot f_1 - f_2 \cdot f_3$ of four polynomials, where each f_i is a "copy" of f under a renaming of some variables. Every combination is tested for equality to the zero polynomial. It can be seen that this gives an algorithm of cubic complexity for factoring multilinear polynomials over $\mathbb{F}_2$.

In Theorem 5 below, we provide a solution to factorization of multilinear polynomials over $\mathbb{F}_2$, which is different from the both algorithms proposed in [22]. The only common feature between the approaches is the application of identity testing, which seems to be inevitable in factorization. Our solution is based on computation of partial derivatives of a polynomial obtained from the input one and gives an algorithm of cubic complexity. More precisely, a product $f_1 \cdot f_2$ is computed once, where f_i are multilinear polynomials obtained from the input, and then for each variable y, the partial derivative of $f_1 \cdot f_2$ by y is tested for equality to zero. In particular, our algorithm operates polynomials which are smaller than the ones considered in [22]. Moreover, we note that the algorithm can be implemented without computing the product $f_1 \cdot f_2$ explicitly, which is particularly important for dealing with large inputs. We present the factorization algorithm as the theorem below to follow the complexity oriented style of exposition used in this paper.

Theorem 5 (Tractability of Linear Polynomial Factorization over $\mathbb{F}_2$**)** *The problem* `Dec`$\mathbb{F}_2$ *is in P and for any factorable multilinear polynomial, its factors can be computed in polynomial time.*

Proof. Let F be a non-constant multilinear polynomial over $\mathbb{F}_2$. We will describe a number of important properties which hold if F is factorable over $\mathbb{F}_2$. Based on these properties, we will derive a polynomial procedure for partitioning the variables of F into disjoints sets Σ_1 and Σ_2 such that if F

is factorable, then it must have factors which are polynomials having these sets of variables. Having obtained Σ_1 and Σ_2, it suffices to check whether F is indeed factorable wrt this partition: if the answer is "no", then F is non-factorable, otherwise we obtain the corresponding factors. Checking whether F is factorable wrt a variable partition can be done efficiently due the following fact:

Lemma 2 (Factorization Under a Given Variable Partition) *In the notations above, for $i = 1, 2$, let S_i be the set of monomials obtained by restricting every monomial of F onto Σ_i (for instance, if $F = xy+y$ and $\Sigma_1 = \{x\}$, then $S_1 = \{x, 1\}$). Let F_i be the polynomial consisting of the monomials of S_i for $i = 1, 2$. Then F is factorable into some polynomials with the sets of variables Σ_1 and Σ_2 iff $F = F_1 \cdot F_2$.*

Proof of the lemma. The "if" direction is obvious, since for $i = 1, 2$, each F_i necessarily contains all the variables from Σ_i. Now assume that F has a factorization $F = G_1 \cdot G_2$ which corresponds to the partition Σ_1, Σ_2. Then every monomial of F is a product of some monomials from G_1, G_2, i.e. it either contains variables of both Σ_1 and Σ_2, or only from Σ_i for some $i = 1, 2$ iff G_{3-i} contains the constant monomial. This means that S_i is the set of monomials of G_i for $i = 1, 2$, i.e. $F_i = G_i$. □

Let us proceed to properties of factorable polynomials. Let $F_{x=v}$ be the polynomial obtained from F by setting x equal to v. Note that $\frac{\partial F}{\partial x} = F_{x=1} + F_{x=0}$.

First of all, note that if some variable x is contained in every monomial of F, then F is either non-factorable (in case $F = x$), or trivially factorable, i.e. $F = x \cdot \frac{\partial F}{\partial x}$. We further assume that there is no such variable in F. We also assume that $F \neq x + 1$, i.e. F contains at least two variables[4].

Let F be a polynomial over the set of variables $\{x, x_1, \ldots, x_n\}$. If F is factorable, then it can be represented as

$$F = (x \cdot Q + R) \cdot H, \quad \text{where}$$

- the polynomials Q,R, and H do not contain x;
- Q and R do not have variables with H in common;
- R is a non-empty polynomial (since F is not trivially factorable);
- the left-hand side of this product is a non-factorable polynomial.

Then we have $F_{x=0} = R \cdot H$ and also $\frac{\partial F}{\partial x} = Q \cdot H$. Obviously, the both polynomials can be computed in polynomial time[5]. Let y be a variable of F different from x and consider the following derivative of the product of these polynomials:

$$\frac{\partial}{\partial y}(Q \cdot R \cdot H^2) = \frac{\partial Q}{\partial y} RH^2 + Q\frac{\partial}{\partial y}(RH^2) = \frac{\partial Q}{\partial y} RH^2 + \frac{\partial R}{\partial y} QH^2 + 2\frac{\partial H}{\partial y} QRH.$$

Since in $\mathbb{F}_2$ for all z it holds that $2z = z + z = 0$, we have:

$$\frac{\partial}{\partial y}(Q \cdot R \cdot H^2) = H^2 \cdot \left(\frac{\partial Q}{\partial y} R + \frac{\partial R}{\partial y} Q\right) = H^2 \cdot \frac{\partial}{\partial y}\left(Q \cdot R\right).$$

It follows that in case y is a variable from H, we have $\frac{\partial}{\partial y}\left(Q \cdot R\right) = 0$ and thus, $\frac{\partial}{\partial y}(Q \cdot R \cdot H^2) = 0$. Let us now show the opposite, assume that the variable y does not belong to H and prove that the derivative is not equal to zero.

[4]We note that besides the factors of the form x and $x + 1$, there is a number of other simple cases of factorization that can be recognized easily.

[5]Note that since $(x \cdot Q + R)$ is non-factorable, the GCD of $R \cdot H$ and $Q \cdot H$ is exactly H, so the factorization problem is solved given there exists a polytime algorithm for GCD computation. In fact, polytime factorization yields the existence of such algorithm.

Since y does not belong to H, in general, Q and R have the form

$$Q = Ay + B, \qquad R = Cy + D,$$

for some polynomials A, B, C, D not containing y. Then $Q \cdot R = ACy^2 + (AD + BC)y + BD$ and hence, $\frac{\partial}{\partial y}(Q \cdot R) = AD + BC$.

Thus, we need to show that $AD + BC \neq 0$. Assume the contrapositive, i.e. that $AD + BC = 0$. Note that AD and BC can not be zero, because otherwise at least one of the following holds: $A = B = 0$, $A = C = 0$, $D = B = 0$, or $D = C = 0$. The first two conditions are clearly not the case, since we have assumed that x and y are not contained in H, while the latter conditions yield that F is trivially factorable (wrt the variable y or x, respectively). From this we obtain that $AD + BC = 0$ holds iff $AD = BC$ (since we are in $\mathbb{F}_2$).

Let $B = f_1 \cdot \ldots \cdot f_m$ and $C = g_1 \cdot \ldots \cdot g_n$ be the (unique) factorizations of B and C into non-factorable polynomials. We have $AD = f_1 \cdot \ldots \cdot f_m \cdot g_1 \cdot \ldots \cdot g_n$, thus this may assume that $A = f_1 \cdot \ldots \cdot f_k \cdot g_1 \cdot \ldots \cdot g_l$ for some $0 \leqslant k \leqslant m$ and $0 \leqslant l \leqslant n$ (when $k = l = 0$, we assume that $A = 1$). The polynomials B, C, D can be represented in the same form. Let us denote for some polynomials U, V by (U, V) the greatest common divisor of U and V. Then $A = (A, B) \cdot (A, C)$, $B = (A, B) \cdot (D, B)$, similarly for C and D, and we obtain

$$x \cdot Q + R = x \cdot (Ay + B) + (Cy + D) =$$

$$= x \cdot ((A, B)(A, C)y + (A, B)(D, B)) + ((A, C)(D, C)y + (D, B)(D, C)) =$$

$$= ((A, B)x + (D, C))((A, C)y + (D, B)),$$

which is a contradiction, because we have assumed that $x \cdot Q + R$ is non-factorable.

We have obtained a procedure for partitioning the variables of F into disjoint sets Σ_1 and Σ_2 in the following way. Having chosen some initial variable x from F, we first assign $\Sigma_1 = \{x\}$, $\Sigma_2 = \varnothing$ and compute the polynomial $Q \cdot R \cdot H^2$ (which is $\frac{\partial F}{\partial x} \cdot F_{x=0}$). Then for every variable y from F (distinct from x), we compute the derivative $\frac{\partial}{\partial y}(Q \cdot R \cdot H^2)$. If it equals zero, we put y into Σ_2, otherwise we put y into Σ_1. If at the end we have $\Sigma_2 = \varnothing$, then the polynomial F is non-factorable. Otherwise it remains to apply Lemma 2 to verify whether the obtained sets Σ_1 and Σ_2 indeed correspond to a factorization of F. If the answer is "no", then F is non-factorable, otherwise the polynomials F_1 and F_2 defined in Lemma 2 are the required factors.

If n is the size of the input polynomial as a symbol sequence, then it takes $O(n^2)$ steps to compute the polynomial $G = Q \cdot R \cdot H^2$ and test whether the derivative $\frac{\partial G}{\partial y}$ equals zero for a variable y (since both, computing the derivative and identity testing can be done in linear time for polynomials over $\mathbb{F}_2$, given as symbol sequences). As we must verify this for every variable $y \neq x$, we have a procedure that computes a candidate variable partition in $O(n^3)$ steps. Then it takes $O(n^2)$ time to verify by Lemma 2 whether this partition indeed corresponds to factors of F. □

The procedure described above gives at least two ways of computing a candidate variable partition of the input polynomial F. Assuming we have computed the polynomials $A = \frac{\partial F}{\partial x}$ and $B = F_{x=0}$, the first way is to compute the product $A \cdot B$ once, thus obtaining a quadratically large polynomial, and then use it to test identities of derivatives for each variable y (different from x). The other way is to compute the polynomials $D = \frac{\partial}{\partial y}(F_{x=0})$ and $C = \frac{\partial}{\partial y}(\frac{\partial F}{\partial x})$ for each variable y (different from x) and test whether $AD + BC = 0$. On inputs obtained from large positive DNFs, it makes little sense to compute products of polynomials of size "almost" as large as the input (in the worst case). We show however that testing $AD + BC = 0$ can be done without computing the products AD and BC. W.l.o.g. we assume that neither of these polynomials is zero, because otherwise we must have $D = 0$

and either $B = 0$ or $C = 0$, which can be easily verified. Then $AD + BC = 0$ iff $AD = BC$. We may also assume that no variable is present in every monomial of A, B, C, or D and hence, there is no variable z such that the polynomial z divides AD or BC. Now let z be any variable different from x and consider the mentioned polynomials given as $A = A_1z + A_2$, $D = D_1z + D_2$, $B = B_1z + B_2$, $C = C_1z + C_2$ wrt the variable z. Then we have $AD = BC$ iff

$$(A_1z + A_2)(D_1z + D_2) = (B_1z + B_2)(C_1z + C_2),$$

$$A_1D_1z^2 + (A_1D_2 + A_2D_1)z + A_2D_2 = B_1C_1z^2 + (B_1C_2 + B_2C_1)z + B_2C_2.$$

and the equality holds iff the corresponding coefficients are equal:

$$A_1D_1 = B_1C_1 \quad (1) \qquad A_2D_2 = B_2C_2 \quad (2) \qquad A_1D_2 + A_2D_1 = B_1C_2 + B_2C_1 \quad (3)$$

If at least one of the identities (1), (2) does not hold, then we have $AD \neq BC$. Otherwise, we can use these identities to verify (3) in the following way. We may assume that the multiplier and the residual of A, B, C or D wrt z are both not equal to zero; let it be the case for A, i.e. $A_1, A_2 \neq 0$. Multiplying the both sides of (3) by A_1A_2 gives

$$A_1^2A_2D_2 + A_1A_2^2D_1 = A_1A_2B_1C_2 + A_1A_2B_2C_1 \quad \text{and next, by using the identities (1) and (2),}$$

$$A_1^2B_2C_2 + A_1A_2B_2C_1 = A_2^2B_1C_1 + A_1A_2B_1C_2 \implies A_1B_2(A_1C_2 + A_2C_1) = A_2B_1(A_2C_1 + A_1C_2).$$

Hence, it suffices to check $(A_1B_2 + A_2B_1)(A_1C_2 + A_2C_1) = 0$, i.e. at least one of these factors equals zero. It turns out that we need to test at most 4 polynomial identities and each of them are smaller than the original identity $AD = BC$. Then we proceed recursively until trivial polynomials are obtained.

4 Discussion

We have proved that AND–decomposability is intractable in general for boolean formulas given in CNF or DNF. On the other hand, we have shown the existence of polytime algorithms for computing decomposition components of positive formulas in CNF and DNF, and formulas given in Full DNF and the Algebraic Normal Form. We believe that the tractability result on positive DNF can contribute to improving efficiency of existing model counting techniques, while the result on Full DNF can be applied in optimization of boolean functions given by lookup tables. Since AND–decomposability and OR-decomposability are the dual notions, our results are applicable to this notion as well. For instance, the quality of multilevel decomposition (i.e. alternating AND/OR–decomposition) of monotone functions given in DNF strongly depends on the kind of decomposition used at the topmost level. As a rule, OR–decomposition has a priority over AND-decomposition, since it is simple[6] for positive DNF. However, choosing AND-decomposition at the topmost level may result in smaller factors. For example, let us consider the following monotone boolean function given as positive DNF (we use the addition/multiplication notation for disjunctions/conjunctions):

$$F = absu + absv + absw + abtu + abtv + abtw + abxy + abxz + acsu + acsv + acsw + actu + \\ actv + actw + acxy + acxz + desu + desv + desw + detu + detv + detw + dexy + dexz$$

Applying "AND–first" decomposes F into the following factors: $F = (ab + ac + de)(su + sv + sw + tu + tv + tw + xy + xz)$. OR–decomposition of the second factor (the first factor OR-decomposes syntactically too) gives the components: $su + sv + sw + tu + tv + tw$ and $xy + xz$.

[6]for the same reason as AND-decomposition is simple for positive/negative CNF, cf. Theorem 2

Next, AND-decomposition is applied to each of the obtained factors and finally, we obtain

$$F = (a(b+c)+de)((s+t)(u+v+w)+x(y+z)),$$

which is a read-once formula of depth 4 having 13 occurrences of variables.
ESPRESSO[7] which implements "OR–first" decomposition at the topmost level gives a longer expression:

$$F = x(a(c+b)+de)(z+y)+(a(c+b)+de)(t+s)(w+v+u)$$

which is a formula of depth 5 having 18 occurrences of variables and this formula is not read-once.
Disjoint AND-decomposition can be viewed as a "constructive" way to recognize read–once formulas. By "constructive" we mean that it allows not only to recognize that a formula is read–once (see Golumbic et al. [8]), but also to obtain its read–once form efficiently.

The result on tractability of AND-decomposition for full DNF given in Theorem 3, can be applied to the class of formulas which can be called pre-full DNF. This is the class of formulas in DNF whose conversion into the full DNF causes only polynomial increase in the size of the formula. One can consider the following simple conversion process: if some conjunct of DNF does not include a variable x, then this conjunct is duplicated with the literals x and $\bar{x}$ respectively. Then it is not hard to see that the class of pre-full DNF contains formulas such that their "deficit" (i.e. the maximum of absent variables over all conjunctions) is $O(\log n)$, where n the is number of formula variables.
For example, given the formula $a\bar{b}\bar{x}y + a\bar{b}\bar{z} + c\bar{x}y + c\bar{z}$, the corresponding full DNF is obtained as:

$$\begin{gathered}
abcxy\bar{z} + abcx\bar{y}\bar{z} + abc\bar{x}yz + abc\bar{x}y\bar{z} + abc\bar{x}\bar{y}\bar{z} + a\bar{b}cxy\bar{z} + a\bar{b}cx\bar{y}\bar{z} + a\bar{b}c\bar{x}yz + a\bar{b}c\bar{x}y\bar{z} + \\
a\bar{b}c\bar{x}\bar{y}\bar{z} + a\bar{b}\bar{c}xy\bar{z} + a\bar{b}\bar{c}x\bar{y}\bar{z} + a\bar{b}\bar{c}\bar{x}yz + a\bar{b}\bar{c}\bar{x}y\bar{z} + a\bar{b}\bar{c}\bar{x}\bar{y}\bar{z} + \bar{a}bcxy\bar{z} + \bar{a}bcx\bar{y}\bar{z} + \bar{a}bc\bar{x}yz + \\
\bar{a}bc\bar{x}y\bar{z} + \bar{a}bc\bar{x}\bar{y}\bar{z} + \bar{a}\bar{b}cxy\bar{z} + \bar{a}\bar{b}cx\bar{y}\bar{z} + \bar{a}\bar{b}c\bar{x}yz + \bar{a}\bar{b}c\bar{x}y\bar{z} + \bar{a}\bar{b}c\bar{x}\bar{y}\bar{z}.
\end{gathered}$$

Then, by using the idea from the proof of Theorem 3, computing AND-decomposition of this formula by polynomial factorization gives the components $(xy\bar{z}+x\bar{y}\bar{z}+\bar{x}yz+\bar{x}y\bar{z}+\bar{x}\bar{y}\bar{z})(abc+a\bar{b}c+a\bar{b}\bar{c}+\bar{a}bc+\bar{a}\bar{b}c)$, which are again formulas in full DNF. Finally, by using the standard minimization techniques for each of these factors independently, one obtains the representation of the initial formula as: $(\bar{z}+\bar{x}y)(c+a\bar{b})$.

Finally, it is important to note that a decomposable formula uniquely defines the finest partition of its variables, thus in practice the only decomposition parameter to be specified is the set of shared variables between the components. The polytime factorization algorithm for multilinear polynomials over $\mathbb{F}_2$ provides a solution for the case when this set is empty, i.e. when decomposition is disjoint. It is an open question whether a similar result could be applied for obtaining non-disjoint decompositions. Further research questions include implementation of the proposed polytime decomposition algorithms and their evaluation on industrial benchmarks for boolean circuits.

Acknowledgements

The first author was supported by the Russian Foundation for Humanities, grant No. 13-01-12003B. The second author was supported by the German Research Foundation within the Transregional Collaborative Research Center SFB/TRR 62 "Companion-Technology for Cognitive Technical Systems".

[7]a well–known heuristic optimizer used as a reference tool for optimization of boolean functions

References

[1] Tomas Bengtsson, Andres Martinelli, and Elena Dubrova, *A fast heuristic algorithm for disjoint decomposition of Boolean functions*, in Notes of the 11th IEEE/ACM International Workshop on Logic & Synthesis (IWLS'02), 2002, pp. 51–55.

[2] Jan C. Bioch, *The complexity of modular decomposition of Boolean functions*, Discrete Applied Mathematics, 149 (2005), pp. 1–13.

[3] ——, *Decomposition of Boolean functions*, in Boolean Models and Methods in Mathematics, Computer Science, and Engineering, Yves Crama and Peter. L. Hammer, eds., vol. 134 of Encyclopedia of Mathematics and its Applications, Cambridge University Press, New York, NY, USA, 2010, pp. 39–78.

[4] Randal E. Bryant, *Graph-based algorithms for Boolean function manipulation*, IEEE Transactions on Computers, 35 (1986), pp. 677–691.

[5] Huan Chen, Mikoláš Janota, and João Marques-Silva, *QBF-based Boolean function bi-decomposition*, in Proceedings of the Design, Automation & Test in Europe Conference (DATE'12), IEEE, 2012, pp. 816–819.

[6] Mihir Choudhury and Kartik Mohanram, *Bi-decomposition of large Boolean functions using Blocking Edge Graphs*, in Proceedings of the 2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'10), Piscataway, New Jersy, USA, 2010, IEEE Press, pp. 586–591.

[7] Adnan Darwiche and Pierre Marquis, *A knowledge compilation map*, Journal of Artificial Intelligence Research, 17 (2002), pp. 229–264.

[8] Martin C. Golumbic, Aviad Mintz, and Udi Rotics, *An improvement on the complexity of factoring read-once Boolean functions*, Discrete Applied Mathematics, 156 (2008), pp. 1633–1636.

[9] Sunil P. Khatri and Kanupriya Gulati, eds., *Advanced Techniques in Logic Synthesis, Optimizations and Applications*, Springer, New York Dordrecht Heidelberg London, 2011.

[10] Boris Konev, Carsten Lutz, Denis Ponomaryov, and Frank Wolter, *Decomposing description logic ontologies*, in Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning (KR'10), Palo Alto, California, USA, 2010, AAAI Press.

[11] Ian Kuon, Russell Tessier, and Jonathan Rose, *FPGA Architecture: Survey and Challenges*, Now Publishers Inc, Boston - Delft, 2008.

[12] Alan Mishchenko and Robert Brayton, *Faster logic manipulation for large designs*, in Proceedings of the 22nd International Workshop on Logic and Synthesis, 2013.

[13] Alan Mishchenko and Tsutomu Sasao, *Large-scale SOP minimization using decomposition and functional properties*, in Proceedings of the 40th ACM/IEEE Design Automation Conference (DAC'03), New York, NY, USA, 2003, ACM, pp. 149–154.

[14] Alan Mishchenko, Bernd Steinbach, and Marek A. Perkowski, *An algorithm for bi-decomposition of logic functions*, in Proceedings of the 38th ACM/IEEE Design Automation Conference (DAC'01), New York, NY, USA, 2001, ACM, pp. 103–108.

[15] Andrey Morozov and Denis Ponomaryov, *On decidability of the decomposability problem for finite theories*, Siberian Mathematical Journal, 51 (2010), pp. 667–674.

[16] Marek A. Perkowski and Stanislaw Grygiel, *A survey of literature on function decomposition, Version IV*, PSU Electrical Engineering Department Report, Department of Electrical Engineering, Portland State University, Portland, Oregon, USA, November 1995.

[17] Denis Ponomaryov, *On decomposability in logical calculi*, Bulletin of the Novosibirsk Computing Center, 28 (2008), pp. 111–120, available at http://persons.iis.nsk.su/files/persons/pages/delta–decomp.pdf.

[18] Denis Ponomaryov, *The algorithmic complexity of decomposability in fragments of first-order logic*, Research Note. Abstract appears in Proc. Logic Colloquium' 14. Available at http://persons.iis.nsk.su/files/persons/pages/sigdecomp.pdf, 2014.

[19] Tsutomu Sasao and Jon T. Butler, *On bi-decompositions of logic functions*, in Notes of the 1997 IEEE/ACM International Workshop on Logic & Synthesis (IWLS'97), 1997, pp. 1–6.

[20] ——, *On the minimization of SOPs for bi-decomposition functions*, in Proceedings of the 2001 Asia and South Pacific Design Automation Conference (ASP-DAC'01), New York, NY, USA, 2001, ACM, pp. 219–224.

[21] Igor E. Shparlinski, *Computational and Algorithmic Problems in Finite Fields*, Springer, New York Dordrecht Heidelberg London, 1992.

[22] Amir Shpilka and Ilya Volkovich, *On the relation between polynomial identity testing and finding variable disjoint factors*, in Proceedings of the 37^{th} International Colloquium on Automata, Languages and Programming. Part 1 (ICALP 2010), vol. 6198 of Lecture Notes in Computer Science, Springer, 2010, pp. 408–419.

[23] Bernd Steinbach and Christian Lang, *Exploiting functional properties of Boolean functions for optimal multi-level design by bi-decomposition*, Artificial Intelligence Review, 20 (2003), pp. 319–360.

[24] Joachim von zur Gathen and Jürgen Gerhard, *Modern Computer Algebra*, Cambridge University Press, New York, NY, USA, Third ed., 2013.

Appendix (Missing Proofs for Section 3)

Additional Conventions and Notations in Proofs

Throughout the text, we assume that conjuncts and clauses of boolean formulas do not contain duplicate occurrences of literals and that formulas do not contain duplicate occurrences of clauses or conjuncts. If φ is a formula in CNF (respectively, DNF), then the notation $\xi \in \varphi$ means that ξ is a clause (respectively, conjunct) of φ and $\varphi \setminus \{\xi\}$ denotes the formula given by the remaining clauses (conjuncts). A variable $x \in \mathtt{var}(\varphi)$ is called *inessential* in φ if φ is equivalent to some formula φ' such that $x \notin \mathtt{var}(\varphi')$. Note that for any subset $\Delta \subseteq \mathtt{var}(\varphi)$, if a variable $x \in \mathtt{var}(\varphi) \setminus \Delta$ is inessential in φ, then φ is Δ–decomposable (into the components $\mathtt{taut}(x)$ and φ').

Let Σ be a set of variables. An *assignment to* Σ is some assignment v to the variables from Σ. A satisfying assignment of a boolean formula φ (or *assignment of* φ, for short) is an assignment to the variables of φ which makes φ true. We will slightly abuse our terminology and call *assignment of* φ any conjunction ξ of literals over the variables from $\mathtt{var}(\varphi)$ such that the assignment to these variables, which corresponds to their negative/positive occurrences in ξ, makes φ true. A *partial assignment* of φ is an assignment to some subset of $\mathtt{var}(\varphi)$. An *expansion* of a partial assignment v is any assignment to $\mathtt{var}(\varphi)$ which agrees with v. If $\sigma \subseteq \mathtt{var}(\varphi)$ is a subset, then $\wp(\sigma)$ denotes the set of conjuncts which represents all the possible assignments to σ.

Definition 2 (Set of Formulas Uniquely Defines Partition) *Let Φ be a set of boolean formulas over variables Σ and let $\Delta \subseteq \Sigma$ be a subset. Denote $\Phi_\Delta = \{\varphi \in \Phi \mid \mathtt{var}(\varphi) \subseteq \Delta\}$ and consider the bipartite graph $\Gamma^\Phi_\Delta = (\Sigma \setminus \Delta,\ \Phi \setminus \Phi_\Delta,\ E)$, where for $x \in \Sigma \setminus \Delta$ and $\varphi \in \Phi \setminus \Phi_\Delta$, we have $\langle x, \varphi \rangle \in E$ iff $x \in \mathtt{var}(\varphi)$.*

We say that Φ uniquely defines the partition of Σ wrt Δ if for any $x, y \in \Sigma \setminus \Delta$ the following are equivalent:

- *x and y belong to the same connected component of Γ^Φ_Δ;*
- *$\bigwedge_{\varphi\in\Phi} \varphi$ is not Δ–decomposable into some components ψ_1 and ψ_2, with $x \in \mathtt{var}(\psi_1)$ and $y \in \mathtt{var}(\psi_2)$.*

Example 3 *Consider the set of formulas $\Phi = \{x,\ x \vee y\}$ and let $\Delta = \varnothing$. Both x and y belong to the same connected component of the graph Γ^Φ_Δ from Definition 2. The set Φ does not uniquely define the partition of $\Sigma = \{x, y\}$ wrt Δ, since $x \wedge (x \vee y)$ is $\varnothing$–decomposable into the components x and $\mathtt{taut}(\{y\})$. Note that the set of formulas $\{x,\ \mathtt{taut}(\{y\})\}$ is equivalent to Φ and defines the partition of Σ uniquely (in fact, wrt any Δ).*

The following fact is a simple consequence of Definition 2.

Remark 1 *There exists a polynomial time algorithm which, given a set Φ of boolean formulas over variables Σ and a subset $\Delta \subseteq \Sigma$, computes the graph Γ^Φ_Δ from Definition 2 and outputs "yes" iff Γ^Φ_Δ is not connected.*

Every proof of an upper complexity bound of Δ–decomposition will be based on a procedure for computing a "canonical representation" of a given formula φ as a conjunction $\bigwedge_{i=1,\dots,n} \psi_i$ such that the set $\{\psi_i\}_{i=1,\dots,n}$ uniquely defines the partition of $\mathtt{var}(\varphi)$ wrt Δ. Then, to verify whether φ is Δ–decomposable (and to obtain the corresponding variable partition and decomposition components), it suffices to compute the graph Γ over the set of formulas $\{\psi_i\}_{i=1,\dots,n}$ by using Remark 1 and verify whether it is connected.

Missing Proofs

Theorem 1 (Complexity for CNF) *For boolean formulas given in CNF,*

1. *the problems* $\varnothing$`DecPart` *and* Δ`DecPart` *are coNP–complete.*
2. *the problem* $\varnothing$`Dec` *is coNP–hard and is in* P^{NP}*;*

Proof. Let us note the following property of decomposable formulas in CNF, which will be the key to proving the complexity bounds.

Lemma 3 (Property of Decomposable Formulas in CNF) *Let φ be a formula, $\Delta \subseteq \mathtt{var}\,(\varphi)$ be a subset, and let φ be equivalent to the conjunction of formulas ψ_1 and ψ_2 such that $\mathtt{var}\,(\psi_1) \cap \mathtt{var}\,(\psi_2) = \Delta$. Let φ have the form $\chi \wedge (\xi_1 \vee \xi_2)$, where $\xi_1 \vee \xi_2$ is a clause such that $\mathtt{var}\,(\xi_i) \subseteq \mathtt{var}\,(\psi_i)$ for $i = 1, 2$. Then $\varphi \vdash (\xi_1 \vee \theta_\Delta) \;\wedge\; (\neg\theta_\Delta \vee \xi_2)$ for some formula θ_Δ over variables Δ.*

Proof of the lemma. We have $\psi_1 \wedge \psi_2 \vdash \xi_1 \vee \xi_2$, hence $\psi_1 \wedge \neg\xi_1 \vdash \psi_2 \to \xi_2$ and, by the conditions of the lemma, we have $\mathtt{var}\,(\psi_1 \wedge \neg\xi_1) \cap \mathtt{var}\,(\psi_2 \to \xi_2) = \Delta$. By interpolation, there exists a formula θ_Δ over variables Δ such that $\psi_1 \wedge \neg\xi_1 \vdash \theta_\Delta$ and $\theta_\Delta \vdash \psi_2 \to \xi_2$ hold. Therefore we have $\psi_1 \vdash \xi_1 \vee \theta_\Delta$, $\psi_2 \vdash \neg\theta_\Delta \vee \xi_2$ and hence, $\varphi \vdash (\xi_1 \vee \theta_\Delta) \;\wedge\; (\neg\theta_\Delta \vee \xi_2)$. $\square$

Corollary 2 *In the conditions of the lemma, if $\Delta = \varnothing$, then $\varphi \vdash \xi_i$ for some $i = 1, 2$.*

Let us prove coNP-hardness in point 1 of the theorem by showing that the set of formulas in CNF which are valid or unsatisfiable (denote it by Ω) is Karp-reducible to the set of $\varnothing$–decomposable formulas in CNF. We recall that the set Ω is coNP-complete, since the set of valid boolean formulas in CNF is in P. Let φ be a formula in CNF. Consider the following formula

$$\psi = (\varphi \vee p) \wedge \bigwedge_{x \in \mathtt{var}\,(\varphi)} (q \vee x)$$

constructed for φ, where $p, q \notin \mathtt{var}\,(\varphi)$ are "fresh" variables. Clearly, ψ can be converted into CNF in linear time (in the size of φ). We claim that ψ is $\varnothing$–decomposable iff $\varphi \in \Omega$.

The "if" direction is simple: if φ is valid, then ψ is equivalent to $(p \vee \neg p) \wedge \bigwedge_{x\in\mathtt{var}\,(\varphi)}(q \vee x)$ and if φ is unsatisfiable, then ψ is equivalent to $p \wedge \bigwedge_{x\in\mathtt{var}\,(\varphi)}(q \vee x)$. For the "only if" direction, suppose that $\varphi \notin \Omega$, but ψ is $\varnothing$–decomposable.

Note that the variable q must then belong to the same decomposition component of ψ together with every variable $x \in \mathtt{var}\,(\varphi)$. Indeed, for every such x, $(q \vee x)$ is a clause in ψ, hence by Corollary 2, we would otherwise have $\psi \vdash q$ or $\psi \vdash x$. Neither condition holds, because there exists an assignment of ψ in which q is false and there is also an assignment of ψ in which x is false. Therefore, all the variables of φ together with q must belong to exactly one $\varnothing$–decomposition component of ψ. Now let us consider a clause $\xi \vee p$ from the CNF representation of $\varphi \vee p$, where ξ is a clause from φ. Let us demonstrate that the variable p must be in the same decomposition component together with at least one variable $x \in \mathtt{var}\,(\varphi)$. If it is not the case, then by Corollary 2, we would have $\psi \vdash p$ or

$\psi \vdash \xi$ (actually, for every clause ξ from φ). The first condition obviously does not hold, since we have assumed that φ is satisfiable and hence, there is an assignment of ψ in which p is false. The second condition yields $\varphi \vee p \vdash \xi$, because any assignment to $\mathtt{var}\,(\varphi) \cup \{p\}$ satisfying $\varphi \vee p$ can be expanded to an assignment of ψ (by setting q equal to true). Since $p \notin \mathtt{var}\,(\xi)$, the condition $\varphi \vee p \vdash \xi$ can hold only if every clause ξ of φ is tautological, i.e. if φ is a tautology, which is not the case by our assumption. Thus, we have arrived at contradiction with $\varnothing$–decomposability of ψ, having shown that all its variables must be in exactly one decomposition component.

The above mentioned reduction shows coNP–hardness of $\varnothing$`DecPart` and hence, of Δ`DecPart` as well: it suffices to note that ψ is $\varnothing$–decomposable iff it is $\varnothing$–decomposable with the variable partition $\{\{p\}, \mathtt{var}\,(\varphi) \cup \{q\}\}$.

Let us now prove that $\varnothing$`DecPart` and Δ`DecPart` are in coNP by showing a Karp-reduction to the set of valid boolean formulas.

We consider separately the case $\Delta = \varnothing$. For a given formula φ and a partition $\{\Sigma_1, \Sigma_2\}$ of $\mathtt{var}\,(\varphi)$, let us consider the formula χ defined as the conjunction of the formulas from the set

$$\{(\varphi \to \xi_1) \vee (\varphi \to \xi_2) \mid \xi_1 \vee \xi_2 \in \varphi \text{ and } \mathtt{var}\,(\xi_i) \subseteq \Sigma_i \text{ for } i = 1, 2\}.$$

Then it is not hard to verify by Corollary 2 that χ is valid iff φ is $\varnothing$–decomposable.

Now let $\Delta \subseteq \mathtt{var}\,(\varphi)$ and $\{\Sigma_1, \Sigma_2\}$ be an arbitrary partition of $\mathtt{var}\,(\varphi) \setminus \Delta$. Consider the formulas $\varphi^*_{\Sigma_1}$ and $\varphi^*_{\Sigma_2}$ such that $\mathtt{var}\,(\varphi^*_{\Sigma_1}) \cap \mathtt{var}\,(\varphi^*_{\Sigma_2}) = \Delta$ and for $i = 1, 2$, each formula $\varphi^*_{\Sigma_i}$ is obtained from φ by renaming the variables from Σ_{3-i} into "fresh" ones, not present in φ. We claim that φ is Δ–decomposable with the partition $\{\Sigma_1, \Sigma_2\}$ iff the formula $\varphi^*_{\Sigma_1} \wedge \varphi^*_{\Sigma_2} \to \varphi$ is valid.

If φ is Δ–decomposable with components ψ_1 and ψ_2 such that $\mathtt{var}\,(\psi_i) \subseteq \Sigma_i \cup \Delta$ for $i = 1, 2$, then clearly we have $\varphi^*_{\Sigma_i} \vdash \psi_i$. Since φ is equivalent to $\psi_1 \wedge \psi_2$, we obtain $\varphi^*_{\Sigma_1} \wedge \varphi^*_{\Sigma_2} \vdash \varphi$. In the reverse direction, if $\varphi^*_{\Sigma_1} \wedge \varphi^*_{\Sigma_2} \vdash \varphi$, then by interpolation (cf. Lemma 1 in [17]), there exist formulas ψ_1 and ψ_2 such that $\mathtt{var}\,(\psi_i) = \Sigma_i \cup \Delta$ and $\varphi^*_{\Sigma_i} \vdash \psi_i$ for $i = 1, 2$, and $\psi_1 \wedge \psi_2 \vdash \varphi$. Note that any assignment to $\mathtt{var}\,(\varphi)$ satisfying φ can be expanded to an assignment of $\varphi^*_{\Sigma_i}$, for $i = 1, 2$. This yields $\varphi \vdash \psi_i$, $i = 1, 2$ and hence, φ is equivalent to $\psi_1 \wedge \psi_2$. Due to the definition of ψ_1 of ψ_2, we conclude that φ is Δ–decomposable with the variable partition $\{\Sigma_1, \Sigma_2\}$.

Finally, let us prove point 2 of the theorem. The proof of point 1 shows coNP-hardness of $\varnothing$`Dec`, so it remains to provide a P^{NP}-algorithm for solving this problem. Corollary 2 says that if a formula φ in CNF is $\varnothing$–decomposable and contains a clause ξ with variables from both decomposition components, then ξ must contain a subclause $\xi' \subset \xi$ such that $\varphi \vdash \xi'$. Let us demonstrate that this property gives a P^{NP}–algorithm for deciding $\varnothing$`Dec`.

Given a formula φ in CNF, we apply exhaustively the following trasformation rule to φ. For every clause $\xi = \xi' \vee l$ from φ, where l is a literal and ξ' is a subclause, we check the condition $\varphi \vdash \xi'$ (by using an NP–oracle) and eliminate l from ξ if the condition holds. Let φ' be the resulting formula obtained after exhaustive application of this trasformation; clearly, φ' and φ are equivalent. We have $\mathtt{var}\,(\varphi') \subseteq \mathtt{var}\,(\varphi)$ and by Corollary 2, every clause of φ' contains variables only from one $\varnothing$-decomposition component of φ (if decomposition exists). Therefore, the union of the set of clauses of φ' with the set of formulas $\mathtt{taut}(\{x\})$ for each $x \in \mathtt{var}\,(\varphi) \setminus \mathtt{var}\,(\varphi')$ uniquely defines the partition of $\mathtt{var}\,(\varphi)$ (and thus, the decomposition components of φ if it is $\varnothing$–decomposable). Together with Remark 1 this implies that the problem $\varnothing$`Dec` for formulas in CNF belongs to P^{NP}. This completes the proof of Theorem 1. □

We now prove two auxiliary propositions which will be used in the tractability results below.

Proposition 1 (Positiveness Criterion) *A boolean formula φ is equivalent to a positive formula iff the following condition holds:*

($*$) *if* $\eta = \xi \wedge \bigwedge_{x\in\sigma} \neg x$ *is an assignment of* φ*, where* $\varnothing \neq \sigma \subseteq \mathtt{var}\,(\varphi)$ *and* ξ *is a positive conjunct, then for every* $\rho_\sigma \in \wp(\sigma)$*,* $\xi \wedge \rho_\sigma$ *is an assignment of* φ*.*

Proof. ($\Rightarrow$) : Let φ' be a positive formula in DNF which is equivalent to φ. If η is an assignment of φ, then η satisfies at least one of the conjuncts μ of φ', which means that $\mu \subseteq \eta$ and hence, condition ($*$) holds.

($\Leftarrow$) : W.l.o.g. we may assume that φ is in DNF. Suppose that φ contains some conjunct ζ with negative occurrences of variables: $\zeta = \xi \wedge \bigwedge_{x\in\sigma} \neg x$, $\sigma \subseteq \mathtt{var}\,(\varphi)$, where ξ is a positive conjunct. The conjunct ζ defines a set of possible assignments of φ. Denote $\Omega = \wp(\sigma) \setminus \bigwedge_{x\in\sigma} \neg x$. By the condition ($*$), for every $\rho \in \Omega$, there is an assignment $\xi\wedge\rho\wedge\chi$ of φ, where χ is some assignment to the variables from $\mathtt{var}\,(\varphi) \setminus \mathtt{var}\,(\zeta)$. This yields that φ is equivalent to

$$(\varphi \setminus \{\zeta\}) \vee (\xi \wedge \bigwedge_{x\in\sigma} \neg x) \vee \bigvee_{\rho\in\Omega} (\xi \wedge \rho),$$

which is equivalent to the formula $(\varphi \setminus \{\zeta\}) \vee \xi$. Thus, we have eliminated the negative occurrences of σ-variables which appeared in ζ and since the choice of ζ was arbitrary, we conclude that φ is equivalent to a positive formula. □

Remark 2 (Conjunction of DNFs Gives Cartesian Structure) *Taking conjunction of some formulas* $\xi_1 \vee \ldots \vee \xi_m$ *and* $\zeta_1 \vee \ldots \vee \zeta_n$ *in DNF gives the formula in DNF which has the form* $\bigvee(\xi_i \wedge \zeta_j)$*, for all pairs* $\langle \xi_i, \zeta_j\rangle$*,* $1 \leqslant i \leqslant m$*,* $1 \leqslant j \leqslant n$*.*

Proposition 2 (Existence of Positive Components) *If a positive formula* φ *without inessential variables is* Δ*–decomposable for some* $\Delta \subseteq \mathtt{var}\,(\varphi)$*, then it has decomposition components which are positive formulas.*

Proof. Let ψ_1 and ψ_2 be some Δ–decomposition components of φ in DNF and assume that ψ_1 contains a conjunct ζ with negative occurrences of variables: $\zeta = \xi \wedge \bigwedge_{x\in\sigma} \neg x$, $\sigma \subseteq \mathtt{var}\,(\psi_1)$, where ξ is a positive conjunct. If there is no assignment of φ which expands ζ, then from Remark 2 we conclude that φ is equivalent to $(\psi_1\setminus\{\zeta\})\wedge\psi_2$; besides, we have $\mathtt{var}\,(\psi_1\setminus\{\zeta\})\setminus\Delta = \mathtt{var}\,(\psi_1)\setminus\Delta$, since otherwise φ would contain inessential variables. If there exists an assignment of φ which expands ζ, then from Proposition 1 and the idea of its proof we conclude that ψ_1 is equivalent to $(\psi_1 \setminus \{\zeta\}) \vee \xi$. As all the variables are essential in φ, every variable from $\sigma \setminus \Delta$ must be present in some conjunct of $\psi_1 \setminus \{\zeta\}$. Due to the arbitrary selection of the decomposition component ψ_1 and conjunct ζ, we conclude that in both cases, negative occurrences of variables from ψ_i, $i = 1, 2$ are eliminated giving positive Δ–decomposition components of φ. □

Theorem 2 (Complexity for Positive CNF) *For positive boolean formulas in CNF, the problem* $\Delta\mathtt{Dec}$ *is in P. Moreover,* Δ*-decomposition components can be computed in polynomial time (if decomposition exists).*

Proof. The result follows from Proposition 2 and the three short lemmas below which justify the fact that a positive formula (in CNF) has a unique prime and irredundant representation. We provide the full proof here for the sake of completeness.

Lemma 4 (Positive CNF: Criterion for Inessential Variables) *If a variable* x *is inessential in a positive formula* φ *in CNF, then any clause* $\xi \in \varphi$ *containing* x *is redundant in* φ*.*

Proof of the lemma. Assume the opposite, i.e. that the variable x is inessential, but there is a clause $\xi = (\eta \vee x) \in \varphi$, for which no other clause ξ' exists in φ such that $\xi' \subseteq \xi$. Let φ have the form $\xi \wedge \bigwedge_{i \in I} \zeta_i$ for some index set I (which is not empty, because otherwise $\varphi = \xi$ and x would be essential). By our assumption, every clause ζ_i, $i \in I$, must contain a variable y_i which is not in $\mathtt{var}\,(\xi)$. Thus, there is an assignment v of φ, in which all y_i are set to true, x is true, and η is false. Since x is inessential in φ, the assignment, which coincides with v on the values of all variables except x, must satisfy φ, so we arrive at contradiction. □

Lemma 5 (Entailment Criterion) *If φ and ψ are positive formulas in CNF such that $\varphi \vdash \psi$, then for any clause $\xi \in \psi$, there must exist a clause $\xi' \in \varphi$ such that $\xi' \subseteq \xi$.*

Proof of the lemma. Assume there does not exist such a clause ξ' for some $\xi \in \psi$. This means that every clause ζ from φ contains a variable which is not present in ξ. Since φ and ψ are positive, setting these variables to true and the remaining variables of $\mathtt{var}\,(\varphi) \cup \mathtt{var}\,(\psi)$ to false gives an assignment of φ, but clearly it does not satisfy ξ and hence, does not satisfy ψ. □

Lemma 6 (Canonical Form) *If two positive formulas φ and ψ in CNF are equivalent and do not contain redundant clauses, then they are syntactically equal, i.e. consist of the same clauses.*

Proof of the lemma. Let ξ be a clause in φ. We have $\psi \vdash \varphi$, hence by Lemma 5, there must be a clause $\xi' \in \psi$ such that $\xi' \subseteq \xi$. Since $\varphi \vdash \psi$, there exists a clause $\xi'' \in \varphi$ such that $\xi'' \subseteq \xi'$. Thus, there exist clauses ξ and ξ'' in φ such that $\xi'' \subseteq \xi$. If $\xi'' \subset \xi$, then ξ is redundant in φ. Therefore we have $\xi'' = \xi$ and $\xi = \xi'$ and thus, φ and ψ consist of the same clauses. □

Let us complete the proof of Theorem 2. Assume we are given a positive formula φ in CNF and a subset $\Delta \subseteq \mathtt{var}\,(\varphi)$. Let φ' be a formula obtained from φ by removing all redundant clauses; this operation can be done in polynomial time and clearly, φ' is equivalent to φ. If $\Sigma = \mathtt{var}\,(\varphi) \setminus (\mathtt{var}\,(\varphi') \cup \Delta) \neq \varnothing$, then φ is equivalent to the conjunction $\varphi' \wedge taut(\Sigma)$, and thus Δ-decomposable. If $\Sigma = \varnothing$, then φ' is Δ-decomposable iff so is φ. Since φ' does not contain redundant clauses, by Lemma 4, it also does not contain inessential variables. Hence, by Proposition 2, if φ' is Δ-decomposable with some components ψ_1, ψ_2, we may assume that ψ_1 and ψ_2 are positive formulas in CNF.

To apply Lemma 6, let us show that if φ' is Δ-decomposable into positive formulas ψ_1 and ψ_2, then the conjunction $\psi_1 \wedge \psi_2$ does not contain redundant clauses. Assume that every Δ-decomposition of φ' of this kind gives a conjunction $\psi_1 \wedge \psi_2$ having a redundant clause. Eliminating redundant clauses from this conjunction gives an equivalent formula $\psi_1' \wedge \psi_2'$, where $\mathtt{var}\,(\psi_i') \subseteq \mathtt{var}\,(\psi_i)$ for $i = 1, 2$. By our assumption, ψ_1' and ψ_2' are not Δ-decomposition components of φ', thus one of the conditions of points 1-3 from Definition 1 must be violated. The condition in point 2 is preserved by elimination of redundant clauses and one of the remaining conditions is violated iff $\mathtt{var}\,(\psi_1' \wedge \psi_2') \subset \mathtt{var}\,(\psi_1 \wedge \psi_2)$. This means that some variable is inessential in φ', but this is a contradiction due to Lemma 4, since φ' is positive and does not contain redundant clauses.

Finally, by Lemma 6, φ' must be syntactically equal to $\psi_1 \wedge \psi_2$ and thus the set of clauses from φ' uniquely defines the partition of $\mathtt{var}\,(\varphi')$ wrt Δ (and also Δ-decomposition components of φ'). Applying Remark 1 concludes the proof of the theorem. □

Theorem 3 (Complexity for Full DNF) *For boolean formulas in full DNF,*

1. *the problem* Δ`DecPart` *is in P;*

2. *the problem* $\varnothing$`Dec` *is reducible to* `Dec`$\mathbb{F}_2$ *and hence is in P.*

In each of the cases, the corresponding decomposition components can be computed in polynomial time.

Proof. Note the following important characterization:

Lemma 1 (Semantic Criterion of Decomposability) *Let φ be a boolean formula, V be the set of its satisfying assignments, and let $\Delta \subseteq \texttt{var}(\varphi)$ be a subset of variables. The formula φ is Δ–decomposable with a variable partition $\{\Sigma_1, \Sigma_2\}$ iff $V = V|_{\Sigma_1\cup\Delta} \uplus V|_{\Sigma_2\cup\Delta}$, where for $i = 1, 2$, $V|_{\Sigma_i\cup\Delta}$ is the restriction of V onto the variables from $\Sigma_i \cup \Delta$ and $V|_{\Sigma_1\cup\Delta} \uplus V|_{\Sigma_2\cup\Delta}$ is the set of all assignments v such that the restriction of v onto $\Sigma_i \cup \Delta$ belongs to $V|_{\Sigma_i\cup\Delta}$.*

Proof of the lemma. $(\Leftarrow)$: W.l.o.g. we may assume that V and $V|_{\Sigma_i\cup\Delta}$, for $i = 1, 2$, are sets of conjunctions of literals. Consider the formulas in DNF: $\psi_i = \bigvee_{\xi\in V|_{\Sigma_i\cup\Delta}} \xi$, for $i = 1, 2$. We have $\texttt{var}(\psi_i) = \Sigma_i \cup \Delta$. By the conditions of the lemma and by Remark 2, φ is equivalent to $\psi_1 \wedge \psi_2$ and thus Δ–decomposable with the variable partition $\{\Sigma_1, \Sigma_2\}$.

$(\Rightarrow)$: Let ψ_1 and ψ_2 be Δ–decomposition components of φ corresponding to the partition $\{\Sigma_1, \Sigma_2\}$. Then the equivalence of φ and $\psi_1 \wedge \psi_2$ immediately yields the required statement, since for $i = 1, 2$, each $V|_{\Sigma_i\cup\Delta}$ is the set of assignments of ψ_i. □

Proof of point 1. Let φ be a formula in full DNF and $\{\Sigma_1, \Sigma_2\}$ be a partition of $\texttt{var}(\varphi)$. Since full DNF is the explicit representation of all satisfying assignments, one can compute the sets $V|_{\Sigma_i\cup\Delta}$, for $i = 1, 2$ and check whether the condition in Lemma 1 holds in polynomial time. If this is the case, the formulas ψ_i, $i = 1, 2$ from the proof of the lemma are the required Δ–decomposition components.

Proof of point 2. Let φ be a formula in full DNF. Let Σ be a set of variables disjoint from $\texttt{var}(\varphi)$ such that there is an injection $\iota : \texttt{var}(\varphi) \to \Sigma$. Consider the multilinear polynomial F constructed from φ as follows. If $x_1 \wedge \ldots, x_k \wedge \neg x_{k+1} \wedge \ldots \wedge \neg x_n$ is a conjunct of φ, then F has the monomial, which is the product of variables $x_1 \cdot \ldots \cdot x_k \cdot \iota(x_{k+1}) \cdot \ldots \cdot \iota(x_n)$, and no other monomials are in F. In other words, when constructing F, all the negative literals are replaced by "fresh" variables. We claim that φ is $\varnothing$-decomposable iff F is factorable over $\mathbb{F}_2$.

$(\Leftarrow)$: If G_1 and G_2 are factors of F, then they do not have variables in common. Note that for any variable $x \in \texttt{var}(\varphi)$, x occurs in G_i for some $i = 1, 2$ iff $\iota(x)$ occurs in G_i. Otherwise F would have a monomial with both x and $\iota(x)$, which is impossible, since φ is in full DNF. For the same reason, every G_i must contain at least one variable $x \in \texttt{var}(\varphi)$.

Now for $i = 1, 2$, let ψ_i be the boolean formula in DNF constructed from G_i by converting monomials into conjuncts and applying the inverse of the function ι, i.e. if $x_1 \cdot \ldots \cdot x_k \cdot \iota(x_{k+1}) \cdot \ldots \cdot \iota(x_n)$ is a monomial of G_i, then $x_1, \ldots, x_k, \neg x_{k+1}, \ldots, \neg x_n$ is a conjunct of ψ_i. By the above mentioned, we have $\texttt{var}(\psi_1) \cap \texttt{var}(\psi_2) = \varnothing$. Then by Remark 2, the condition $F = G_1 \cdot G_2$ yields that φ is equivalent to $\psi_1 \wedge \psi_2$, i.e. ψ_1 and ψ_2 are the required decomposition components.

$(\Rightarrow)$: If φ is $\varnothing$-decomposable, then it has decomposition components ψ_1 and ψ_2 as described in the proof of Lemma 1 for $\Delta = \varnothing$. Since $\texttt{var}(\psi_1) \cap \texttt{var}(\psi_2) = \varnothing$, we conclude that F has factors corresponding to ψ_1 and ψ_2. □

Theorem 4 (Decomposition of Positive DNF and Factorization)
For positive boolean formulas in DNF without redundant conjuncts, the problem $\varnothing$`Dec` *is equivalent to* `Dec`$\mathbb{F}_2$.

Proof. Let φ be a positive formula in DNF without redundant conjuncts and let F be the multilinear polynomial corresponding to φ. We show that φ is $\varnothing$–decomposable iff F is factorable over $\mathbb{F}_2$.

($\Leftarrow$): If $F = G_1 \cdot G_2$, then G_1 and G_2 do not have variables in common and every monomial of F is a product of some monomials from G_1 and G_2. Neither of the polynomials G_1 and G_2 can have the constant monomial (i.e. 1), because otherwise φ would contain redundant conjuncts. For $i = 1, 2$, let ψ_i be the formula in DNF corresponding to G_i, i.e. ψ_i is the disjunction of conjuncts corresponding to the monomials of G_i. By Remark 2, then φ is $\varnothing$–decomposable with the components ψ_1 and ψ_2.

($\Rightarrow$): Let us first prove the auxiliary lemma which is analogous to Lemma 4, but is formulated for the case of DNF.

Lemma 7 (Positive DNF: Criterion for Inessential Variables) *If a variable x is inessential in a positive formula φ in DNF, then any conjunct $\xi \in \varphi$ containing x is redundant in φ.*

Proof of the lemma. Assume the opposite, i.e. the variable x is inessential in φ, but there is a conjunct $\xi = (\eta \wedge x)$ of φ, for which no other conjunct ξ' exists in φ such that $\xi' \subseteq \xi$. Let φ have the form $\xi \vee \bigvee_{i \in I} \zeta_i$ for some index set I (which is not empty, since otherwise $\varphi = \eta \wedge x$ and x is essential). By our assumption, every conjunct ζ_i, $i \in I$, contains a variable y_i, which is not in $\mathtt{var}\,(\xi)$. Thus, there is an assignment v of φ, in which all y_i are set to false, η is true and x is true. Since x is inessential in φ, the assignment, which agrees with v on all the variables except x, must also satisfy φ, so we arrive at contradiction. □

Let us complete the proof of Theorem 4. Let φ be a positive formula in DNF without redundant conjuncts. Let φ be $\varnothing$–decomposable into some components ψ_1 and ψ_2. By Lemma 7 and Proposition 2, we may assume that these formulas are positive. Let us prove that taking conjunction of ψ_1 and ψ_2 gives a formula which is syntactically equivalent to φ (i.e. it consists of the same conjuncts as φ). By Remark 2, every conjunct from $\psi_1 \wedge \psi_2$ has the form $\xi_1 \wedge \xi_2$ for some $\xi_i \in \psi_i$, $i = 1, 2$. Let ξ be a conjunct from φ. As $\varphi \vdash \psi_1 \wedge \psi_2$, we have $\xi \vdash \psi_1 \wedge \psi_2$. Then there must be a conjunct ξ' in $\psi_1 \wedge \psi_2$ such that $\xi' \subseteq \xi$. Otherwise, if every conjunct from $\psi_1 \wedge \psi_2$ has a variable not in ξ, we immediately conclude $\xi \nvdash \psi_1 \wedge \psi_2$, which is a contradiction. Indeed, then there would exist an assignment v of φ, which makes ξ true and all the variables from $\mathtt{var}\,(\psi_1 \wedge \psi_2) \setminus \mathtt{var}\,(\xi)$ false.

Similarly, as $\psi_1 \wedge \psi_2 \vdash \varphi$, for ξ' there must be a conjunct ξ'' in φ such that $\xi'' \subseteq \xi'$. We obtain that there are conjuncts ξ and ξ'' in φ such that $\xi'' \subseteq \xi$. Since φ does not contain redundant conjuncts, we conclude that $\xi'' = \xi$ and hence, $\xi = \xi'$. The argument shows that φ and $\psi_1 \wedge \psi_2$ consist of the same conjuncts, i.e. we have shown that if φ is $\varnothing$–decomposable, then there exists a decomposition into components ψ_1 and ψ_2 such that the conjunction of ψ_1 and ψ_2 gives φ. By Remark 2, we obtain that the polynomial F is factorable. □

Corollary 1 (Complexity for DNF)

1. *For formulas in DNF, the problems* $\varnothing$`DecPart` *and* Δ`DecPart` *are coNP-complete;*

2. *for positive boolean formulas in DNF, the problem* $\varnothing$`Dec` *is in P and the corresponding decomposition components can be computed in polynomial time.*

Proof. (1): coNP–hardness of $\varnothing$`DecPart` (and hence, of Δ`DecPart`) follows the the proof of Theorem 1. The set of formulas in DNF which are valid or unsatisfiable is coNP-complete and it suffices to note that if φ is a formula in DNF, then by Remark 2, the formula ψ constructed in the proof of point 1 of Theorem 1 can be converted into DNF in polynomial time, since $\bigwedge_{x \in \mathtt{var}(\varphi)} (q \vee x)$ is equivalent to $q \vee \bigwedge_{x \in \mathtt{var}(\varphi)} x$. The containment of $\Delta\mathtt{DecPart}$ in coNP is shown by the second part of the proof of point 1 of Theorem 1, since the construction there does not depend on the normal form, in which the formula φ is given.

(2): Let φ be a positive formula in DNF and let φ' be a formula obtained from φ by removing all redundant conjuncts. This operation can be done in polynomial time and clearly, φ' is equivalent to φ. If $\Sigma = \mathtt{var}(\varphi) \setminus \mathtt{var}(\varphi') \neq \varnothing$, then φ is equivalent to the conjunction $\varphi' \wedge \mathit{taut}(\Sigma)$, and thus $\varnothing$-decomposable. If $\Sigma = \varnothing$, then φ' is $\varnothing$-decomposable iff so is φ. Since φ' does not contain redundant conjuncts, applying Theorems 4 and 5 proves the required statement. $\square$