On Tractability of Disjoint AND-Decomposition of Boolean Formulas*

Pavel Emelyanov[1], Denis Ponomaryov[2]

[1] Institute of Informatics Systems, Novosibirsk, Russia;
Novosibirsk State University
[2] Institute of Artificial Intelligence, University of Ulm, Germany;
Institute of Informatics Systems, Novosibirsk, Russia

emelyanov@iis.nsk.su, ponom@iis.nsk.su

Abstract. Disjoint AND-decomposition of a boolean formula means its representation as a conjunction of two (or several) formulas having disjoint sets of variables. We show that deciding AND-decomposability is intractable in general for boolean formulas given in CNF or DNF and prove tractability of computing AND-decompositions of boolean formulas given in positive DNF, Full DNF, and ANF. The results 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 [11] contains an extensive survey of decomposition methods till the mid–1990's. The results of the next fifteen years of research are presented in [8, 15, 7, 2, 4, 5, 3].

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: [9, 1, 8, 4, 2, 3] and [5, 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 $X \backslash \Delta$. As a rule, a decomposition into more than two components can be obtained by iterative computation of bi-decomposition. If $\Delta = \varnothing$ then decomposition is called disjoint and considered as optimal for many reasons.

Bioch [2] studied 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 $F(X) =$

* An extended version of the paper containing proofs is available from http://persons.iis.nsk.su/files/persons/pages/and-decomp-full.pdf

$H(G(A), B)$, 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.

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 DNF, Full DNF, and ANF. It is not obvious, whether the technique used by Bioch for positive DNF is applicable to these cases of AND-decomposition. We note however that in our Lemma 1, the idea of computing decomposition components resembles the final step of constructing components in [2, Sect. 2.9].

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. Shpilka and Volkovich [14] noted the strong connection 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. We note however that while staying in the cubic time complexity, the same can be achieved without computing the product explicitly, thus contributing to efficiency of factorization of large input polynomials.

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 $\mathtt{var}(\varphi)$. If Σ is a set of propositional variables and $\mathtt{var}(\varphi) \subseteq \Sigma$, then we say that the formula φ is *over variables* Σ (or *over* Σ, for short); $\mathtt{taut}(\Sigma)$ denotes a valid formula over Σ. 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 DNF, then a conjunct

ξ of φ is called *redundant* in φ if there exists another 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 [12], where it is given in a general form.

Definition 1 (Decomposability) *A boolean formula φ is called disjointly AND–decomposable (or 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. $\mathtt{var}(\psi_1) \cup \mathtt{var}(\psi_2) = \mathtt{var}(\varphi)$;
2. $\mathtt{var}(\psi_1) \cap \mathtt{var}(\psi_2) = \varnothing$;
3. $\mathtt{var}(\psi_i) \neq \varnothing$, *for* $i = 1, 2$.

The formulas ψ_1 and ψ_2 are called decomposition components of φ. We say that φ is decomposable with a variable partition $\{\Sigma_1, \Sigma_2\}$ if φ has some decomposition components ψ_1 and ψ_2 over the variables Σ_1 and Σ_2, 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 iff $\neg\varphi$ is OR–decomposable.

Observe that Definition 1 is formulated with the two components ψ_1 and ψ_2, which in turn can be 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 $\mathtt{var}(\varphi)$ and the fact below (which follows from a property of a large class of logical calculi shown in [12]) says that this variable partition is unique.

Fact 1 (Uniqueness of Decompositions - Corollary of Thm. 1 in [12]) *If a boolean formula φ is decomposable, then there is a unique partition $\{\pi_1, \ldots, \pi_n\}$ of $\mathtt{var}(\varphi)$, $2 \leqslant n$, such that φ is equivalent to $\bigwedge\{\psi_i \mid \mathtt{var}(\psi_i) = \pi_i,\ i = 1, \ldots, n\}$, where each formula ψ_i is not decomposable.*

This means that *any* possible algorithm[1] 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.

2.2 The Computational Problems Considered in the Paper

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

$\boxed{\varnothing\mathtt{Dec}}$ *For a given boolean formula φ, decide whether φ is decomposable.*

[1] Existence and complexity of decomposition algorithms in various logics have been studied in [10, 6, 13, 12].

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

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 $\mathbb{F}_2$. 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 $\mathbb{F}_2$.*

3 Main Results

First, we formulate the hardness result on decomposition of formulas given in the Conjunctive Normal Form and then proceed to formulas in full DNF, positive DNF, and ANF. We note that decomposition itself is conceptually closer to the CNF representation, since it gives a conjunction of formulas. 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 problem $\varnothing\texttt{DecPart}$ is coNP–complete;*
2. *the problem $\varnothing\texttt{Dec}$ is coNP–hard and is in P^{NP}.*

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 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 decomposition of formulas in Full DNF, even though such formulas contain negative literals. The proof of the theorem below uses the trick that negative literals can be encoded as "fresh" variables giving a positive DNF.

Theorem 2 (Complexity for Full DNF) *For boolean formulas in Full DNF,*

1. *the problem* $\varnothing$`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.

It turns out that for a positive formula φ in DNF without redundant conjuncts, 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 $(x \vee z) \wedge \mathtt{taut}(\{y\})$ and thus, 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 3 (Decomposition of Positive DNF and Factorization) *For positive boolean formulas in DNF without redundant conjuncts, the problem* $\varnothing$`Dec` *is equivalent to* `Dec`$\mathbb{F}_2$*.*

We formulate the main result on formulas given in DNF in the following corollary which is a consequence of Theorems 3, 4, and the constructions from the proof of Theorem 1 given in the extended version of the paper.

Corollary 1 (Complexity for DNF)

1. *For formulas in DNF, the problem* $\varnothing$`DecPart` *is 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.*

We now turn to tractability of the problem `Dec`$\mathbb{F}_2$, to which the decomposition problems in Theorem 2 and Corollary 1 are reduced. Originally, tractability of `Dec`$\mathbb{F}_2$ is a consequence of the results from [14], 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 this gives a decomposition algorithm of quartic complexity for factoring multilinear polynomials over the field $\mathbb{F}_2$. 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 4 below, we provide a solution to factorization of multilinear polynomials over $\mathbb{F}_2$, which is different from the both algorithms proposed in [14]. The only common feature between the approaches is application of identity testing, which seems to be inevitable in factorization. Our solution is based on computation of partial derivatives of polynomials obtained from the input one and gives an algorithm of cubic complexity. More precisely, the product $f_1 \cdot f_2$ is computed, where f_i are polynomials obtained from the input, and then for each variable x, the partial derivative of $f_1 \cdot f_2$ is tested for equality to zero. In particular, our algorithm operates polynomials which are smaller than the ones considered in [14]. Moreover, we note in the extended version of the paper that the same can be achieved without computing the product $f_1 \cdot f_2$ explicitly, which is particularly important on large inputs. We present the factorization algorithm as the theorem below to follow the complexity oriented style of exposition used in this paper.

Theorem 4 (Tractability of Linear Polynomial Factorization over $\mathbb{F}_2$) *The problem* $\mathtt{Dec}\mathbb{F}_2$ *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 1 (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. $F_i = G_i$. $\square$

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[2].

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. 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:

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

It follows that in case y is a variable from H, we have $\frac{\partial}{\partial y}(Q \cdot R) = 0$ 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.

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

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

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

[2] 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.

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

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 equals $\frac{\partial F}{\partial x} \cdot F_{x=0}$). 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 to 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 1 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 1 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 identity testing is trivial in $\mathbb{F}_2$). 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 1 whether this partition indeed corresponds to factors of F.

4 Conclusions

We have noted that 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 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 also applicable to the latter case. The factorization algorithm for multivariate polynomials over $\mathbb{F}_2$ given in this paper can be used to implement an efficient solution to disjoint AND-decomposition of formulas in DNF and ANF. It is an open question whether the algorithm can be used for obtaining decompositions of boolean formulas with a non-empty shared set of variables between the components. Further research questions include implementation of the 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".

References

1. Bengtsson, T., Martinelli, A., Dubrova, E.: A fast heuristic algorithm for disjoint decomposition of Boolean functions. In: Notes of the 11^{th} IEEE/ACM International Workshop on Logic & Synthesis (IWLS'02). (2002) 51–55
2. Bioch, J.C.: Decomposition of Boolean functions. In Crama, Y., Hammer, P.L., eds.: Boolean Models and Methods in Mathematics, Computer Science, and Engineering. Volume 134 of Encyclopedia of Mathematics and its Applications. Cambridge University Press, New York, NY, USA (2010) 39–78
3. Chen, H., Janota, M., Marques-Silva, J.: QBF-based Boolean function bi-decomposition. In: Proceedings of the Design, Automation & Test in Europe Conference (DATE'12), IEEE (2012) 816–819
4. Choudhury, M., Mohanram, K.: 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, IEEE Press (2010) 586–591
5. Khatri, S.P., Gulati, K., eds.: Advanced Techniques in Logic Synthesis, Optimizations and Applications. Springer, New York Dordrecht Heidelberg London (2011)
6. Konev, B., Lutz, C., Ponomaryov, D., Wolter, F.: Decomposing description logic ontologies. In: Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning (KR'10), Palo Alto, California, USA, AAAI Press (2010)
7. Kuon, I., Tessier, R., Rose, J.: FPGA Architecture: Survey and Challenges. Now Publishers Inc, Boston - Delft (2008)
8. Mishchenko, A., Sasao, T.: Large-scale SOP minimization using decomposition and functional properties. In: Proceedings of the 40^{th} ACM/IEEE Design Automation Conference (DAC'03), New York, NY, USA, ACM (2003) 149–154
9. Mishchenko, A., Steinbach, B., Perkowski, M.A.: An algorithm for bi-decomposition of logic functions. In: Proceedings of the 38^{th} ACM/IEEE Design Automation Conference (DAC'01), New York, NY, USA, ACM (2001) 103–108
10. Morozov, A., Ponomaryov, D.: On decidability of the decomposability problem for finite theories. Siberian Mathematical Journal **51**(4) (2010) 667–674
11. Perkowski, M.A., Grygiel, S.: 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)
12. Ponomaryov, D.: On decomposability in logical calculi. Bulletin of the Novosibirsk Computing Center **28** (2008) 111–120, available at http://persons.iis.nsk.su/files/persons/pages/delta–decomp.pdf
13. Ponomaryov, D.: 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)
14. Shpilka, A., Volkovich, I.: 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). Volume 6198 of Lecture Notes in Computer Science., Springer (2010) 408–419
15. Steinbach, B., Lang, C.: Exploiting functional properties of Boolean functions for optimal multi-level design by bi-decomposition. Artificial Intelligence Review **20**(3–4) (2003) 319–360