Electronic Colloquium on Computational Complexity, Report No. 13 (2000)

Algebraic and Uniqueness Properties of Parity Ordered Binary Decision Diagrams and their Generalization

Daniel Král'
Charles University
Malostranské náměstí 25, 118 00 Praha 1
Czech Republic
E-mail: kral@atrey.karlin.mff.cuni.cz

Abstract

Ordered binary decision diagrams (OBDDs) and parity ordered binary decision diagrams have proved their importance as data structures representing Boolean functions. In addition to parity OBDDs we study their generalization which we call parity AOBDDs, give the algebraic characterization theorem and compare their minimal size to the size of parity OBDDs.

We prove that the constraint that no arcs test conditions of type $x_i = 0$ does not affect the node–size of parity (A)OBDDs and we give an efficient algorithm for finding node–minimal parity (A)OBDDs with this additional constraint. We define so–called uniqueness conditions, use them to obtain a canonical form for parity OBDDs and discuss similar results for parity AOBDDs.

Algorithms for minimization and creating the form which satisfies the uniqueness conditions for parity OBDDs running in time $O(S^3)$ and for parity AOBDDs running in time $O(nS^3)$ are presented (n is the number of variables, S is the number of vertices); both the algorithms are quite simple.

All the results are also extended to case of shared parity OBDDs and shared parity AOBDDs — data structures for representation of Boolean function sequences.

1 Introduction

Data structures representing Boolean functions play a key role in formal circuit verification. They are also important as combinatorial structures corresponding to Boolean functions and have applications also in other fields. Once a data structure representing Boolean functions is chosen it should allow compact representation of many important functions and fast implementation of fundamental algorithms (for surveys see [2], [7], [8]), e.g. the equivalence testing (decision whether two functions represented by different structures are identical), the minimization (finding the smallest structure representing the given function), the synthesis (applying operations to represented functions to obtain new ones). Data structures for representation of Boolean function sequences are also studied and used.

Graph–based data structures for Boolean functions allow to implement algorithms for Boolean function manipulation using standard graph algorithms. Ordered binary decision diagrams (OBDDs), binary decision diagrams (BDDs) with an additional variable–ordering constraint, as a data structure for Boolean functions were introduced in [1]. Excellent algorithmic properties of OBDDs are the reason why they are applied in many cases. Their more

ISSN 1433-8092

powerful modification — parity OBDDs were introduced in [4] and further investigated in [6]. Further information may be found in [7]. In addition to parity OBDDs we also consider their generalization — Parity Arc–ordered binary decision diagrams (AOBDDs). We restrict the order in which the variables are to be queried in parity AOBDDs but we allow to test different variables in the same vertex (for definitions of both parity OBDDs and parity AOBDDs see Section 2). The minimal node–size of a parity AOBDD never exceeds the minimal node–size of a parity OBDD for the same function. We also study shared parity OBDDs and shared parity AOBDDs — structures for representation of Boolean function sequences.

The size–minimality of used data structures plays an essential role in efficiency of used algorithms; the smallest possible sizes of used data structures are studied. We prove the algebraic characterization theorem for parity AOBDDs (Theorem 3.4 and Theorem 3.5); it is analogous to the characterization theorem for parity OBDDs (Theorem 3.6) which was proved in [6]. Next, the relationship between the size of a minimal parity OBDD and the size of a minimal parity AOBDD for the given function is discussed (Theorem 3.7).

The minimal data structure representing the given function can possibly have more different non-isomorphic forms. It is demanded that the canonical form must be size–minimal, it must exist for all Boolean functions and it must be polynomially computable from any non-canonical form of the data structure (see [2]). Existence, variability of the definition and properties of canonical forms of used data structures are interesting from the theoretical point of view. Ivestigation of canonical forms leads to understanding of the freedom in the choice of data structures representing the given function and may lead to the design of better data structures. We choose one of the forms to be a canonical one and present an algorithm for constructing such a canonical form for a given parity OBDD similar to Waack's algorithm for minimization. Actually, it is simpler, at least its last (transform) phase which does not use the Gaussian elimination.

In Section 2 we give definitions of used data structures and introduce notation used in the paper. In Section 3 we study the size–minimality of parity OBDDs and parity AOBDDs. We prove that the constraint that parity BDDs, parity OBDDs and parity AOBDDs do not contain negative arcs, i.e. arcs testing conditions $x_i = 0$, does not affect the node–size of the node–minimal diagram representing the given function (Theorem 3.1 and Theorem 3.2). We give an efficient algorithm (*Removal of zero arcs*) for finding such parity (A)OBDDs (see Subsection 6.2). The main theorem of Section 3 is the algebraic characterization theorem for parity AOBDDs (Theorem 3.4 and Theorem 3.5) in which the size of the node–minimal parity AOBDD representing a given function is expressed in terms of the dimension of an appropriate linear space.

In Section 4 we define the uniqueness conditions for parity AOBDDs. The properties of parity AOBDDs satisfying the uniqueness conditions are in depth studied in Theorem 4.3. Unfortunately, parity AOBDDs representing the same function which satisfy the uniqueness conditions need not to be isomorphic. In Section 5 we define the uniqueness conditions for parity OBDDs and prove the canonicity of the representations which satisfy the uniqueness conditions for parity OBDDs (Theorem 5.3). We give an efficient algorithm (*Unification*) for finding structures which satisfy the uniqueness conditions (see Subsection 6.5) both for parity AOBDDs and parity OBDDs. In case of parity OBDDs we also prove directly (non–algorithmically) the existence of such representation and we give a linear–time algorithm for finding the isomorphism between canonical forms (the *PDFS algorithm*). All the results are extended to case of shared parity OBDDs and shared parity AOBDDs.

Waack ([6]) presented the algorithm for node–minimization of parity OBDDs running in time $O(nS^{\omega})$ where ω is the exponent of matrix multiplication (currently the best achieved one is 2.376 [3], but the practicle–usable algorithms achieve only $\omega = 3$) where S is the size of the diagram (the number of its vertices). Löbbing, Sieling and Wegener ([5]) proved that if there exists an algorithm for node–minimization of parity OBDDs running in time $O(t(S))$ then there exist an algorithm for computing the rank of a Boolean $S \times S$ matrix running in time $O(t(S))$; thus we can hardly hope to find a practicle–usable algorithm for node–minimization of parity OBDDs running in time $o(S^3)$. In Section 6, we desribe the algorithm for node–minimization of parity OBDDs running in time $O(S^3)$. Our algorithm does not use any of methods for fast matrix multiplication. The application of Gaussian elimination procedure is completely eliminated from the last (transform) phase of minimization. For overview of running times for presented algorithms see Table 1 at the beginning of Section 6.

2 Definitions and Basic Properties

Let us denote by F_2 the two–element field. We understand the set B_n of Boolean functions of n variables as 2^n–dimensional vector space over F_2 (see also [6]).

A *parity arc–ordered binary decision diagram* with respect to a permutation π of the set $\{1, 2, \ldots, n\}$ is an acyclic digraph[1] with the properties described below. We abbreviate the name of the structure to *⊕AOBDD* or to π–*⊕AOBDD*. The permutation π restricts the order of the input variables in which they are to be queried: we demand to query the variables in the following order: $x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}$. There are two special vertices, we call them a *source* and a *sink*. If the function is all–zero then the presence of the sink is not necessary. There exists a dipath from the source to each of the vertices and a dipath from them to the sink. Each arc except for those from the source is labelled with a pair consisting of a variable x_i and an element of F_2. Arcs from the source are unlabelled with either variables or elements of F_2. We call arcs labelled with zero *negative arcs* and arcs labelled with one *positive arcs*. Every two vertices can be joined by no, one or more arcs. Every sequence of variable–indices induced by variables labelled to arcs of any dipath from the source to the sink is strictly π–increasing; that means the variables on any dipath from the source to the sink are queried in the prescribed order.

With an additional constraint that arcs from any vertex (except for those from the source) have to be labelled with the same variable our definition becomes the definition of *parity OBDDs* (for the definition of parity OBDDs see [6]). In this case we consider instead of variable–labelling of the arcs a variable–labelling of the vertices of the ⊕OBDD. If we leave the variable–ordering constraint the definition of parity OBDDs becomes the definition of *parity BDDs*.

We consider the number of vertices as the size of the ⊕AOBDD as in case of ⊕OBDDs ([6]). As in case of ⊕OBDDs the actual storage size of ⊕AOBDDs can be larger. The storage size of a ⊕AOBDD with the size S belongs both to $\Omega(S)$ and $O(nS^2)$. Let π' be the reversed permutation to π, i.e. $\pi'(i) = \pi(n + 1 - i)$. It is clear that the size of the minimal π–⊕AOBDD is equal to the size of the minimal π'–⊕AOBDD.

We write f_{B} for the function represented by a ⊕AOBDD B. The value of $f_{\mathrm{B}}(w_1, w_2, \ldots, w_n)$ is 1 iff there is an odd number of dipaths from the source to the sink using only admissible arcs for an assignment w_i to x_i $(1 \leq i \leq n)$. For an assignment of Boolean values to the variables the set of admissible arcs is the set of all arcs from the sink and arcs for which their variables and Boolean values labelled to them are consistent with the given assignment. Notice that the representation of Boolean functions with the additional constraint from the previous paragraph becomes the represenation of Boolean function for ⊕OBDDs. It is possible to extend our definition of a parity arc–ordered BDD

[1] We abbreviate *directed graph* to *digraph*.

to a *shared parity AOBDD* with more sources to represent the set of Boolean functions in the same manner as in [6].

It is a straightforward use of the definition to implement an algorithm for computing the function represented by a $\oplus$AOBDD with running time linear in the number of arcs.

Let v be a vertex of a $\oplus$AOBDD then f_v is a function of Boolean variables $x_1, \ldots, x_n$ which is equal 1 iff there is an odd number of admissible dipaths for the given variable–assignment from v to the sink. Note that if v is the source then f_v is the function represented by the $\oplus$AOBDD; if v is the sink then f_v is the all–one Boolean function. We call the function f_v the function represented by the vertex v. Let V be a set of vertices then f_V is $\oplus_{v \in V} f_v$ where $\oplus$ is F_2–addition. Let us denote by $\wedge$ F_2–multiplication and by span F the linear span of the set of Boolean functions F.

Let f be a Boolean function of n variables and F be a set of n–variable Boolean functions. Let the function $\triangle_i f$ be defined in the following way ($\oplus$ is F_2-addition):

$$(\triangle_i f)(x_1, \ldots, x_n) = f(0, \ldots, 0, 1, x_{i+1}, \ldots, x_n) \oplus f(0, \ldots, 0, 0, x_{i+1}, \ldots, x_n)$$

Let $\triangle f$ be a set $\{\triangle_i f, 1 \leq i \leq n\}$ and $\triangle F$ be the union $\bigcup_{f \in F} \triangle f$. Let $\triangle^j F$ be $\triangle \triangle^{j-1} F$ for $j \geq 1$ and $\triangle^0 F$ be F and $\triangle^\star f$ be an union $\bigcup_{i=0}^{\infty} \triangle^i \{f\}$. Note that $\triangle^\star f = \bigcup_{i=0}^{n} \triangle^i \{f\}$ and that $\triangle^\star f$ is the smallest set of Boolean functions containing f and closed under all operations $\triangle_i$ $(1 \leq i \leq n)$.

Let $\diamond_i f$ be the set of all n-variable boolean functions g for which there exists constants $c_1, \ldots, c_i$ such that $g(x_1, \ldots, x_n) = f(c_1, \ldots, c_i, x_{i+1}, \ldots, x_n)$. Let $\diamond f$ be equal to $\cup_{i=0}^{n} \diamond_i f$.

3 Size–Minimality

In the whole section we assume that the permutation of variable indices is an identity, i.e. $\pi(i) = i$. First, we prove that the usage of negative arcs is needless.

Theorem 3.1 *Let* B *be a* $\oplus$*AOBDD. Then there exists a* $\oplus$*AOBDD representing the same function and of the same size without negative arcs.*

Proof: First we allow the digraph to contain unlabelled arcs from all its vertices; these arcs are admissible for every variable–assignment. In the $\oplus$AOBDD B we replace each arc labelled with variable x_i and value 0 with an unlabelled arc and an arc labelled with x_i and 1. It is easy to see that this operation does not affect the function represented by the $\oplus$AOBDD.

Let $v_1, v_2, \ldots, v_n$ be an ordering of the vertices of B such that there is no arc from v_i to v_j for any $i > j$. This ordering exists because the digraph is acyclic. Note that v_1 has to be the source and v_n has to be the sink. Define an operation *remove*(e) for an unlabelled arc e from v to w as follows: First we duplicate the vertex v and create a new vertex v' such that arcs from the same vertices and with the same labels lead to the vertex v' as to v. We remove e from arcs leading from v and make e to be the only arc leading from v'. It is obvious that the function represented by the $\oplus$AOBDD has not changed. Now we contract the arc leading from v', i.e. arcs leading to v' are redirected to w (their labels are not changed). Remember there was only one arc leading from v' and this arc was unlabelled. If there arise two arcs with the same label joining the same pair of vertices we remove both of them. Note that the whole operation does not affect the represented function, the size of the $\oplus$AOBDD and it can create a new unlabelled arc leading only from descendants of v. If we apply the operation remove to all unlabelled arcs from v_{n-1}, and then to all unlabelled arcs from $v_{n-2}, \ldots,$ and then to all unlabelled arcs from v_2, the resulting $\oplus$AOBDD has got the properties described in the theorem. ■

Notice that in the same way it is possible to prove the following theorems:

Theorem 3.2 *Let* B *be a* $\oplus$*OBDD. Then there exists a* $\oplus$*OBDD representing the same function and of the same size without negative arcs.*

Theorem 3.3 *Let* B *be a* $\oplus$*BDD. Then there exists a* $\oplus$*BDD representing the same function and of the same size without negative arcs.*

Notice that it is possible to exchange the role of negative and positive arcs in Theorem 3.1, Theorem 3.2 and Theorem 3.3. In the following we consider only $\oplus$AOBDDs which do not contain negative arcs. Hence every labelled arc is positive.

Before proving the main theorem of this section we state the following three lemmas:

Lemma 3.1 *Let* V *be a set of vertices and let the source be not an element of* V. *Then for each* $1 \leq i \leq n$ *there exists a set of vertices* W *such that* $\triangle_i f_V = f_W$ *and* W *does not contain the source.*

Proof: It is enough to set W to the set of all vertices to which odd number of arcs labelled with variable x_i lead from the vertices of the set V. ■

Lemma 3.2 *Let* f *be an* n*-variable Boolean function. Then there are uniquely determined* n*-variable Boolean functions* $h_1, \ldots, h_n$ *and Boolean constant* c *which satisfy:*

- *The function* h_i *does not essentially depend on the first* i *variables.*
- $f = c \oplus \bigoplus_{i=1}^{n} (x_i \wedge h_i)$

The function h_i *is equal to* $\triangle_i f$ *and the constant* c *is equal to* $f(0, \ldots, 0)$.

Proof: We prove the lemma by the induction on n. For $n = 1$ the lemma is clear. Let $n > 1$. Since the expression $c \oplus \bigoplus_{i=2}^{n} (x_i \wedge h_i)$ does not essentially depend on the first variable and $x_1 \wedge h_1$ is zero for $x_1 = 0$ it must hold that $f(0, x_2, \ldots, x_n) = c \oplus \bigoplus_{i=2}^{n} (x_i \wedge h_i)$. It follows from the induction hypothesis applied to the restriction of f to the last $n-1$ variables that $c = f(0, \ldots, 0)$ and $h_i = \triangle_i f$ for $i \geq 2$ and that these functions are uniquely determined. Thus it holds:

$$f(x_1, x_2, \ldots, x_n) = (x_1 \wedge h_1) \oplus c \oplus \bigoplus_{i=2}^{n} (x_i \wedge h_i)$$

$$f(x_1, x_2, \ldots, x_n) = (x_1 \wedge h_1) \oplus f(0, x_2, \ldots, x_n)$$

$$f(1, x_2, \ldots, x_n) = h_1 \oplus f(0, x_2, \ldots, x_n)$$

This implies that $h_1 = \triangle_1 f$ and that h_1 is uniquely determined. ■

The main result of this section is the following theorem:

Theorem 3.4 *Let* $\oplus$*AOBDD* B *be the size–minimal* $\oplus$*AOBDD representing the function* f, *i.e. one with the minimal number of vertices. Then the number of its vertices is equal to* $1 + \dim_{F_2} \operatorname{span} \ \triangle^\star f$

Proof: Let V be the vertices where an odd number of arcs from the source lead to. Because all arcs from the source are unlabelled it is clear that $f_V = f_{\mathrm{B}}$. From Lemma 3.1 it follows that for each function $g \in \triangle^\star f$ there exists a set of vertices W not containing the source such that $g = f_W$. Thus also for every $g \in \operatorname{span} \ \triangle^\star f$ there exists a set of vertices W not containing the source such that $g = f_W$. Thus the number of vertices distinct from the source must be at least $\dim_{F_2} \operatorname{span} \ \triangle^\star f$ and thus there must be at least $1 + \dim_{F_2} \operatorname{span} \ \triangle^\star f$ vertices altogether.

Now, we prove that the number of vertices mentioned in the theorem is sufficient. If the function is all–zero then $\dim_{F_2} \operatorname{span} \ \triangle^\star f$ equals 0 and the $\oplus$AOBDD with the only vertex (the source) represents the function. Otherwise $I \in \triangle^\star f$ where I is the all–one Boolean function. Let $\triangle^{\star(i)} f$ be those functions which belong to $\triangle^\star f$ and do not essentially depend on the variables $x_1, \ldots, x_i$ (remember we have assumed that π is the identity), clearly $\triangle^\star f = \triangle^{\star(0)} f \supseteq \triangle^{\star(1)} f \supseteq \ldots \supseteq \triangle^{\star(n)} f \supset \{I\}$. Let $f_1, \ldots, f_k$ be a basis of span $\triangle^\star f$ with the following property: For each i there exists j that $f_j, \ldots, f_k$ is a basis of span $\triangle^{\star(i)} f$. It is clear that f_k necessarilly equals I. W.l.o.g. we can assume $i < k$ $f_i(0, \ldots, 0) = 0$; if this is not the case then it is enough to replace f_i with $f_i + I$. Now, we construct a $\oplus$AOBDD, each vertex of which represents one of the basis functions and which contains only these vertices and the source.

The $\oplus$AOBDD construction is made from the end of the basis sequence. The vertex representing $f_k = I$ is the sink. Let $i < k$ and assume the structure with vertices for $f_{i+1}, \ldots, f_k$ has been constructed. Let l be the index such that $f_i \in$ span $\triangle^{\star(l)} f \setminus$span $\triangle^{\star(l+1)} f$. From the definition and Lemma 3.2 it follows that $f_i = \oplus_{j=l+1}^{n} (x_j \wedge \triangle_j f_i)$. Since $\triangle_j f_i \in \triangle^{\star(j)} f$ for each $l+1 \leq j \leq n$ the function $\triangle_j f_i$ is expressible as the combination of some of the functions $f_{i+1}, \ldots, f_k$. Add a new vertex w to represent the function f_i and for j such that $l+1 \leq j \leq n$ add arcs from this vertex to the vertices whose combination represent $\triangle_j f_i$ and label these arcs with x_j (and the value one). Note that arcs leading from the vertex representing $f_i \in$ span $\triangle^{\star(l)} f \setminus$span $\triangle^{\star(l+1)} f$ are labelled only with variables $x_{l+1}, \ldots, x_n$ and those labelled with the variable x_j lead to vertices representing the basis of span $\triangle^{\star(j)} f$. Thus the constructed digraph really satisfies the variable–ordering condition. Now add (unlabelled) arcs from the source to the vertices whose combination represents the function f. ■

Notice that w.l.o.g. we can in the proof assume that $f_1 \in \{f, f+I\}$ and thus we obtain the following corollary (the similar result also holds for $\oplus$OBDDs):

Corollary 3.1 *For each function f there exists a size–minimal $\oplus$AOBDD with at most 2 unlabelled arcs from the source and without any negative arcs. Moreover if $f(0, \ldots, 0)$ equals 0 then there exists a size–minimal $\oplus$AOBDD with at most one unlabelled arc. If $f(0, \ldots, 0)$ equals 1 then one of the unlabelled arcs leads from the source to the sink.*

Notice that in the same way we could prove the following theorem:

Theorem 3.5 *Let* B *be the size–minimal shared $\oplus$AOBDD representing functions $f_1, \ldots, f_k$. Then its size equals:*

$$k + \dim_{F_2} \text{span } \cup_{i=1}^{k} \triangle^\star f_i$$

Note that the functions in span $\triangle^\star f$ need not to be expressible as the linear combinations of functions represented by the vertices of the diagram. The just proven theorems give only the formula for the size of the size–minimal diagram, they do not relate the functions represented by the vertices of the diagram to functions in span $\triangle^\star f$.

The following theorem is proved in [6]:

Theorem 3.6 *Let* B *be the size–minimal shared $\oplus$OBDD representing functions $f_1, \ldots, f_k$. Then its size equals:*

$$k + \dim_{F_2} \text{span } \cup_{i=1}^{k} \diamond f_i$$

Note that span $\triangle^\star f \subseteq$ span $\diamond f$. Thus the following corollary holds:

Corollary 3.2 *The size of a size–minimal $\oplus$AOBDD representing functions $f_1, \ldots, f_k$ is at most the size of a size–minimal $\oplus$OBDD representing the same set of functions.*

The relation between the size of a size–minimal $\oplus$AOBDD and a size–minimal $\oplus$OBDD is described in the next theorem. All the $\oplus$(A)OBDDs considered since now to the end of the section may contain both positive and negative arcs. Let us first prove the following lemma:

Lemma 3.3 *Let $h_n(x_1, \ldots, x_n) = x_1 \oplus \ldots \oplus x_n$. Every $\oplus$OBDD representing function h_n has at least $2n$ arcs.*

Proof: Suppose the existence of a $\oplus$OBDD representing h_n with at most $2n-1$ arcs and select of them B_0 with the most arcs leading from the source.

Let D_i be the sum of the outdegrees of vertices labelled with x_i. Clearly $\sum_{i=1}^n D_i \leq 2n-2$ and $D_i \geq 1$. Choose i_0 such that $D_{i_0} = 1$ and there is no arc leading from the source to the (only) vertex labelled with x_{i_0} — it must exist: otherwise there is for all i such that $D_i = 1$ an arc from the source to the vertex labelled with x_i, thus the outdegree of the source is at least $|\{i | D_i = 1\}|$ and the sum of all outdegrees in B_0 is at least $2n$. Let v_0 denote the vertex labelled with x_{i_0}. Remove v_0 (with all adjacent arcs) from B_0 and get a $\oplus$OBDD with at most $2n-3$ arcs representing either h_{n-1} or $h_{n-1} \oplus 1$ (a function of the variables without x_{i_0}). Now add a vertex w, an arc from the source to w and an arc from w to the sink labelled as the only arc leading from v_0 in the original $\oplus$OBDD. The constructed $\oplus$OBDD represents h_n and has at most $2n-1$ arcs and more arcs from the source than the original one — the contradiction to the choice of B_0. ∎

Notice that in the proof of this (rather technical) lemma we did not use that the parity binary decision diagram was ordered.

Theorem 3.7 *Let f be an arbitrary Boolean function.*

1. *Let B_1 be an arbitrary $\oplus$OBDD representing the function f. Then there exists a $\oplus$AOBDD B_2 with at most the same number of vertices as B_1 has and with at most the same number of arcs as B_1.*
2. *Let B_1 be an arbitrary $\oplus$AOBDD representing the function f. Then there exists a $\oplus$OBDD B_2 with at most $O(n)$–times more vertices than B_1 has and with at most $O(1)$–times more arcs than B_1.*

All bounds given in this theorem are assymptotically sharp.

Proof: The first part of the theorem is clear since $\oplus$AOBDDs generalize $\oplus$OBDDs. The sharpness is witnessed by function $g(x_1, \ldots, x_n) = x_1 \vee \ldots \vee x_n$. Let $g_k(x_1, \ldots, x_n) = x_k \vee \ldots \vee x_n$. Then set $\Diamond g$ is equal to $\{g_1, \ldots, g_n, I, 0\}$ and the set Δg is equal to $\{I \oplus g_2, \ldots, I \oplus g_n, I\}$. Thus it is clear that span $\Diamond g$ = span $\Delta^\star g$ and the sizes of the minimal $\oplus$AOBDD for g and the minimal $\oplus$OBDD for g are equal (see Theorem 3.4 and Theorem 3.6).

In order to prove the second part of the theorem we show how to construct a $\oplus$OBDD B_2 with at most n–times more vertices and at most three–times more arcs than a $\oplus$AOBDD B_1 has. Let v be a vertex of B_1 and $x_{i_1}, \ldots, x_{i_k}$ be variables used in labels of arcs leading from v. Create new vertices $v^1, \ldots, v^k$ and redirect arcs leading to v to vertex v^1 (preserving their labels). Join v^1 to each of vertices $v^2, \ldots, v^k$ with the pair of arcs, one labelled with x_{i_1} and 0 and the other labelled with x_{i_1} and 1. Then we make leading from v^j the arcs from v labelled with x_{i_j}. Now we remove the vertex v from the digraph (there are no arcs leading either from or to v at the present). Notice that the represented function has not changed during the process. If we apply the above process to all vertices of B_2 then we obtain a $\oplus$OBDD with at most $O(n)$–times more vertices and at most $O(1)$–times more arcs.

The sharpness is witnessed by function $h(x_1, \ldots, x_n) = x_1 \oplus \ldots \oplus x_n$. The size of the minimal $\oplus$AOBDD is clearly 3 and it contains $n+1$ arcs; the size of the minimal $\oplus$OBDD is $n+2$ and it contains $2n$ arcs (see Lemma 3.3). ∎

4 Parity AOBDD uniqueness properties

In order to formulate the uniqueness conditions let us define the *PDFS*[2] *algorithm for parity AOBDDs*: The algorithm is the usual graph depth–first

[2]P states for priority

search algorithm started from the source of the $\oplus$AOBDD with one additional rule: if there are more possibilities to select an arc to continue through it always continues through the arc with the π–greatest variable and it prefers an arc leading to the sink among all such arcs. We call the (rooted) tree with labelled arcs produced by the algorithm *PDFS–tree*. Notice that there might exist more different PDFS–trees for the same parity AOBDD. We call a parity (A)OBDD *linearly reduced* if functions represented by its vertices different from the source are linearly independent. We say that a parity AOBDD satisfies *the uniqueness conditions* if it satisfies the following four conditions:

- It is linearly reduced.
- It contains no negative arcs.
- It contains at most one unlabelled arc to a non–sink vertex and at most one to the sink; in particular the degree of its source is at most two.
- Its PDFS–tree is unique.

The ordering of the vertices and the tree–arcs of an acyclic digraph *induced by (P)DFS* is the ordering induced by the pre–order listing of its vertices and tree–arcs, i.e. a vertex or a tree–arc precedes all vertices and tree–arcs first visited by the (P)DFS algorithm after it. It is clear that after each arc immediately follows the vertex to which it leads. The sequence of vertices induced by (P)DFS is the sequence of all vertices ordered consisently to the ordering induced by (P)DFS.

We study the properties of uniqueness conditions for parity AOBDDs in the following four theorems.

Theorem 4.1 *For each Boolean function f there exists a π–$\oplus$AOBDD which satisfies the uniqueness conditions.*

The proof of this theorem is postponed to the section 6.

Theorem 4.2 *Any π–$\oplus$AOBDD* B *which satisfies the uniqueness conditions is size–minimal.*

Proof: Let f be the function represented by B and $v_0, \ldots, v_m$ be the sequence of B's vertices induced by PDFS; note that v_0 is the source. We prove by induction on i that $f_{v_i} \in \text{span }\ \triangle^\star f$. If there are two unlabelled arcs in B then v_1 is the sink and the function represented by v_2 is $f + I$. If there is only one unlabelled arc in B then the function represented by v_1 is f itself. Thus $v_i \in \text{span }\ \triangle^\star f$ holds for vertices visited through an unlabelled arc. Let v_i be a vertex visited through an arc labelled with x_j and let v_k $(k < i)$ be its tree–parent. If v_i is the sink then it is clear that $v_i \in \text{span }\ \triangle^\star f$. Otherwise, from the induction hypothesis we know that $f_{v_k} \in \text{span }\ \triangle^\star f$. Let W be all the vertices accessible from v_k by an arc labelled with x_j. The vertex v_i is the vertex with the greatest index among the vertices in W; if this was not the case then there would be more arcs labelled with x_j leading to unvisited vertices when the algorithm continued through the arc to v_i and the PDFS–tree of B would not be unique. From the facts that[3] (see Lemma 3.1) $\triangle_j f_{v_k} = \bigoplus_{w \in W} f_w$, $f_{v_k} \in \text{span }\ \triangle^\star f$ and $f_w \in \text{span }\ \triangle^\star f$ for all $w \in W, w \neq v_i$ it follows that $f_{v_i} = \triangle_j f_{v_k} \oplus \bigoplus_{w \in W, w \neq v_i} f_w$ and thus $f_{v_i} \in \text{span }\ \triangle^\star f$.

From the just proven fact that all the functions represented by vertices of B are in span $\triangle^\star f$ and from the fact that B is linearly reduced we conclude that B is size–minimal (see Theorem 3.4). ∎

There can be more $\oplus$–AOBDDs satisfying the uniqueness conditions representing the same boolean function; however they have in some sense the same structure; this is stated and proven in the following theorem.

[3]Remember that $\triangle_j$ fix variables π–smaller or equal to x_j.

Lemma 4.1 *Let $F \subseteq G$ be sets of Boolean functions, f_1,f_2 functions which are not contained in F,* $\text{span}\,\{f_1\} \cup F = \text{span}\,\{f_2\} \cup F$ *and let* $\text{span}\,F$ *be closed under all operations $\triangle_k, \ldots, \triangle_n$. Then* $\text{span}\,G \cup \{\triangle_k f_1\} = \text{span}\,G \cup \{\triangle_k f_2\}$.

Proof: Clearly $f' = f_1 \oplus f_2 \in \text{span}\,F$ and thus $\triangle_k f' = (\triangle_k f_1) \oplus (\triangle_k f_2) \in F$ (F is closed under $\triangle_k$). The lemma immediately follows. ■

Theorem 4.3 *Let* B_1 *and* B_2 *be two π–$\oplus$AOBDDs which satisfy the uniqueness conditions and represent the same function f. Let $v_0^1, v_1^1, \ldots, v_m^1$ be the vertices of* B_1 *and $v_0^2, v_1^2, \ldots, v_m^2$ be the vertices of* B_2 *in the ordering induced by PDFS*[4]*. Then their PDFS–trees are isomorphic and it holds for each $1 \leq k \leq m$:*

$$\text{span}\,\{f_{v_i^1}, 1 \leq i \leq k\} = \text{span}\,\{f_{v_i^2}, 1 \leq i \leq k\}$$

Proof: We prove by the induction on i that the PDFS–subtrees induced by $v_0^1, \ldots, v_i^1$ and $v_0^2, \ldots, v_i^2$, i.e. subtrees containing the tree–arcs leading only between $v_1^1, \ldots, v_i^1$ and $v_1^2, \ldots, v_i^2$, are isomorphic and $\text{span}\,\{f_{v_1^1}, \ldots, f_{v_i^1}\} = \text{span}\,\{f_{v_1^2}, \ldots, f_{v_i^2}\}$.

Let a be the next arc of the PDFS–tree of B_1. If the next arc of the PDFS–tree of B_2 does not correspond to a, w.l.o.g. we assume that a leads from the vertex with a greater index than the next PDFS–tree arc in B_2 and if they both lead from the vertex with the same index then a is labelled with a π–greater variable. Let j be the index of the vertex a leads from and x_k the variable it is labelled with. Both functions $f_{v_j^1}$ and $f_{v_j^2}$ are in the following set (because the $\oplus$AOBDDs are linearly reduced):

$$\text{span}\,\{f_{v_1^1}, \ldots, f_{v_j^1}\} \setminus \text{span}\,\{f_{v_1^1}, \ldots, f_{v_{j-1}^1}\} =$$

$$\text{span}\,\{f_{v_1^2}, \ldots, f_{v_j^2}\} \setminus \text{span}\,\{f_{v_1^2}, \ldots, f_{v_{j-1}^2}\}$$

Both $\text{span}\,\{f_{v_1^1}, \ldots, f_{v_{j-1}^1}\} = \text{span}\,\{f_{v_1^2}, \ldots, f_{v_{j-1}^2}\}$ are closed under all operations $\triangle_k, \ldots, \triangle_n$ because there are no arcs labelled with $x_{\pi(k)}, \ldots, x_{\pi(n)}$ leading from $v_1^1, \ldots, v_{j-1}^1$ and $v_1^2, \ldots, v_{j-1}^2$ to the vertices in the ordering induced by PDFS greater or equal to v_j^1 or v_j^2, i.e. to the vertices first visited after v_j^1 and v_j^2. To see this use the priority rule of the PDFS algorithm to the vertices on the path from v_0^1 to v_j^1 and on the path from v_0^2 to v_j^2. Thus due to Lemma 4.1 ($F = \{f_{v_1^1}, \ldots, f_{v_{j-1}^1}\} = \{f_{v_1^2}, \ldots, f_{v_{j-1}^2}\}$, $f_1 = f_{v_j^1}$, $f_2 = f_{v_j^2}$, $G = \{f_{v_1^1}, \ldots, f_{v_{i-1}^1}\} = \{f_{v_1^2}, \ldots, f_{v_{i-1}^2}\}$):

$$\text{span}\,\{f_{v_1^1}, \ldots, f_{v_{i-1}^1}, \triangle_k f_{v_j^1}\} = \text{span}\,\{f_{v_1^2}, \ldots, f_{v_{i-1}^2}, \triangle_k f_{v_j^2}\}$$

For the contradiction assume that the PDFS–trees differ at a, i.e. the arc corresponding to a is missing in B_2 (due to our assumption and the priority rule of the PDFS algorithm). The function $\triangle_k f_{v_j^1}$ is not expressible as a sum of functions represented by vertices of B_1 PDFS–smaller than a (i.e. visited before it) because B_1 is linearly reduced. Since in B_2 all the edges labelled with x_k from v_j^2 lead to the vertices smaller than v_j^2, this function is expressible as a sum of functions represented by corresponding vertices of B_2 but the linear spans of functions represented by them are equal in B_1 and B_2 — the contradiction. ■

Unfortunately the equality of the linear spans does not imply that $f_{v_i^1} = f_{v_i^2}$ for all $1 \leq i \leq m$; one can find non–isomorphic parity AOBDDs representing the same function which both satisfy the uniqueness conditions.

[4]The numbers of the vertices are equal due to Theorem 4.2.

5 Parity OBDD uniqueness properties

We define *the PDFS algorithm for parity OBDDs* first. The algorithm is the usual graph depth–first search algorithm started from the source of the $\oplus$OBDD with one additional rule: if there are more possibilities to select an arc to continue through it always continues through the arc leading to the vertex labelled with the π–greatest variable; if there is an arc leading to the sink it prefers this arc to any other. We call the (rooted) tree with labelled vertices produced by the algorithm *PDFS–tree*. As in case of parity AOBDDs there might exist more different PDFS–trees for the same parity OBDD.

We say that a parity OBDD satisfies *the uniqueness conditions* if it satisfies the following four conditions:

- It is linearly reduced.
- It contains no negative arcs.
- Its PDFS–tree is unique.
- If there is a tree arc leading from its vertex v to the vertex labelled with x_i then there are no other arcs leading from v to any vertex labelled with x_i.

Notice that there can be either exactly one tree arc leading from v to a vertex labelled with x_i (and no other arcs) or any number of non-tree arcs leading from v to vertices labelled with x_i.

Theorem 5.1 *For each Boolean function f there exists a π–$\oplus$OBDD representing f which satisfies the uniqueness conditions.*

Proof: As in case of π–$\oplus$AOBDDs we could postpone the proof to the section 6 but it is useful to present also the following non–algorithmic proof.

W.l.o.g. assume π is an identity. First we create a sequence $f_1, \ldots, f_{2^{n+1}-1}$ of (not necessarily distinct) Boolean functions. We define the operation $\square^i$ $(1 \leq i \leq n+1)$ as follows:

- For $1 \leq i \leq n$ let $\square^i f = \square^{i+1} f^0, f^0 \oplus f, \square^{i+1}(f^0 \oplus f^1)$ where f^0 is obtained from f by fixing[5] x_i to zero and f^1 by fixing it to one.
- For $i = n+1$ let $\square^{n+1} f = f$.

We abbreviate $\square^1 f$ to $\square f$. We create the basis $g_1, \ldots, g_m$ of span $\square f =$ span $\diamond f$ (see Theorem 3.6 and Lemma 5.2) from sequence $\square f$ by greedy algorithm, i.e. we take the first non–trivial function in $\square f$ and then (repeatedly) extend the basis with the first linearly independent function until we obtain the basis of $\diamond f$. We say the function in the sequence $\square f$ was *created by the operation* $\square^k$ if it was added to the sequence directly by the operation $\square^k$, i.e. not by the operation $\square^{k+1}$ in the definition of $\square^k$. Note that the definition of $\square^k$ guarantees that the function created by $\square^k$ does not essentially depend on the first $k-1$ variables.

Let us state some properties of the just defined operation and the selected basis in the following lemmas:

Lemma 5.1 *Consider the set Z of all $\{0, 1, \star\}$–n-tuples $(z_1, \ldots, z_n)$ for which $z_i = \star$ implies $z_j = \star$ for all $j \geq i$. Let $f[z]$ for $z \in Z$ be a subfunction of f obtained by fixing all variables x_i corresponding to entries different from $\star$ to z_i[6]. Define the ordering $\prec$: $0 \prec \star \prec 1$; order Z lexicographically with respect to $\prec$ and let $f'_1, \ldots, f'_{2^{n+1}-1}$ be the resulting sequence of functions. Let $\square f = f_1, \ldots, f_{2^{n+1}-1}$. Then the following holds for every $1 \leq k \leq 2^{n+1}-1$:*

$$\text{span}\ \{f_1, \ldots, f_k\} = \text{span}\ \{f'_1, \ldots, f'_k\}$$

[5]The resulting function is a subfunction of f, i.e. the function of the same number of variables as f which does not esentially depend on the fixed variables.

[6]Note that $\diamond f = \{f[z], z \in Z\}$.

Proof: The proof proceeds by the induction on n (the number of variables). For $n = 1$ the lemma is clear since $\Box^1 f = f(0), f(x) - f(0), f(1) - f(0)$. Suppose $n > 1$ now. The equality of linear spans for $k < 2^n$ follows from the application of the induction hypothesis to the restriction of $f(0, x_2, \ldots, x_n)$ to the last $n-1$ variables. The function $f_{2^n} = f(x_1, \ldots, x_n) \oplus f(0, x_2, \ldots, x_n)$. Thus the equality holds also for $k = 2^n$ because of $f[0, \star, \ldots, \star] = f(0, x_2, \ldots, x_n) \in$ span $\{f_1, \ldots, f_{2^n}\}$. The equality below follows from the application of the induction hypothesis to the restriction of $g(x_2, \ldots, x_n) = f(0, x_2, \ldots, x_n) \oplus f(1, x_2, \ldots, x_n)$ to the last $n - 1$ variables ($k > 2^n$); all the function $f[z]$ with $z_1 = 0$ are among the functions $f'_1, \ldots, f'_{2^n-1}$:

$$\text{span } \{f_{2^n+1}, \ldots, f_k\} = \text{span } \{g[0, \ldots, 0, 0], g[0, \ldots, 0, \star], g[0, \ldots, 0, 1], \ldots\} =$$

$$\text{span } \{f[0, 0, \ldots, 0, 0] \oplus f[1, 0, \ldots, 0, 0], f[0, 0, \ldots, 0, \star] \oplus f[1, 0, \ldots, 0, \star],$$
$$f[0, 0, \ldots, 0, 1] \oplus f[1, 0, \ldots, 0, 1], \ldots\}$$

Since the equality of the linear spans holds also for $k > 2^n$. ■

We use notation $\Box f = f_1, \ldots, f_{2^{n+1}-1}$ in the rest of the proof of Theorem 5.1.

The corollary of Lemma 5.1 is the following lemma:

Lemma 5.2

$$\text{span } \Diamond f = \text{span } \{\Box f\}$$

Lemma 5.3 *If $f_j \in \Box f$ is non–constant and x_i is the first variable on which f_j essentially depends then $f_j(x_1, \ldots, x_n)$ is of the form $x_i \wedge h(x_1, \ldots, x_n)$ where h is an n-variable Boolean function which does not essentially depend on the first i variable. Moreover, the function f_j was created by $\Box^i$ and the function h is uniquely determined and satisfies:*

$$h = \triangle_i f_j$$

Proof: Let f_j be created by $\Box^k$. Because f_j essentially depends on x_i it follows that $k \leq i$. Since $k \leq n$, the definition of $\Box^k$ implies that $f = g^0 \oplus g$ where g^0 is obtained from g by fixing x_k to zero. If g did not essentially depend on x_k then $g^0 \oplus g$ would be the all–zero function because of $g = g^0$. Hence g essentially depends on x_k and $g^0 \oplus g$ also essentially depends on x_k and thus $i = k$.

If $x_k = 0$ then the function $f_j = g^0 \oplus g$ is clearly zero. Hence $f_j = x_i \wedge (g^0 \oplus g) = x_i \wedge (g^0 \oplus g^1)$ (remember $k = i$) where g^1 is obtained from g by fixing x_k to one. Now, the uniqueness and the equality $h = \triangle_i f_j$ follows from Lemma 3.2. ■

Lemma 5.4 *If h does not essentially depend on its first k variables then* span $\Diamond h =$ span $\Box^{k+1} h$ *and $\Box^{k+1} h$ contains in particular the following functions (ordered by their appearance in the sequence): $h(0, \ldots, 0)$, $x_n \wedge \triangle_n h$, $x_{n-1} \wedge \triangle_{n-1} h$, …, $x_{k+1} \wedge \triangle_{k+1} h$.*

Proof: The proof proceeds by the induction on k; we start with $k = n$ and continue to $k = 0$. The lemma is clear for $k = n$.

Suppose $k < n$ now. Let h' be the function obtained from h by fixing x_{k+1} to zero; clearly $h(0, \ldots, 0) = h'(0, \ldots, 0)$ and $\triangle_l h = \triangle_l h'$ for $l > k + 1$. The functions $h(0, \ldots, 0)$ and $x_l \wedge \triangle_l h$ (for $l > k + 1$) are contained (in this order) in the sequence $\Box^{k+2} h'$ due to the induction hypothesis. The function $h - h' = x_{k+1} \wedge \triangle_{k+1} h$ is contained in the sequence $\Box^{k+1} h$ after all the other above mentioned functions due to the definition of $\Box^{k+1}$.

The equality span $\Diamond h =$ span $\Box^{k+1} h$ follows by the application of Lemma 5.2 to the restriction of h to the last $n - k$ variables. ■

Lemma 5.5 *If $h \in \operatorname{span} \Diamond f$ and h does not essentially depend on its first k variables then it can be expressed*[7] *as a combination of $g_1, \dots, g_m$ not essentially depending on their first k variables.*

Proof: Let $h = \bigoplus_{i \in I} g_i$ and $I' \subseteq I$ be the indices of functions g_i essentially depending on some of variables $x_1, \dots, x_k$. Since h does not essentially depend on any of $x_1, \dots, x_k$ then fixing these variables to zero does not affect h. But all the functions g_i, $i \in I'$, are all-zero functions after fixing these variables to zero (Lemma 5.3) and thus $h = \bigoplus_{i \in I \setminus I'} g_i$. Because $g_1, \dots, g_m$ is a basis of span $\Diamond f$ it follows $I' = \emptyset$. ∎

Lemma 5.6 *If function $f_j \in \Box f$ was not selected to the basis and x_k is the first variable on which f_j essentially depends then f_j is expressed as a combination of functions, preceding it in $\Box f$, whose first variables on which they essentially depend are all x_k.*

Proof: Let $f_j = \bigoplus_{i \in I} g_i$ where for all $i \in I$ holds $i < j$; this is possible because f_j was not selected to the basis. Let $I' \subseteq I$ be the indices of functions g_i essentially depending only on variables $x_{k+1}, \dots, x_n$. No function g_i, $i \in I$, essentially depends on any of variables $x_1, \dots, x_{k-1}$ (Lemma 5.5). Fix x_k to zero; then f_j and $g_i, i \in I \setminus I'$, are the all–zero functions (Lemma 5.3). But the fixing x_k to zero does not affect any function g_i, $i \in I'$, and from $f_j = \bigoplus_{i \in I} g_i$ it follows that $\bigoplus_{i \in I'} g_i$ is the all–zero function and thus $f_j = \bigoplus_{i \in I \setminus I'} g_i$. Because $g_1, \dots, g_m$ is a basis of span $\Diamond f$ it follows $I' = \emptyset$. ∎

Let us continue the proof of Theorem 5.1 and construct the desired parity OBDD now. We call its vertices $v_0, \dots, v_m$; v_0 is the source and $v_1, \dots, v_m$ are vertices representing $g_1, \dots, g_m$. If g_k $(1 \leq k \leq m)$ was created by $\Box^i$ then it does not essentially depend on the first $i-1$ variables and if g_k is not constant then it essentially depends on the i-th variable (see Lemma 5.3). If g_k is constant then it must be the all–one function, otherwise it would not have been chosen to the basis, and v_k is the sink. If g_k is not constant then $g_k = x_i \wedge \left(\left(\bigoplus_{l=i+1}^{n} x_l \wedge \triangle_l \triangle_i g_k\right) \oplus (\triangle_i g_k)(0, \dots, 0)\right)$ due to Lemma 3.2 and Lemma 5.3. From Lemma 5.4 it follows that $\Box^{i+1} \triangle_i g_k$ (the sequence of functions immediately following g_k) contains all the following functions: $(\triangle_i g_k)(0, \dots, 0)$, $x_n \wedge \triangle_n \triangle_i g_k, \dots, x_{i+1} \wedge \triangle_{i+1} \triangle_i g_k$. The vertex v_k is labelled with x_i and the arcs from it lead to the vertices representing $(\triangle_i g_k)(0, \dots, 0)$,$x_n \wedge \triangle_n \triangle_i g_k, \dots, x_{i+1} \wedge \triangle_{i+1} \triangle_i g_k$. If the function itself is not represented by a vertex there are arcs leading from v_k to the set of vertices representing the corresponding linear combination. All the new arcs are labelled by ones. It is a straightforward use of Lemma 3.2 and Lemma 5.3 (these lemmas imply the already mentioned equality $g_k = x_i \wedge \left(\left(\bigoplus_{l=i+1}^{n} x_l \wedge \triangle_l \triangle_i g_k\right) \oplus (\triangle_i g_k)(0, \dots, 0)\right)$) to check that v_k represents g_k.

The resulting digraph is really a π-$\oplus$OBDD, i.e. we have not violated the ordering constraint during its construction: if x_i is the first variable on which g_k essentially depends then v_k is is labelled with x_i (due to the way of the construction); if $f_j \in \Box f$ was not selected to the basis and x_i is the first variable on which it essentially depends, it can be expressed as a linear combination of some functions $g_1, \dots, g_m$ but for each of them the first variable which it essentially depends on is x_i (Lemma 5.6) and thus the vertices corresponding to them are labelled with x_i — i.e. the arcs from the vertex labelled with x_i lead to the vertices labelled with x_j, $j > i$.

It remains to prove that the constructed parity OBDD satisfies the uniqueness conditions. It is obviously linearly reduced and it contains no negative arcs. Let P_k be the set of all dipaths from the source to v_k; assign to each dipath the string of indices of the variables assigned to the vertices on the

[7] Since $g_1, \dots, g_m$ is a basis of span $\Diamond f$ h is uniquely expressible as a combination of $g_1, \dots, g_m$.

dipath and assign to each vertex v_k the lexicographically greatest string p_k from set P_k; we consider that the prefix of any string is greater than the string itself. It is a straightforward work (using the way in which the parity OBDD was constructed) to check that $p_1, \ldots, p_m$ is a decreasing sequence of strings; the vertex to which is assigned a lexicographically greater string is visited by the PDFS algorithm before the vertex with a smaller one. Thus the vertices are visited by the PDFS algorithm in the order of increasing indices $(v_0, v_1, v_2, \ldots, v_m)$. Now, suppose for the contradiction that the PDFS–tree is not unique - that is there is a vertex v_k (labelled with x_i) representing g_k where the algorithm can choose an arc to continue through from several arcs; all these arcs lead to the vertices labelled with the same variable (say x_j) and due to Lemma 3.2, Lemma 5.3 and Lemma 5.6 the sum of the functions they represent is $x_j \wedge \triangle_j \triangle_i g_k$. Function $x_j \wedge \triangle_j \triangle_i g_k$ in the sequence created by $\Box^{i+1}$ immediately following g_k was not included to the basis of span $\Diamond f$ (if it was included the corresponding vertex is the only vertex labelled with x_j to which an arc from v_k leads) and thus can be expressed as a linear combination of other functions (let G' be the set of these functions) preceding $x_j \wedge \triangle_j \triangle_i g_k$ in $\Box f$. Due to Lemma 5.3 and Lemma 5.6 the first variable on which any function in G' essentially depend is x_j. But because of the definition of $\Box$ no function between g_k and $x_j \wedge \triangle_j \triangle_i g_k$ essentially depends on x_j (consider the position of $x_j \wedge \triangle_j \triangle_i g_k$ in the sequence $\Box f$ as discussed in the proof of Lemma 5.4) and thus all functions in G' precede the function g_k in $\Box f$. But that means that all the vertices labelled with x_j to which an arc from v_k leads were already visited — the contradiction. The last uniqueness condition immediately follows from the above discussion and Lemma 5.6. ∎

Let turn our attention to properties of any π–$\oplus$OBDDs satisfying the uniqueness conditions.

Theorem 5.2 *Each π–$\oplus$OBDD* B *which satisfies the uniqueness conditions is size–minimal.*

Proof: Let f be the function represented by B and $v_1, \ldots, v_m$ be the sequence of B's vertices induced by PDFS; note that v_1 is the source. We prove by induction on i that $f_{v_i} \in$ span $\Diamond f$. For $i = 1$ the statement is trivial since v_1 is the source and $f_{v_1} = f$. Suppose i is greater than 1. Let v_k be the tree–parent of v_i (in particular, $k < i$). We distinguish several cases:

- The vertex v_k is the source and v_i is the sink. All the arcs are positive and thus $f_{v_k}(0, \ldots, 0)$ is equal to 1. Thus the all–one function f_{v_i} is in span $\Diamond f$.
- The vertex v_k is not the source and v_i is the sink. Let v_k be labelled with $x_{k'}$. All the arcs are positive and thus $f_{v_k}(x_1, \ldots, x_n)$ is equal to 1 for $x_{k'} = 1$ and $x_l = 0, l \neq k'$. Thus the all–one function f_{v_i} is in span $\Diamond f$.
- The vertex v_k is the source and v_i is not the sink. Let v_i be labelled with $x_{i'}$. Note that v_i is the only son of the source (v_k) labelled with $x_{i'}$. Let W be the set of all the sons of the source (v_k) labelled with variables π–greater than $x_{i'}$ including the sink if it is the son of the source (v_k). Due to the induction hypothesis $\{f_w; w \in W\} \subseteq$ span $\Diamond f$. Let f' be the function obtained from $f = f_{v_k}$ by fixing all the variables strictly π-smaller than $x_{i'}$ to zero. Clearly $f' \in$ span $\Diamond f f$ and $f' = f_{v_i} \oplus \bigoplus_{w \in W} f_w$ and thus also $f_{v_i} \in$ span $\Diamond f$.
- Neither v_k is the source nor v_i is the sink. Let v_k and v_i be labelled with $x_{k'}$ and $x_{i'}$. Note that v_i is the only son of v_k labelled with $x_{i'}$. Let W be the set of all the sons of v_k labelled with variables π–greater than $x_{i'}$ including the sink if it is the son of v_k. Due to the induction hypothesis $\{f_w; w \in W\} \subseteq$ span $\Diamond f$. Let f' be the function obtained from f_{v_k} by fixing $x_{k'}$ to one and all the variables strictly π-smaller

than $x_{i'}$ except for $x_{k'}$ to zero. Clearly $f' \in \text{span} \diamond f_{v_k} \subseteq \text{span} \diamond f$ and $f' = f_{v_i} \oplus \bigoplus_{w \in W} f_w$ and thus also $f_{v_i} \in \text{span} \diamond f$.

From the just proven fact that all the functions represented by vertices of B are in span $\diamond f$ and from the fact that B is linearly reduced we conclude that B is size–minimal (Theorem 3.6). ∎

Theorem 5.3 *Let* B_1 *and* B_2 *two* π*–*$\oplus$*OBDDs which satisfy the uniqueness conditions and represent the same function* f*. Then their PDFS–trees and the diagrams themselves are isomorphic.*

Proof: Let $v_0^1, \ldots, v_m^1$ be the sequence of B_1's vertices induced by PDFS and $v_0^2, \ldots, v_m^2$ be the sequence of B_2's vertices induced by PDFS; the numbers of the vertices are equal due to Theorem 5.2. By the induction of i we prove that the PDFS–subtrees induced by $v_0^1, \ldots, v_i^1$ and $v_0^2, \ldots, v_i^2$ are isomorphic and $f_{v_i^1} = f_{v_i^2}$.

Let a be the next arc of the PDFS–tree of B_1. If the next arc of the PDFS–tree of B_2 does not correspond to a, w.l.o.g. we assume that a leads from the vertex with a greater index than the next PDFS–tree arc in B_2 and if they both lead from the vertex with the same index then a leads to the vertex testing a π–greater variable. Let j be the index of the vertex a leads from, x_k the variable tested by v_j^1 and x_l the variable tested by the vertex to which a leads. Both functions $f_{v_j^1}$ and $f_{v_j^2}$ are due to the induction hypothesis equal. The function represented by the vertex to which a leads is due to Lemma 3.2 $x_l \wedge \triangle_l \triangle_k f_{v_j^1}$. For the contradiction assume that PDFS–trees differ at a, i.e. the corresponding arc to a is missing in B_2 (due to our assumption and the priority rule of the PDFS algorithm). The function $x_l \wedge \triangle_l \triangle_k f_{v_j^1}$ is expressible as a sum of functions represented by vertices of B_1 PDFS–smaller than a (i.e. visited before it) because B_1 is linearly reduced. But this function is expressible as a sum of functions represented by corresponding vertices of B_2 but the functions represented by them are equal — the contradiction.

We have proved that the functions represented by corresponding vertices (v_i^1 and v_i^2) in B_1 and B_2 are equal. Because B_1 and B_2 are linearly reduced they are isomorphic. ∎

Both the PDFS algorithm for $\oplus$AOBDDs and $\oplus$OBDDs can be modified to shared $\oplus$AOBDDs and shared $\oplus$OBDDs representing functions $f_1, \ldots, f_k$ using the following procedure: We create a new vertex, call it the supersource, and add arcs leading from it to all its k sources. The PDFS algorithm starts from the supersource, first visiting the source representing f_1, then the source representing f_2 etc. During the search it applies all its rules. After removing the supersource we obtain *PDFS–forest.* The uniqueness conditions for shared $\oplus$AOBDDs and $\oplus$OBDDs remain unchanged; of course we consider a PDFS–forest instead of a PDFS–tree. In the same way the following two theorems can be proved:

Theorem 5.4 *For each sequence of Boolean functions there exists a shared* π*-*$\oplus$*AOBDD* B_0 *which satisfies the uniqueness conditions. Moreover,* B_0 *is size–minimal. The PDFS–forest of any other shared* π*-*$\oplus$*AOBDD* B *representing the same sequence of functions and satisfying the uniqueness conditions is isomorphic to the PDFS–forest of* B_0 *and the linear spans of functions represented by the first* k *vertices visited by the algorithm PDFS in* B_0 *and* B *are equal for all* k*.*

Theorem 5.5 *For each sequence of Boolean functions there exists a shared* π*-*$\oplus$*OBDD* B_0 *which satisfies the uniqueness conditions. Moreover,* B_0 *is size–minimal. Any other shared* π*-*$\oplus$*OBDD representing the same sequence of Boolean functions and satisfying the uniqueness conditions is isomorphic to* B_0*.*

Table 1: Running times for presented algorithms

Algorithm	Parity OBDDs	Parity AOBDDs
The storage size (T)	$O(S^2)$	$O(nS^2)$
Evaluation	$O(T)$	$O(T)$
Removal of negative–arcs	$O(S^3)$	$O(nS^3)$
Linear reduction	$O(S^3)$	$O(nS^3)$
Unification	$O(S^3)$	$O(nS^3)$
Minimization	$O(S^3)$	$O(nS^3)$
The PDFS algorithm	$O(T+n)$	$O(T+n)$

6 Algorithms

In this section we discuss basic algorithms for parity OBDDs and parity AOBDDs. They include: *Evaluation* — evaluation of the represented function for the given assignment of Boolean values to the variables, *Removal of negative–arcs* — modification of a $\oplus$OBDD ($\oplus$AOBDD) in order to get rid of all negative arcs (without enlarging its size), *Linear reduction* — modification of a $\oplus$OBDD ($\oplus$AOBDD) to a linearly reduced one, *Unification* — modification of a $\oplus$OBDD ($\oplus$AOBDD) to one which satisfies the uniqueness conditions and *PDFS algorithm* itself. The *Minimization* algorithm can be implemented by one call of Unification since the diagram which satisfies the uniqueness conditions is size–minimal (Theorem 4.2 and Theorem 5.2). The achieved running times of discussed algorithms are presented in Table 1. All discussed algorithms are easily adaptable to case of shared $\oplus$OBDDs and shared $\oplus$AOBDDs yielding the same running time.

Recall that S is the size of $\oplus$OBDD or $\oplus$AOBDD, i.e. the number of its vertices; the actual storage size of the diagram (T) is linear in the sum of the number of its vertices and its arcs.

6.1 Evaluation and the PDFS algorithm

Implementation of Evaluation using DFS approach in time $O(T)$ is trivial (see also [4], [6]). Implementation of the PDFS algorithm uses the bucket–sort algorithm and the usual DFS algorithm. The bucket–sort algorithm is used to sort all the arcs together according to the priority rule (this takes time $O(T+n)$) — let the resulting sequence of the arcs be $a_1,a_2,\ldots,a_k$. We store two lists of arcs at each vertex — one contains unmarked arcs, the other marked arcs. All arcs are unmarked at the beginning. We take the arcs in the order in the sequence $a_1,a_2,\ldots,a_k$; each time we move the arc a_i from the list of unmarked arcs to the end of the list of marked arcs of the vertex a_i leads from. This clearly takes time $O(T)$. At the end the lists of marked arcs contain all the arcs and moreover the lists are sorted according to the priority rule. The usual DFS algorithm is started and it visits the arcs according to their order in the lists of marked arcs (this clearly takes time $O(T)$). Testing violation of the uniqueness conditions is simple — it is enough to compare the first and the second unvisited arcs in the list. The whole algorithm runs in time $O(T+n)$.

6.2 Matrix representation of (A)OBDDs and Removal of negative–arcs

The remaining algorithms discussed in this section use *matrix represenation* of $\oplus$OBDDs and $\oplus$AOBDDs without negative arcs (see also [6]). A parity OBDD is represented by a $S\times S$ matrix whose rows and columns are indexed by vertices of the $\oplus$OBDD; its entry is one iff there is an arc leading from the row–vertex to the column–vertex in the $\oplus$OBDD, the other entries are zero.

A parity AOBDD is represented by a $S \times nS$ matrix whose rows are indexed by vertices of the $\oplus$AOBDD and columns are indexed by pairs of vertices of the $\oplus$AOBDD and variables; its entry is one iff there is an arc labelled with the column–variable leading from the row–vertex to the column–vertex in the $\oplus$AOBDD, the other entries are zero. *Extended matrix representation* of $\oplus$OBDDs and $\oplus$AOBDDs is similar to the matrix representation but entries of the matrix contain subsets of $\{0, 1, \star\}$; these sets represent the labellings of arcs between the corresponding vertices (and labelled with the appropriate variable in case of $\oplus$AOBDDs). Extended matrix represenation is used only in the algorithm Removal of negative–arcs. Transformation to and from the matrix representation and extended matrix representation is easily implementable in time $O(S^2)$ in case of $\oplus$OBDDs and $O(nS^2)$ in case of $\oplus$AOBDDs for all their used representations — it is enough to fill the matrix with zeroes in case of matrix representation or empty sets in case of extended matrix representation and then to take one diagram arc after another and modify appropriately the matrix; this clearly takes time $O(S^2+T)$ in case of $\oplus$OBDDs and $O(nS^2+T)$ in case of $\oplus$AOBDDs. The transformation in the oposite direction can be done by going through the whole matrix and adding arcs corresponding to non–zero (non–empty–set) matrix entries to the diagram.

The *rank* of a vertex of a $\oplus$OBDD is the variable which it is labelled with, the *rank* of a vertex of a $\oplus$AOBDD is the π-smallest variable labelled to any of arcs leading from it.

Let us turn our attention to the implementation of Removal of negative–arcs: First we create the extended matrix representation. We follow the ideas in the proofs of Theorem 3.1 and Theorem 3.2. We replace all arcs labelled by zero by a pair of arcs — one labelled with one and one labelled with $\star$ (admissible for all variable assignments). We produce (using the topological–sort algorithm) the ordering of its vertices $v_1, \ldots, v_n$ such that there is no arc from v_i to v_j for any $i > j$. Operation remove(e) for unlabelled arc e leading from w can be easily implemented in matrix representation of a $\oplus$OBDD ($\oplus$AOBDD) in running time $O(S)$ in case of $\oplus$OBDDs and $O(nS)$ in case of $\oplus$AOBDDs. As discussed in the proof of Theorem 3.1 newly created unlabelled arcs lead from the vertices preceding w in the ordering and do not violate the ordering condition. For each of $O(S)$ vertices there can be at most $O(S)$ (to each other vertex at most one) arcs labelled with $\star$ and thus operation remove is applied at most $O(S^2)$ times; this yields with the bound for the running time of remove the desired running time. Note that the algorithm does not affect the size (i.e. the number of vertices) of a $\oplus$(A)OBDD.

6.3 Operation reexpress

An essential operation for both algorithms Linear reduction and Unification is the operation *reexpress*(w, W) where w is a vertex (different from the source and the sink) of a $\oplus$(A)OBDD and W is a set of its vertices containing w. There must be neither the source nor the sink contained in W. In case of $\oplus$OBDDs we demand that the rank of all vertices in W is the same (i.e. it equals to the rank of w), in case of $\oplus$AOBDDs we demand that the rank of all vertices in W is equal or π-greater than the rank of w. Operation reexpress expects as input the matrix representation of a $\oplus$(A)OBDD without negative arcs. The goal of reexpress is to change the function represented by w to $\bigoplus_{u \in W} f_u$ and change the structure of the $\oplus$(A)OBDD in order not to affect either functions represented by the vertices different from w or the size of the $\oplus$(A)OBDD. The implementation of reexpress consist of two phases:

1. Duplicate all the arcs leading from vertices of W different from w and let the copies lead from w; remove pairs of identical arcs. In the matrix representation that means: Add (in the sense of F_2 addition) all rows corresponding to vertices of W different from w to the row representing w.

2. Let U be the set of vertices from which an arc leads to w. Create new arcs from each $u \in U$ to all vertices of W except for w. In case of $\oplus$AOBDDs label newly created arcs correspondingly to the arcs leading from u to w. Remove pairs of identical arcs if they arise. In the matrix representation that means: add (in the sense of F_2 addition) the column coresponding to w to the columns corresponding to $u \in W \setminus \{w\}$; in case of $\oplus$AOBDDs split the matrix to n parts with S columns each representing arcs labelled with the same variable and proceed in the same way in each of these parts.

Clearly, the new function represented by w is $\bigoplus_{u \in W} f_u$. It is a straightforward use of definition of $\oplus$(A)OBDDs to check that f_w is the only represented function affected by the whole operation. Because the rank of all vertices in W was the same (in case of $\oplus$AOBDDs the rank of w was the π–smallest) we did not violate the order constraint. Both the first phase and the second phase of the operation require running time $O(|W|S) = O(S^2)$ in case of $\oplus$OBDDs and $O(|W|nS) = O(nS^2)$ in case of $\oplus$AOBDDs. Thus the whole operation requires time $O(S^2)$ for $\oplus$OBDDs and $O(nS^2)$ for $\oplus$AOBDDs. Note that if a $\oplus$(A)OBDD is linearly reduced it is also linearly reduced after performing the operation reexpress.

6.4 Linear reduction

The presented algorithm follows the ideas of Waack's algorithm for linear reduction of $\oplus$OBDDs (see [6]) but we use the possibility to remove negative arcs from the $\oplus$(A)OBDD.

First we get rid of negative arcs by calling Removal of zero arcs. Then we sort the vertices according to their rank (using the bucket–sort algorithm) into a π–decreasing sequence — $v_1, \dots, v_n$; we find the first i such that span $\{f_{v_1}, \dots, f_{v_{i-1}}\} =$ span $\{f_{v_1}, \dots, f_{v_i}\}$ and express f_{v_i} as a linear combination of $f_{v_1}, \dots, f_{v_{i-1}}$: $f_{v_i} = \bigoplus_{w \in W} f_w$. Then we call reexpress$(v_i, W \cup \{v_i\})$ — clearly the new function represented by v_i is the all–zero function and thus it can be removed with all the arcs leading from and to v_i without affecting any of functions represented by the other vertices. But it is necessary to check that the call of reexpress$(v_i, W \cup \{v_i\})$ is legal, i.e. the rank constraint is satisfied. In case of $\oplus$AOBDDs this is trivial, in case of $\oplus$OBDDs this is true due to the following lemma:

Lemma 6.1 *Consider a $\oplus$OBDD which does not contain negative arcs and let $f_v = \bigoplus_{w \in W} f_w$, let $f_w, w \in W$, be linearly independent functions and let the rank of v be the π–smallest among the ranks of vertices in W. Then the rank of all vertices in W is equal to the rank of v.*

Proof: Let x_i be the rank of v. Let $W' \subseteq W$ be those vertices whose rank is π–greater than the rank v. If we fix to zero the variables π–smaller than or equal to x_i all functions f_v and $f_w, w \in W \setminus W'$, become the all–zero functions but any of functions $f_w, w \in W'$, does not change. Thus $\bigoplus_{w \in W'} f_w$ is the all–zero function and because $f_w, w \in W$, are linearly independent we have $W' = \emptyset$. ■

The following two lemmas play a key role in testing linear independence of represented functions:

Lemma 6.2 *Let W be a set of vertices of a $\oplus$OBDD of the same rank and let all the functions represented by vertices with the rank π–greater than the rank of vertices in W be linearly independent. Functions $f_w, w \in W$ are linearly independent iff their rows in the matrix representation are linearly independent.*

Lemma 6.3 *Let W be a set of vertices of a $\oplus$AOBDD, let $w_0 \in W$ be the vertex with the π–smallest rank in W and let all functions represented by* *vertices with the rank π–greater than the rank of w_0 be linearly independent. Functions $f_w, w \in W$ are linearly independent iff their rows in the matrix representation are linearly independent.*

The proof of Lemma 6.2 can be found in [6]; the proof of Lemma 6.3 can be done in the same way.

To check the linear independence we use the well–known Gaussian elimination procedure. We add rows corresponding to the vertices one by one to the matrix representing the $\oplus$AOBDD and check if its rank is full (see Lemma 6.3); in case of $\oplus$OBDDs we only check (see Lemma 6.2) that the rank of its submatrix representing the vertices with the same rank is full. We keep, in order to make the process of adding a new row fast enough, the already created matrix in the following form: The first one–entry in each row strictly precedes the first one–entry in the next row. The algorithm for adding new rows to the matrix and maintaining it in the described form is a standard linear algebra algorithm: Let r be the row to be added and let i be the coordinate of its first one–entry (if r is all–zero, r is not linearly independant with respect to matrix rows). If there is no row with the first one–entry coordinate equal to i we can insert r to the matrix to the appropriate position; in other case let r' be the row with the first one–entry coordinate equal to i. We continue adding the row $r \oplus r'$ instead of r now. Clearly, the span of the rows is the span of the rows of the input matrix and r and the rows of the matrix are linearly independent and the matrix is in the decribed form. The running time is linear in the size of the matrix for each row–addition.

The described algorithm modifies the new row by adding some other rows to it and then it inserts it to the matrix not necessarily as the last row. Let w be the vertex represented by the new row and let W be the vertices represented by rows added to it (if f_w is not linearly independent then W is the set of vertices representing the corresponding linear combination). The modifications of the matrix by the algorithm are the same as in the first phase of reexpress($w, W \cup \{w\}$) and we emulate them by the call of reexpress($w, W \cup \{w\}$); clearly the second phase of reexpress($w, W \cup \{w\}$) affect only rows still not added to the matrix (due to the order constraint in $\oplus$(A)OBDDs and the order in which the rows are added to the matrix). The call of reexpress($w, W \cup \{w\}$) is legal, i.e. w has the same rank as other vertices in W in case of $\oplus$OBDDs and w has the π–smallest rank among the vertices in W in case of $\oplus$AOBDDs because the rows are added to the matrix in π-decreasing order of the rank of the vertices corresponding to them. If the added row is not the last row we appropriately permute the matrix rows and columns which can be easily done in time linear in the matrix size. Thus the addition of one row needs time $O(S^2)$ in case of $\oplus$OBDDs and $O(nS^2)$ in case of $\oplus$AOBDDs. The whole Linear reduction algorithm requires time $O(S^3)$ for $\oplus$OBDDs and $O(nS^3)$ for $\oplus$AOBDDs.

6.5 Unification

The idea of Unification is simple — run the PDFS algorithm and change the structure of the $\oplus$(A)OBDD if the uniqueness conditions are violated. We call Removal of zero arcs and Linear reduction first, then we start the PDFS algorithm and continue running it until we encounter the first violation of the uniqueness conditions, that means we want to continue the PDFS algorithm from a vertex and there are:

- two or more unvisited vertices of the same rank and there are no unvisited vertices of π–greater rank, in case of $\oplus$OBDDs.
- two or more arcs labelled with the same variable (or unlabelled in case that we are in the source) leading to unvisited vertices different from the sink and there are no arcs labelled with a π–greater variable leading to unvisited vertices, in case of $\oplus$AOBDDs.

Let v be the vertex where the uniqueness conditions are violated, i.e. the vertex from which the algorithm PDFS cannot uniquely continue. In case of $\oplus$OBDDs, let W be the set of (both unvisited and visited) vertices of the same rank violating the uniqueness conditions to which an arc leads from v and let w be any unvisited vertex of W. In case of $\oplus$AOBDDs, let W be the set of unvisited vertices to which an arc leads from v and let w be any member of W with the π–smallest rank among vertices in W. Call reexpress(w, W). The operation reexpress clearly affects neither linear reduction of $\oplus$(A)OBDDs nor the already created PDFS–tree — arcs are modified only at the vertices from which an arc led or leads to or from w; w was an unvisited vertex and the new run of the PDFS algorithm would not include w to the PDFS–tree till it comes to the arc from v to w, thus the arcs from w do not affect the PDFS–tree; since no vertex in W is more attractive for the PDFS algorithm than w (in case of parity OBDDs it is due to the rank constraint) and w was an unvisited vertex, each $w \in W$ would be included to the PDFS–tree at the same point as previously and thus the structure of the PDFS–tree is not changed. The call of reexpress ensures there is no more any violation of the uniqueness conditions at the vertex v and the PDFS algorithm can continue through the arc to the vertex w. Unification ends when we have created the PDFS–tree of the whole $\oplus$(A)OBDD. Now remove all vertices not accessible from the source, i.e. not included to the PDFS–tree.

Since after each call of reexpress there is one vertex added to the PDFS–tree, there are at most $O(S)$ calls of reexpress and thus the whole running time of Unification (including preprocessing by Removal of zero arcs and Linear reduction) is $O(S^3)$ in case of $\oplus$OBDDs and $O(nS^3)$ in case of $\oplus$AOBDDs. The just presented algorithm gives the postponed proof of Theorem 4.1 and reproves Theorem 5.1.

Acknowledgement

I am indebted to Petr Savický for his carefull reading of early versions of this paper, his suggestions improving the clarity of the statements and the style of the paper.

References

[1] Bryant, R. E., *Graph–based algorithms for Boolean function manipulation,* IEEE Trans. on Computers 1986, 35, pp. 677–691

[2] Bryant, R. E., *Symbolic Boolean manipulation with ordered binary decision diagrams,* ACM Comp. Surveys 1992, 24, pp. 293–318

[3] Coppersmith, D., Winograd, S., *Matrix multiplication via arithmetic progressions,* J. Symbolic Computation 1990, 9, pp. 251–280

[4] Gergov, J., Meinel, Ch., *Mod–2–OBDDs — a data structure that generalizes EXOR–Sum–of–Products and Ordered Binary Decision Diagrams,* Formal Methods in System Design 1996, 8, pp. 273–282

[5] Löbbing, M., Sieling, D., Wegener, I., *Parity OBDDs cannot be handled efficiently enough,* Information Processing Letters 1998, 67, pp. 163–168

[6] Waack, St., *On the descriptive and algorithmic power of parity ordered binary decision diagrams,* in: Proc. 14th STACS 1997, Lecture Notes in Computer Sci. 1200, Springer Verlag 1997, pp. 201-212

[7] Wegener, I., *Branching Programs and Binary Decision Diagrams — Theory and Applications,* to appear (2000) in the SIAM monograph series Trends in Discrete Mathematics and Applications (ed. P. L. Hammer).

[8] Wegener, I., *Efficient data structures for Boolean functions,* Discrete Mathematics 1994, 136, pp. 347–372

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'