Symbolic Topological Sorting with OBDDs
(Extended Abstract)

Philipp Woelfel*

FB Informatik, LS2, Univ. Dortmund, 44221 Dortmund, Germany
philipp.woelfel@cs.uni-dortmund.de

Abstract. We present a symbolic OBDD algorithm for topological sorting which requires $O(\log^2 N)$ OBDD operations. Then we analyze its true runtime for the directed grid graph and show an upper bound of $O(\log^4 N)$. This is the first true runtime analysis of a symbolic OBDD algorithm for a fundamental graph problem, and it demonstrates that one can hope that the algorithm behaves well for sufficiently structured inputs.

1 Introduction

Algorithms on graphs is one of the best studied areas in computer science. Usually, a graph $G = (V, E)$ is given by an adjacency list or by an adjacency matrix. Such an explicit representation of a graph requires space $\Theta(|V|+|E|)$ or $\Theta(|V|^2)$, and for many graph problems efficient algorithms are known. However, there are several application areas where typical problem instances have such a large size that a linear or even polynomial runtime is not feasible, or where even the explicit representation of the problem instance itself may not fit into memory anymore.

In order to deal with very large graphs, *symbolic* (or *implicit*) graph algorithms have been devised, where the vertex and edge sets representing the involved graphs are stored symbolically, i.e., in terms of their characteristic functions. The characteristic functions are usually represented by so-called Binary Decision Diagrams (BDDs) or more specifically by Ordered Binary Decision Diagrams (OBDDs) — see Section 2 for definitions. Such approaches have been successfully applied in the areas of model checking, circuit verification and finite state machine verification (see e.g. [2–4]). These applications can be viewed as particular cases of symbolic graph problems, which raises the question whether it is also possible to devise symbolic graph algorithms with a good behavior for fundamental graph theoretical problems. One approach in this direction was undertaken by Hachtel and Somenzi [5] who introduced a symbolic OBDD algorithm for the maximum flow problem in 0-1 networks. The promising experimental studies demonstrated that the algorithm is able to handle graphs with over 10^{36} edges and that it is competitive with traditional algorithms on dense random graphs. The paper lacks however, a theoretical analysis of its performance with

* Supported in part by DFG grant We 1066/10-1

respect to runtime. Recently, Sawitzki [9, 11] has analyzed the number of OBDD operations (i.e., the number of required synthesis operations of characteristic functions) required by the flow algorithm of Hachtel and Somenzi and has proposed an improved algorithm. But note that there is only a very weak relation between the number of OBDD operations and the true runtime of a symbolic OBDD algorithm. The time required for one synthesis step is mainly influenced by the sizes of the involved OBDDs which may range from linear to exponential (in the number of variables of the represented characteristic functions). In fact, we are not aware of any analysis of a symbolic OBDD algorithm with respect to its true runtime. (However, there is a recent true runtime analysis for a related type of decision diagrams, called Binary Moment Diagrams, showing that certain multiplier circuits can be verified in polynomial time [7].)

The results and techniques presented here aim to be a first step in filling this gap. First, we present a new OBDD algorithm for topological sorting the N vertices of a directed acyclic graph, which requires only $O(\log^2 N)$ OBDD operations on OBDDs for functions with at most $4\lceil \log N \rceil$ variables. Hence, if all OBDDs obtained during the execution of the algorithm have subexponential size, the total runtime is sublinear in the number of vertices of the graph. Then we analyze its true runtime for the directed grid graph and show an upper bound of $O(\log^4 N)$. This demonstrates that one can in fact hope that such a fundamental graph algorithm behaves well for sufficiently structured inputs. For the analysis, we generalize the notion of threshold functions to multivariate threshold functions. We investigate the OBDD size of multivariate threshold (and modulo) functions and obtain strong results about the effect of OBDD operations such as quantification on these functions.

Clearly, our analysis is a "good-case" analysis which is only valid for one particular input instance. We hope, though, that the techniques presented here are a good starting point for developing a framework which allows to analyze symbolic algorithms for fundamental graph problems for larger classes of input instances. In fact, Sawitzki [10] has already successfully applied our framework to analyze the true runtime of his 0-1 network flow algorithm on the grid network.

2 OBDDs and Implicit Graph Representation

In the following, let B_n denote the class of boolean functions $\{0,1\}^n \to \{0,1\}$ and let $X_n = \{x_1, \ldots, x_n\}$ be a set of boolean variables. Let $f \in B_n$ be a function defined over the variables in X_n. The subfunction of f, where k variables $x_{i_1}, \ldots, x_{i_k}$ are fixed to k constants $c_1, \ldots, c_k \in \{0,1\}$ is denoted by $f_{|x_{i_1}=c_1,\ldots,x_{i_k}=c_k}$.

A *variable ordering* π on X_n is a permutation of the indices $\{1, \ldots, n\}$, leading to the ordered list $x_{\pi(1)}, \ldots, x_{\pi(n)}$ of the variables. A π-*OBDD* on X_n for a variable ordering π is a directed acyclic graph with one root, two sinks labeled with 0 and 1, resp., and the following properties: Each inner node is labeled by a variable from X_n and has two outgoing edges, one of them labeled by 0, the other by 1. If an edge leads from a node labeled by x_i to a node labeled by x_j, then $\pi^{-1}(i) < \pi^{-1}(j)$. This means that any directed path passes the nodes in an order respecting the variable ordering π. A π-OBDD is said to *represent a boolean function* $f \in B_n$, if for any $a = (a_1, \ldots, a_n) \in \{0,1\}^n$, the path starting at the root and leading from any x_i-node over the edge labeled by the value of a_i, ends at a sink with label $f(a)$. The *size* of a π-OBDD G is the number of its nodes and is denoted by $|G|$.

The π-OBDD of minimal size for a given function f and a fixed variable ordering π is unique up to isomorphism. A π-OBDD is called *reduced*, if it is the minimal π-OBDD. It is well-known that the size of any reduced π-OBDD for a function $f \in B_n$ is bounded by $O(2^n/n)$. (see [1] for the upper bound with the best constants known).

Let f and g be functions in B_n and let G_f and G_g be π-OBDDs representing f and g, resp., for an arbitrary variable ordering π. In the following, we summarize the operations on OBDDs to which we will refer in this text. For a more detailed discussion on OBDDs and their operations we refer to the monograph [12].

- **Evaluation:** Given $x \in \{0,1\}^n$ compute $f(x)$. This can trivially be done in time $O(n)$.
- **Minimization:** Compute the reduced π-OBDD for f. This is possible in time $O(|G_f|)$.
- **Binary synthesis:** Given a boolean operation $\otimes \in B_2$ compute a reduced π-OBDD G_h representing the function $h = f \otimes g$. This can be done in time $O(|G_h^*|)$, where G_h^* is the graph which consists of all nodes in the product graph of G_f and G_g reachable from the root. The size of G_h is at most $O(|G_h^*|) = O(|G_f| \cdot |G_g|)$.
- **Replacement by constants:** Given a sequence of variables $x_{i_1}, \ldots, x_{i_k} \in X_n$ and a sequence of constants $c_1, \ldots, c_k$, compute a reduced π-OBDD G_h for the subfunction $h := f_{|x_{i_1}=c_1,\ldots,x_{i_k}=c_k} \in B_{n-k}$. This is possible in time $O(|G_f|)$ and the reduced π-OBDD G_h is of smaller size than G_f.
- **Quantification:** Given a variable $x_i \in X_n$ and a quantifier $Q \in \{\exists, \forall\}$, compute a reduced π-OBDD for the function $h \in B_{n-1}$ with $h := (Qx_i)f$, where $(\exists x_i)f := f_{|x_i=0} \vee f_{|x_i=1}$ and $(\forall x_i)f := f_{|x_i=0} \wedge f_{|x_i=1}$. The time for computing this π-OBDD is determined by the time for determining the π-OBDDs for $f_{|x_i=0}$ and $f_{|x_i=1}$ and the time required for the binary synthesis of the two. Hence, it is bounded by $O\big(|G_f|^2\big)$.
- **SAT enumeration:** Enumerate all inputs $x \in f^{-1}(1)$. Using simple DFS techniques, this can be done in optimal time $O(|G_f| + n\big|f^{-1}(1)\big|)$.

We can use OBDDs for an implicit graph representation by letting them represent the characteristic functions of the vertex and edge sets. For practical reasons, though, we assume throughout this text that the vertex set is $V = \{0,1\}^n$ for some $n \in \mathbb{N}$, so that a representation of V is not needed. It is easy to accommodate the algorithm for other vertex sets. In order to encode integers using the binary notation we define $|x| = 2^{n-1}x_{n-1} + \cdots + 2^0 x_0$ for $x \in \{0,1\}^n$.

3 The Topological Sorting Algorithm

Let $G = (V, E)$, $V = \{0, 1\}^n$, be a directed acyclic graph represented by a π-OBDD as described in the former section. The edge relation E defines in a natural way a partial order $\leadsto$ on V, where $v \leadsto w$ if and only if there exists a path from v to w.

In the explicit case a topological sorting algorithm would enumerate all vertices in such a way that if u is enumerated before v, then $v \not\leadsto u$. In the implicit case, we hope for runtimes in the order of $o(|V|)$ in which the enumeration of all vertices is not possible. Hence, a goal might be to obtain a complete order $\prec$ which inherits the properties of $\leadsto$ (i.e., $v \leadsto u$ implies $v \prec u$). Unless $\leadsto$ is a complete order, $\prec$ is not uniquely defined by $\leadsto$, and thus we assume that an arbitrary complete order $\lessdot$ on the vertex set V is given (this may be fixed in advance for the algorithm or may be given as an additional parameter), which determines the order of the elements which are incomparable with respect to $\leadsto$ (i.e., those with $u \not\leadsto v$ and $v \not\leadsto u$).

An alternative is to compute an OBDD which allows to enumerate the elements in their topological order by simple SAT enumeration operations. For any two vertices u, v we denote by $\Delta(u, v)$ the length of the longest path leading from u to v. (The length of a path is the number of its edges.) If no such path exists, then $\Delta(u, v) := -\infty$. Note that $\Delta(v, v) = 0$, since the graph is acyclic. Furthermore, let $\Delta(v) := \max\{\Delta(u, v) \mid u \in V\}$. We call $\Delta(v)$ the length of the *longest path to* the vertex v. Let now DIST $\in B_{2n}$, be defined to be 1 for an input $(d, v) \in \{0, 1\}^n \times \{0, 1\}^n$, if and only if $\Delta(v) = |d|$. Clearly, $|d_u| < |d_v|$ implies $v \not\leadsto u$, where d_u, d_v are the unique values with $\mathrm{DIST}(d_u, u) = 1$ and $\mathrm{DIST}(d_v, v) = 1$. Hence, if we have a π-OBDD G_{DIST} for the function DIST, we can use it to enumerate the vertices in an order respecting $\leadsto$ by computing the π-OBDDs for $\mathrm{DIST}_{|d=a}$ for $|a| = 0, 1, \ldots$ and enumerating their satisfying inputs using the SAT enumeration procedure. We will see below how the OBDD G_{DIST} can in addition be used to obtain a complete order respecting $\leadsto$.

In order to compute the function DIST, we use a method which is similar to that of computing the transitive closure by matrix squaring. For $i \in \{1, \ldots, n\}$ and $u, v \in V$ let $T_i(u, v)$ be the boolean function with function value 1 if and only if there exists a simple path from u to v which has length exactly 2^i. We can compute OBDDs for all T_i as follows.

$$T_0(u, v) = E(u, v) \quad \text{and} \quad T_{i+1}(u, v) = \exists w: T_i(u, w) \wedge T_i(w, v). \tag{S1}$$

Now we define the function $\mathrm{DIST}_j \in B_{2n-j}$ for $0 \le j \le n$. It takes as input an $(n-j)$-bit value $d^* = d_{n-1} \ldots d_j$ and a vertex v (for $j = n$, d^* is the empty string ϵ). The function value $\mathrm{DIST}_j(d^*, v)$ is defined as

$$\mathrm{DIST}_j(d^*, v) = 1 \quad \Leftrightarrow \quad 2^j |d^*| \le \Delta(v) < 2^j(|d^*| + 1). \tag{$*$}$$

I.e., $\mathrm{DIST}_j(d^*, v)$ is true if the bits $d_{n-1} \ldots d_j$ are exactly the $n - j$ most significant bits of the binary representation of the integer $\Delta(v)$. Clearly, DIST =

DIST_0. The functions DIST_j can be computed by $\mathrm{DIST}_n(v) := 1$ and for $j = n-1, \ldots, 0$

$$\begin{aligned}\mathrm{DIST}_j(d_{n-1} \ldots d_j, v) \;=\; & \mathrm{DIST}_{j+1}(d_{n-1} \ldots d_{j+1}, v) \quad \wedge \\ & \Big(d_j \Leftrightarrow \exists u \, \big(T_j(u,v) \,\wedge\, \mathrm{DIST}_{j+1}(d_{n-1} \ldots d_{j+1}, u)\big)\Big). \end{aligned} \tag{S2}$$

It is easy to verify that the boolean functions DIST_j do in fact fulfill property $(*)$ (the proof can be found in the full version of this paper). Once we have computed the function DIST, we can use it together with an arbitrary given complete order $\lessdot$ in order to compute a complete order $\prec$ by letting

$$\begin{aligned} & u \prec v \;\Leftrightarrow\; \exists d_u, d_v : \\ & \quad \mathrm{DIST}(d_u, u) \;\wedge\; \mathrm{DIST}(d_v, v) \;\wedge\; \big(|d_u| < |d_v| \vee (|d_u| = |d_v| \wedge u \lessdot v)\big). \end{aligned} \tag{S3}$$

It can be easily checked that $\prec$ defines a complete order on V respecting $\rightsquigarrow$. Thus, the following theorem follows from simply counting the number of OBDD operations.

Theorem 1. *Let $V = \{0,1\}^n$ and $G = (V, E)$ be an acyclic directed graph represented by* OBDD*s. Applying the* OBDD *operations as described in (S1)–(S3) yields an* OBDD *for a relation $\prec$ which defines a complete order on V such that $v \prec w$ for all $v, w \in V$ with $(v, w) \in E$. The number of* OBDD *operations is $O(n^2)$, where each* OBDD *represents a function on at most $4n$ variables.*

Note that the algorithm can be easily adapted to an arbitrary vertex set $V \subseteq \{0,1\}^n$ given by an OBDD for the relation V. This is done by simply executing the algorithm from above for the edge relation $E'(u,v) = E(u,v) \wedge V(u) \wedge V(v)$. While the complete order $\prec$ returned by such a modified algorithm is defined on $\{0,1\}^n \times \{0,1\}^n$, its restriction to $V \times V$ is obviously a correct complete order.

Since any (not necessarily reduced) OBDD in n variables has $O(2^n)$ nodes, the theorem shows that the true worst-case runtime of our algorithm is $O(|V|^4 \log^2 |V|)$. Clearly, this is much worse than the $O(|V| + |E|)$ upper bound obtained by a well-known explicit algorithm. On the other hand, if all OBDDs obtained during the execution of the algorithm have a subexponential size (with respect to n), then its runtime is sublinear with respect to the number of vertices. In the following sections we show that it is justifiable to hope that this is the case for very structured input graphs.

4 Runtime Analysis for the Grid Graph

We analyze the behavior of the topological sorting algorithm for a $2^n \times 2^n$-grid, where all edges are directed from left to right and from bottom to up. The directed grid graph consists of the vertex set $V = \{0,1\}^n \times \{0,1\}^n$ and edge set E, where $\big((x,y),(x',y')\big) \in E$ if and only if either $|x| = |x'|$ and $|y'| - |y| = 1$ or $|y| = |y'|$ and $|x'| - |x| = 1$.

In the analysis to follow, we assume an interleaved variable ordering, that is a variable ordering where e.g. for a function depending on two vertices u, v, the variable v_i precedes the corresponding variable u_i. Note that in practice, heuristics such as sifting algorithms [8] are used to optimize the variable orderings during the execution of an algorithm, and it can be expected that a good variable ordering is found this way.

The idea for proving that the topological sorting algorithm is very efficient for the grid graph is that all functions represented by OBDDs after each step of the algorithm belong to a class of functions which have a small OBDD representation. The functions we consider are compositions of certain threshold and modulo functions, which we define and investigate in the following.

We denote by $X_{k,n}$ the set of variables x_j^i with $1 \le i \le k$ and $0 \le j < n$. By x^i we denote the vector of n variables $(x_{n-1}^i, \ldots, x_0^i)$.

Definition 1.

1. *A boolean function $f \in B_{kn}$ defined on the variable set $X_{k,n}$ is called k-*variate threshold function*, if there exist a* threshold *$T \in \mathbb{Z}$ and* weights *$w_1, \ldots, w_k \in \mathbb{Z}$ such that $f(x^1, \ldots, x^k) = 1$ if and only if $\sum_{i=1}^k w_i \cdot |x^i| \ge T$. The* maximum absolute weight *of f is defined as $w(f) := \max\{|w_1|, \ldots, |w_k|\}$. The set of k-variate threshold functions with maximum absolute weight w defined on the set of variables $X_{k,n}$ is denoted by $\mathbb{T}_{k,n}^w$*
2. *A boolean function $g \in B_{kn}$ defined on the variable set $X_{k,n}$ is called k-*variate modulo M function*, if there exists a constant $C \in \mathbb{Z}$ and $w_1, \ldots, w_k \in \mathbb{Z}$ such that $g(x^1, \ldots, x^k) = 1$ if and only if $\sum_{i=1}^k w_i \cdot |x^i| \equiv C \pmod M$. The set of k-variate modulo M functions defined on the set of variables $X_{k,n}$ is denoted by $\mathbb{M}_{k,n}^M$.*

Definition 2. *Let $f \in B_n$ and $\mathcal{C}$ be a class of functions defined on the variable set X_n. We say that f can be* decomposed *into m functions in $\mathcal{C}$, if there exist a formula F on m variables and $f_1, \ldots, f_m \in \mathcal{C}$ such that $f = F(f_1, \ldots, f_m)$. The set of functions decomposable into m functions in $\mathcal{C}$ is denoted by $D[\mathcal{C}, m]$. For any $k \in \mathbb{N}$ we denote by D_k the set of function sequences $(f_n)_{n \in \mathbb{N}}$ such that $\exists m \in \mathbb{N}\, \forall n \in \mathbb{N} : f_n \in D\left[\mathbb{T}_{k,n}^1, m\right]$.*

The main idea in our proof is based on two observations. Firstly, any function decomposable into a constant number of threshold and modulo functions has a small OBDD size. Secondly, all intermediate OBDDs obtained during the execution of the topological sorting algorithm on the directed grid graph represent functions which are decomposable into threshold and modulo functions.

Let $\pi_{k,n}$ be the variable ordering in which the variable in $X_{k,n}$ appear in the order $x_0^1, x_0^2, \ldots, x_0^k, x_1^1, \ldots, x_1^k, \ldots, x_{n-1}^k$. I.e., a $\pi_{k,n}$-OBDD tests all bits of the input integers in an interleaved order with increasing significance of the bits. The following result is a generalization of Proposition 4 in [6]. The proof will be given in the full version of this extended abstract.

Lemma 1. *Let $f_1, \dots, f_m \in \mathbb{T}^w_{k,n} \cup \mathbb{M}^M_{k,n}$ be given by reduced $\pi_{k,n}$-OBDDs for f_i, $1 \le i \le m$. Further, let $f = F(f_1, \dots, f_m)$ for a formula F of size s and let $L = L(k, m, M) = \max\{4kw + 5, M\}$. The minimal $\pi_{k,n}$-OBDD for f has at most $L^{s+1}kn$ nodes and can be computed in time and space $O\big((kn)^2 sL^{s+1} + 1\big)$.*

Now we show for functions being decomposable into threshold functions (and no modulo functions), that the quantification over one of its variable blocks $x^i_0, \dots, x^i_{n-1}$, $1 \le i \le k$, can be done efficiently.

Theorem 2. *Let $(f_n)_{n\in\mathbb{N}}$ such that there exist $w, m \in \mathbb{N}$ with $f_n \in D[\mathbb{T}^w_{k,n}, m]$ for all $n \in \mathbb{N}$, and let $Q \in \{\exists, \forall\}$. If f_n is given as a $\pi_{k,n}$-*OBDD*, then for any $1 \le \ell \le k$ a minimal $\pi_{k,n}$-*OBDD *for $(Qx^\ell)f_n$ can be computed in time $n^3k^{O(1)}$.*

We need the following lemma, which states that the result of quantifying over one variable block of a function decomposable into threshold functions is a function which is decomposable into threshold and modulo functions. The proof has to be omitted due to space restrictions. Let lcm denote the *least common multiple.*

Lemma 2. *Let $f \in D\Big[\mathbb{T}^w_{k,n}, m\Big]$, and $Q \in \{\exists, \forall\}$ and $\ell \in \{1, \dots, k\}$. Then $(Qx^\ell)f \in D\Big[\mathbb{T}^{2w\cdot w^*}_{k-1,n} \cup \mathbb{M}^{w^*}_{k-1,n}, m'\Big]$, where $w^* \le \mathrm{lcm}\{1, 2, \dots, w\}$ and $m' = O(2^m w^* m^2)$. In particular, for any fixed $k \in \mathbb{N}$ and a sequence of functions $(f_n)_{n\in\mathbb{N}} \in D_k$ we have $(Qx^i)f_n \in D\Big[\mathbb{T}^2_{k-1,n}, m\Big]$, where $m = O(1)$.*

Proof (of Theorem 2). Fix $w, m \in \mathbb{N}$ such that $f_n \in D[\mathbb{T}^w_{k,n}, m]$ for all $n \in \mathbb{N}$. W.l.o.g. we assume $\ell = 1$ and for the sake of readability we write x instead of x^1 and f instead of f_n. We only prove the theorem for the case $Q = \forall$; the proof for $Q = \exists$ works analogously. We can write $(\forall x)f$ as $(\forall x_{n-1}\forall x_{n-2} \dots \forall x_0)f(x^2, \dots, x^k)$. If we apply the OBDD quantification operations to the bits $x_0, \dots, x_{n-1}$ in this order, then after the ith quantification $(0 \le i \le n)$ the resulting OBDD G_i represents the function $g_i = (\forall x_{i-1} \dots \forall x_0)f$ in B_{kn-i}. Since each of the n quantification operations can be done in time $O(|G_i|^2)$, the total time required is bounded by $\sum_{i=0}^{n-1} |G_i|^2$. Hence, it suffices to show that G_i has a size of at most $O(nk^{O(1)})$ for all $0 \le i \le n-1$.

Note that g_i does not depend on the variables $x_0, \dots, x_{i-1}$. What we do in the following is to introduce n dummy variables $z_0, \dots, z_{n-1}$ and to show that g_i can be written as $\big((\forall z_0, \dots, z_{n-1})g^*_i\big)_{|x_0=0,\dots,x_{i-1}=0}$, where g^*_i is a function in $D\Big[\mathbb{T}^w_{k+1,n}, m+1\Big]$. Hence, g_i is obtained from the function $(\forall z_0, \dots z_{n-1})g^*_i$ by restricting some variables to constants. By Lemma 2, this function is decomposable into a constant number of threshold functions, and therefore its OBDD size is bounded sufficiently. Note that the variables $z_0, \dots, z_{n-1}$ are merely artifical helper variables, and that none of the functions we "really" deal with (i.e., which are represented by OBDDs) depend on these variables.

Let $f = F(f_1, \dots, f_m)$ for a formula F and $f_1, \dots, f_m \in \mathbb{T}^w_{k,n}$. Since $m = O(1)$, we may assume w.l.o.g. that the size s of F is a constant, too. We introduce

n new variables, which we denote by $z_0, \dots, z_{n-1}$. Then we replace the variables x_j with the variables z_j for $0 \le j \le i-1$. This way we obtain

$$
\begin{aligned}
g_i &= (\forall x_{i-1} \dots x_0) f(x_{n-1} \dots x_i x_{i-1} \dots x_0, x^2, \dots, x^k) \\
&= (\forall z_{i-1} \dots z_0) f(x_{n-1} \dots x_i z_{i-1} \dots z_0, x^2, \dots, x^k) \\
&= (\forall z_{n-1} \dots z_0) \big(|z| \ge 2^i \ \vee \ f(x_{n-1} \dots x_i z_{i-1} \dots z_0, x^2, \dots, x^k) \big) \qquad (1)
\end{aligned}
$$

Now consider an arbitrary threshold function f_j for some $1 \le j \le m$. I.e., $f_j(x, x^2, \dots, x^k) = 1$ if and only if $w_1|x| + w_2|x^2| + \cdots + w_k|x^k| \ge T$. Let $f_j^* \in B_{(k+1)n}$ be the function with

$$
f_j^*(z, x, x^2, \dots, x^k) = 1 \quad \Leftrightarrow \quad w_1|z| + w_1|x^1| + w_2|x^2| + \cdots + w_k|x^k| \ge T
$$

and $f^* = F(f_1^*, \dots, f_m^*)$. Obviously, $f^* \in D[\mathbb{T}^w_{k+1,n}, m]$. If $|z| < 2^i$, then $|x_{n-1} \dots x_i z_{i-1} \dots z_0|$ is the same as $|x_{n-1} \dots x_i 0 \dots 0| + |z|$. Hence, it is easy to conclude from (1) that

$$
\begin{aligned}
g_i &= (\forall z_{n-1} \dots z_0) \big(|z| \ge 2^i \ \vee \ f^*(z, x_n \dots x_i 0 \dots 0, x^2, \dots, x^k) \big) \\
&= (\forall z_{n-1} \dots z_0) \Big(|z| \ge 2^i \ \vee \ f^*_{|x_{i-1} = \cdots = x_0 = 0}(z, x^1, x^2, \dots, x^k) \Big).
\end{aligned}
$$

Now let

$$
g_i^*(x^1, \dots, x^k) \ = \ |z| \ge 2^i \ \vee \ f^*(z, x^1, x^2, \dots, x^k).
$$

Then $g_i^* \in D\Big[\mathbb{T}^w_{k+1,n}, m+1\Big]$ and $g_i = \big((\forall z) g_i^*\big)_{|x_0=0,\dots,x_{i-1}=0}$. Since $g_i^* \in D\Big[\mathbb{T}^w_{k+1,n}, m+1\Big]$ and k, w, and m are constants, we can conclude from Lemma 2 that $(\forall z) g_i^* \in D\Big[\mathbb{T}^{w'}_{k,n} \cup \mathbb{M}^M_{k,n}, m'\Big]$ for some constants w', M, and m'. Thus, by Lemma 1 the $\pi_{k,n}$-OBDD size of $(\forall z) g_i^*$ is bounded by $O(nk^{O(1)})$. But as we have shown above, the $\pi_{k,n}$-OBDD for g_i can be obtained from the $\pi_{k,n}$-OBDD for $(\forall z) g_i^*$ by simply replacing some variables with the constant 0. Hence, the resulting minimal $\pi_{k,n}$-OBDD for g_i can only be smaller than that for $(\forall z) g_i^*$ and thus its size is also bounded by $O(nk^{O(1)})$. $\square$

Remark 1. All the upper bounds in Lemma 1 and Theorem 2 proven for functions decomposable into threshold- and modulo functions hold equivalently for their subfunctions $f_{|\alpha_1 \dots \alpha_i}$, where $\alpha_1 \dots \alpha_i$ is a restriction to arbitrary variables except those being quantified in case of Theorem 2.

The following corollary summarizes the abovely stated results in a more convenient way. It follows from the statements in Lemma 1, Theorem 2, Lemma 2, and Remark 1.

Corollary 1. *Fix a constant $k \in \mathbb{N}$ and let $i, j \in \{1, \dots, k\}$ and $Q, Q' \in \{\exists, \forall\}$. Further, let $(g_n)_{n \in \mathbb{N}} \in D_k$ and $f_n = g_{n|\alpha}$, where α is an assignment of constants to arbitrary variables except to those in $\{x_0^i, \dots, x_{n-1}^i\}$. If g_n is either given by a reduced $\pi_{k,n}$-OBDD or by the reduced $\pi_{k,n}$-OBDDs for the threshold functions into which it is decomposable, then the reduced $\pi_{k,n}$-OBDDs for $(Qx^i)g_n$, $(Qx^i)f_n$, and $(Qx^iQ'x^j)g_n$ can be computed in time $O(n^3)$.*

We can now apply these results in order to analyze the true run time of the topological sorting algorithm for the grid graph. Whenever we talk in the following about an OBDD for some function sequence in D_k, we assume that the variable ordering is $\pi_{k,n}$. We have to specify the complete order $\lessdot$ for the operations in (S3). A very natural order is the lexicographical order, i.e. $(x^1, y^1) \lessdot (x^2, y^2)$ if and only if $|x^1| < |x^2| \vee \big(|x^1| = |x^2| \wedge |y^1| \leq |y^2|\big)$.

Recall the steps (S1)-(S3) of the topological sorting algorithm from Section 3. We start with the analysis of the edge relation E. By the definition of the grid graph, $\big((x^1, y^1), (x^2, y^2)\big) \in E$ if and only if $\big(|x^2| - |x^1| = 0 \wedge |y^2| - |y^1| = 1\big) \vee \big(|y^2| - |y^1| = 0 \wedge |x^2| - |x^1| = 1\big)$. Clearly, this function is in D_4.

Now we look at the functions T_i obtained by (S1). Recall that $T_i(u, v)$ is defined to be 1 if and only if there exists a path from u to v which has length exactly 2^i. Note also that in the directed grid graph all paths from vertex u to vertex v have the same length. Hence, for the directed grid graph $T_i\big((x^1, y^1), (x^2, y^2)\big) = 1$ if and only if

$$|y^2| \geq |y^1| \quad \wedge \quad |x^2| \geq |x^1| \quad \wedge \quad |x^2| - |x^1| + |y^2| - |y^1| = 2^i.$$

Clearly, this function is in D_4 and thus according to Corollary 1, T_{i+1} can be computed from T_i in time $O(n^3)$. (Note also that the quantification over one vertex in the grid graph is a quantification over two integers.) Hence, computing $T_1, \dots, T_n$ requires time $O(n^4)$ in total.

Next, we analyze the construction of the OBDDs for the functions DIST_j in (S2). Recall that for any vertex v and any $d^* = d_{n-1} \dots d_j$, the function $\mathrm{DIST}_j(d^*, v)$ is true if and only if d^* describes the $n - j$ most significant bits of the bitwise representation of $\Delta(v)$. Let $f_j \in B_{3n}$, $0 \leq j \leq n$, be defined by $f_j(d, x, y) = 1$ if and only if $|d| \leq |x| + |y| < |d| + 2^j$. Hence, f_j is the conjunction of two functions in $\mathbb{T}^1_{3,n}$. Furthermore, it is easy to see that $\mathrm{DIST}_j\big(d_{n-1} \dots d_j, (x, y)\big) = f_{|d_{j-1}=\dots=d_0=0}(d, x, y)$. Therefore, DIST_j is obtained from a function in D_3 by replacing some variables with the constant 0. Note also that $\mathrm{DIST} = \mathrm{DIST}_0$ is in fact in D_3. Moreover, due to the analysis of T_j above, it becomes obvious that $T_j(u, v) \wedge \mathrm{DIST}_{j+1}(d_{n-1} \dots d_{j+1}, u)$ is a function in D_5, where some variables are replaced with the constant 0. Hence, according to Corollary 1, the OBDD for

$$g_j \;:=\; \exists u : \; T_j(u, v) \wedge \mathrm{DIST}_{j+1}(d_{n-1} \dots d_{j+1}, u)$$

can be computed in time $O(n^3)$. The function g_j is obtained from a function in $D\big[\mathbb{T}^8_{3,n} \cup \mathbb{M}^2_{3,n}, O(1)\big]$ by replacing some variables with the constant 0 (see Lemma 2). Now it is easy to see that the final two synthesis operations of (S2) required to compute DIST_j run in time $O(n^3)$. (Apply Lemma 1 and Remark 1, and note that the function $d_j \in B_1$ can be viewed as a subfunction of $f \in D_1$ with $f(d_{n-1} \dots d_0) = 1$ if and only if $|d_{n-1} \dots d_0| = 2^j$.) Altogether, the total time required for computing $\mathrm{DIST}_{n-1}, \dots, \mathrm{DIST}_0 = \mathrm{DIST}$ is $O(n^4)$.

Finally, we have to investigate the computation of the complete order $\prec$ using the operations in (S3). Recall that $\mathrm{DIST} \in D_3$ Hence, if one takes the definition of $\lessdot$ into account, the complete term in (S3) before the first quantification describes a function h in D_4. According to Corollary 1 the function $h' = (\exists d_v \exists d_u)h$ can be computed in time and space $O(n^3)$.

Summing up the time bounds for all OBDD operations, we have obtained the following result.

Theorem 3. *The* OBDD *algorithm for topological sorting takes time $O(n^4)$ on the directed $2^n \times 2^n$ grid graph for an appropriate variable ordering $\pi_{k,n}$ and the complete order $\lessdot$ as defined above.*

5 Conclusion

Since the results about the threshold and modulo functions are quite general, we hope that they might as well be applicable to the analysis of other symbolic OBDD algorithms. It would be nice to extend the techniques in such a way that not only single input instances but small graph classes can be handled. An interesting example would be grids where some arbitrary or randomly chosen edges have been removed.

Acknowledgments

I thank Daniel Sawitzki and Ingo Wegener for helpful comments and discussions.

References

1. Y. Breitbart, H. B. Hunt III, and D. J. Rosenkrantz. On the size of binary decision diagrams representing boolean functions. *Theor. Comp. Sci.*, 145:45–69, 1995.
2. J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Inform. and Comp.*, 98:142–170, 1992.
3. H. Cho, G. Hachtel, S.-W. Jeong, B. Plessier, E. Schwarz, and F. Somenzi. ATPG aspects of FSM verification. In *IEEE Int. Conf. on CAD*, pp. 134–137. 1990.
4. H. Cho, S.-W. Jeong, F. Somenzi, and C. Pixley. Synchronizing sequences and symbolic traversal techniques in test generation. *Journal of Electronic Testing: Theory and Applications*, 4:19–31, 1993.
5. G. D. Hachtel and F. Somenzi. A symbolic algorithm for maximum flow in 0-1 networks. *Formal Methods in System Design*, pp. 207–219, 1997.
6. S. Jukna. The graph of integer multiplication is hard for read-k-times networks. Technical Report 95–10, Universität Trier, 1995.
7. M. Keim, R. Drechsler, B. Becker, M. Martin, and P. Molitor. Polynomial formal verification of multipliers. *Formal Methods in System Design*, 22:39–58, 2003.
8. R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *IEEE Int. Conf. on CAD*, pp. 42–47. 1993.
9. D. Sawitzki. Implicit flow maximization by iterative squaring. Manuscript. `http://ls2-www.cs.uni-dortmund.de/~sawitzki`.
10. D. Sawitzki. Implicit flow maximization on grid networks. Manuscript. `http://ls2-www.cs.uni-dortmund.de/~sawitzki`.
11. D. Sawitzki. Implizite Algorithmen für Graphprobleme. Diploma thesis, Univ. Dortmund, 2002.
12. I. Wegener. *Branching Programs and Binary Decision Diagrams - Theory and Applications*. SIAM, 2000.