$\oplus$-OBDD in Symbolic Model Checking [⋆]

Vojtěch Řehák

Department of Computer Science, Faculty of Informatics
Masaryk University Brno, Czech Republic
rehak@fi.muni.cz

Abstract. We present a feasibility study of using $\oplus$-OBDD data structure in symbolic model checking (SMC). $\oplus$-OBDD has been proposed as a more succinct modification of well known OBDD data structure which is of common use in nowadays SMC. We introduce three modifications of $\oplus$-OBDD, analyze their respective efficiency, and present some experimental results based on implementations of $\oplus$-OBDD within a symbolic model checker NuSMV.

1 Introduction

Nowadays, hardware and software systems are widely used in applications where failure is unacceptable. As examples can serve systems for electronic commerce, air traffic control, medical instruments, and many others. Hence, methods for validating these systems are in great demand. This paper focuses on the model checking approach to automatic validation.

Model checking [3] is an automatic technique for verifying finite state systems. In this approach, properties are expressed in a temporal logic and systems are modelled as transition systems. A model checker accepts two inputs, a transition system and a temporal formula, and returns "true" if the system satisfies the formula; otherwise it returns "false".

The basic model checking problem is the *state explosion problem* due to the fact that the size (number of states) of transition system can be exponential with respect to the description of the system. The exponenciality is (mostly) caused by parallel composition of interacting processes. Hence, the basic problem is the space complexity of model checking algorithms. One of the most successful and widely commercially used approach to avoid the explosion problem is a symbolic approach. Symbolic model checking algorithms are based on manipulations with sets of states of the transition system where sets of states are represented by *Ordered Binary Decision Diagrams* [1] (OBDDs).

The first step in using OBDDs in the symbolic model checking is the representation of states of transition systems as boolean vectors and the representation of sets of states as boolean functions. OBDDs effectively represent boolean functions and allow efficient implementation of complementation, union and equality test on sets (of states). In this paper we investigate a possibility of using a novel

[⋆] This work has been partially supported by the GACR grant No. 201/00/1023.

data structure, so called $\oplus$-OBDD, and its modifications, to reduce the space complexity of the symbolic model checking algorithm.

2 Definition of $\oplus$-OBDD

A $\oplus$-OBDD [5] (called also Mod2-OBDD or Parity-OBDD) is an extension of OBDD data structure, namely there are $\oplus$-nodes in $\oplus$-OBDD. This innovation can lead to exponentially more succinct representation of sets of states.

Syntax A $\oplus$-BDD P over a set $X_n = \{x_1, \ldots, x_n\}$ of boolean variables is a directed acyclic connected labelled graph $P = (V, E)$ with three types of nodes:

- a *terminal node* v has a label $l(v) \in \{0, 1\}$ and its out-degree is 0,
- a *variable (branching) node* v has a label $l(v) = x_i$ ($x_i \in X_n$) and two successors denoted by $low(v), high(v) \in V$,
- a $\oplus$*-node* v has a label $l(v) = \oplus$ and two successors $low(v), high(v) \in V$.

Edges from v to $low(v)$ and $high(v)$ are labeled as 0-*edge* and 1-*edge*, respectively. A node with in-degree 0 is the *root*.

A $\oplus$-BDD is *free* if each variable is encountered at most once on each path from the root to a terminal node. A $\oplus$-BDD is *ordered* if it is free and the variables are encountered in the same order on each path from the root to a terminal node. Ordered $\oplus$-BDD is denoted as $\oplus$-OBDD.

Semantics Each node v of $\oplus$-BDD represents a boolean function $f_v : \{0, 1\}^n \to \{0, 1\}$. The definition of f_v is given recursively as follows:

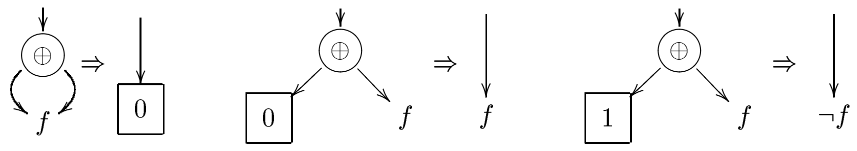
- If v is a *terminal node*, then $f_v(x_1, \ldots, x_n) = l(v)$.
- If v is a *branching (variable) node* with $l(v) = x_i$, then
  $f_v(x_1, \ldots, x_n) = (\neg x_i \wedge f_{low(v)}(x_1, \ldots, x_n)) \vee (x_i \wedge f_{high(v)}(x_1, \ldots, x_n))$.
- If v is a $\oplus$*-node*, then
  $f_v(x_1, \ldots, x_n) = f_{low(v)}(x_1, \ldots, x_n) \oplus f_{high(v)}(x_1, \ldots, x_n)$,
  where $\oplus$ is a boolean exclusive or (XOR) operator.

A $\oplus$-BDD with the root v represents the boolean function $f_v(x_1, \ldots, x_n)$.

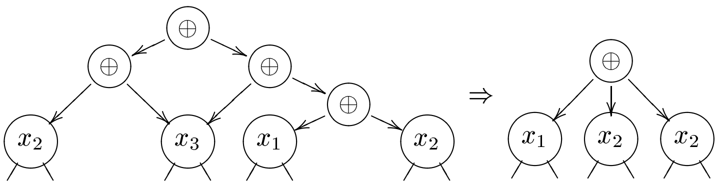
3 Reduced Forms of $\oplus$-OBDD

In order to achieve even more succinct representation several reduction rules are introduced. Duplicate nodes can be unified (merging rule) and redundant variable nodes can be omitted (elimination rule) [13]. The semantics of $\oplus$-nodes enables many other reductions – some of them are proposed here.

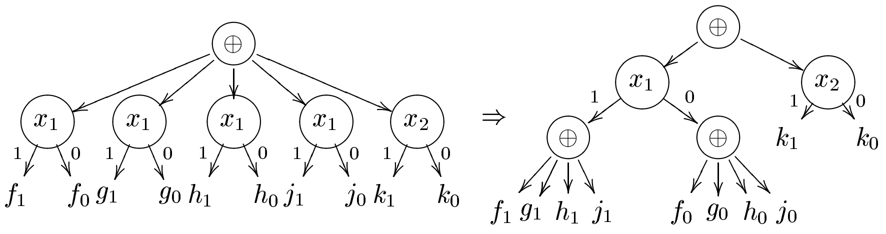
Basic $\oplus$-OBDD The basic form can be obtained by applying the following three rules for $\oplus$-nodes.



$\oplus$-OBDD with $\oplus$-meta-nodes A $\oplus$-*meta-node* is a $\oplus$-node which can have more than two successors. Successors of $\oplus$-meta-node are sorted. This reduction is based on commutativity and associativity of the boolean operation XOR.



Merged $\oplus$-OBDD This reduced form unifies $\oplus$-node's successors with the same label.



Exact definitions of these modifications are presented in [11]. The basic disadvantage of merged $\oplus$-OBDD is the loss of strictly bottom-up implementation of the reducing algorithm. The reduction must be performed top-down. Hence, the careful creation (keeping the reduced form) of a new node is not of constant complexity but it has a linear time complexity.

4 Operations

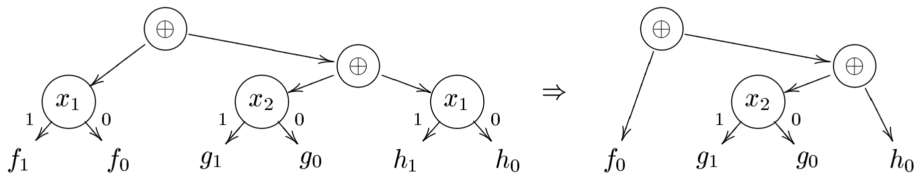
The simplest operation is the complementation (negation) which is implemented using *complemented edges* [7].

The implementation of the union (disjunction) is based on the recursive call for co-factors for the topmost variable. Co-factors of the union can be computed as:

$$(f \vee g)|_{x\leftarrow 0} = f|_{x\leftarrow 0} \vee g|_{x\leftarrow 0} \quad \text{and} \quad (f \vee g)|_{x\leftarrow 1} = f|_{x\leftarrow 1} \vee g|_{x\leftarrow 1}$$

where $f|_{x\leftarrow 0}$ and $f|_{x\leftarrow 1}$ are co-factors of f for $x = 0$ and $x = 1$, respectively.

Co-factoring for the topmost variable is easy provided if the root is a variable node. If the root is $\oplus$-node then its successors must be co-factored.

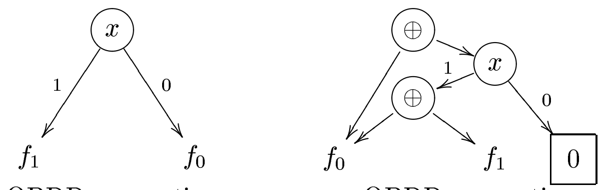


An example of co-factoring $\oplus$-OBDD for $x_1 = 0$.

Co-factors of the result cannot be connected as in OBDD because the OBDD connection does not create $\oplus$-nodes. $\oplus$-OBDD connection is performed according to the *positive Davio expansion rule* [6]:

$$f = f|_{x\leftarrow 0} \oplus x(f|_{x\leftarrow 1} \oplus f|_{x\leftarrow 0}).$$

Both OBDD and $\oplus$-OBDD connections are presented in the following figure (f_0, f_1 are co-factors of the result).



OBDD connection $\oplus$-OBDD connection

The most complex operation is the equality test which can be performed in time $O(n * (|F| + |G|)^3)$ and in space $O((|F| + |G|)^2)$, where n is the number of variables [12] and thus it is not efficient. On the other hand, the probabilistic equivalence test [4] for $\oplus$-OBDD needs linear time only.

The time complexities of all operations are presented in Table 1. For detailed descriptions of algorithms and proofs of its complexities see [11].

	OBDD	**$\oplus$-OBDD**		
procedure		**basic**	**meta nodes**	**merged**
careful creation	$O(1)$	$O(1)$	$O(1)$	$O(\|F\|)$
negation	$O(1)$	$O(1)$	$O(1)$	$O(1)$
$F\|_{x_{top}\leftarrow 0}$	$O(1)$	$O(\|F\|)$	$O(\|F\|)$	$O(1)$
union	$O(\|F\|*\|G\|)$	$O(\|F\|*\|G\|)$	$O(\|F\|^2*\|G\|^2)$	$O(\|F\|^2*\|G\|^2)$
equivalence test	$O(1)$	$O(\|F\|+\|G\|)$	$O(\|F\|+\|G\|)$	$O(\|F\|+\|G\|)$

Table 1. Comparison of the time complexities.

5 Experimental Results

I have modified the implementation of the widely used OBDD package CUDD [10] so as it allows manipulations with $\oplus$-OBDDs. The main reason for choosing CUDD package is its compatibility with the symbolic model checker NuSMV [2].

The CUDD package has been augmented with a node counter *MaxUsedKey* which keeps the maximal number of active nodes during the computation. A node is active if it is created and it is not designated to be erased. Therefore *MaxUsedKey* corresponds to the maximal size of memory which is occupied during the computation of NuSMV.

I have implemented all three types of $\oplus$-OBDD introduced here into the CUDD package. All new nodes are created according to the positive Davio expansion rule and many $\oplus$-nodes may be thus superfluous. $\oplus$-meta-nodes were not implemented as single special nodes as the original CUDD package does not support meta-nodes. Therefore, $\oplus$-meta-nodes are implemented as chains of $\oplus$-nodes. Even this implementation is beneficial as it allows efficient detection of redundancy. Any heuristics reordering variables have not been implemented.

	OBDD	**$\oplus$-OBDD**		
protocol		**basic**	**meta nodes**	**merged**
dartes	-	-	-	-
counter	47	54	54	51
dme1	-	-	-	-
mutex	104	145	131	127
mutex1	306	895	573	567
ring	124	252	175	170
semaphore	237	490	376	358
short	22	22	22	22
gigamax	52633	-	127864	191084
hwb6	789	3274	1955	1878
newring	60	97	82	73
p_error	8182446	-	-	-
p	5194783	-	-	-
philo	5543735	-	8491390	-
robot	33346	-	44723	-

Table 2. The maximal number of active nodes during the computation of NuSMV.

I have verified some protocols by NuSMV and compared the *MaxUsedKeys* with respect to the representation used. Results of these comparisons are presented in Table 2. The column *protocol* contains the names of protocols which have been verified. Next columns contain the values of *MaxUsedKeys*. A dash indicates the computation did not finish in twelve hours. The probabilistic test of equality did not cause any incorrect result.

Results presented in Table 2 indicate that the stricter modifications of $\oplus$-OBDD is used the more succinct representation is obtained. Nevertheless, the OBDD representations remain the best one for the verified protocols.

Computation of large protocols (**philo** and **robot**) did not terminate for the merged modification. It is a consequence of the fact that the careful creation takes a linear time with respect to the size of merged $\oplus$-OBDD.

6 Conclusions

$\oplus$-OBDDs seem to be a promising alternative to OBDDs because they admit a more compact representation of boolean functions. However, our comparisons indicate that the $\oplus$-OBDDs are not so good for symbolic model checking as OBDDs.

The adverse result of our comparisons may be induced by incompleteness of our implementation such as representing $\oplus$-meta-node as a chain of $\oplus$-nodes and absence of heuristic algorithms for reordering and $\oplus$-node placement. Elimination of these imperfections may lead to the applicability of $\oplus$-OBDDs. Some elementary heuristic algorithms are presented in [8, 9].

$\oplus$-OBDDs are very suitable for performing boolean operation XOR. Symbolic model checking algorithms are performed according to the CTL formula which is made by a person and people are not accustomed to use XOR. So, it is better to apply $\oplus$-OBDDs into algorithms where the need for XOR operations springs up naturally.

References

1. R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. In *IEEE Transactions on Computers*, volume C-35-8, pages 677–691, August 1986.
2. A. Cimatti, E. M. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: A New Symbolic Model Checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.
3. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* The MIT Press, Cambridge, Massachusetts, 1999.
4. J. Gergov and Ch. Meinel. Frontiers of Feasible and Probabilistic Feasible Boolean Manipulation with Branching Programs. In *Proceedings of STACS*, volume 665 of *LNCS*. Springer, 1993.
5. J. Gergov and Ch. Meinel. Mod-2-OBDD's: A Generalization of OBDD's and EXOR-Sum-of-Products. Technical Report 93–21, Universität Trier, 1993.
6. Ch. Meinel and H. Sack. Case Study: Manipulating $\oplus$-OBDDs by Means of Signatures. In *3rd International Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, Oxford, UK, 1997.
7. Ch. Meinel and H. Sack. Parity-OBDDs - a BDD Structure for Probabilistic Verification. In *ENTCS*, volume 22. Elsevier, 2000.
8. Ch. Meinel and H. Sack. A Heuristic for $\oplus$-OBDD Minimization. Technical report, Universität Trier, 2001.
9. Ch. Meinel and H. Sack. Improving XOR-Node Placement for $\oplus$-OBDDs. Technical report, Universität Trier, 2001.
10. F. Somenzi. CUDD: CU Decision Diagram Package Release, 1998.
11. V. Řehák. Randomized symbolic model checking. Master's thesis, Masaryk University Brno, 2002.
12. S. Waack. On the Descriptive and Algorithmic Power of Parity Ordered Binary Decision Diagrams. In *Proceedings of STACS*, volume 1200 of *LNCS*. Springer, 1997.
13. I. Wegener. *Branching Programs and Binary Decision Diagrams: Theory and Applications.* Society for Industrial and Apllied Mathematics, 2000.