RexBDDs: Reduction-on-Edge Complement-and-Swap Binary Decision Diagrams

Gianfranco Ciardo
Iowa State University
Ames, IA, USA
ciardo@iastate.edu

Andrew S. Miner
Iowa State University
Ames, IA, USA
asminer@iastate.edu

Lichuan Deng
Iowa State University
Ames, IA, USA
lcdeng@iastate.edu

Junaid Babar
Collins Aerospace
Cedar Rapids, IA, USA
junaid.babar@collins.com

ABSTRACT

We introduce RexBDDs, binary decision diagrams (BDDs) that exploit reduction opportunities well beyond those of reduced ordered BDDs, zero-suppressed BDDs, and recent proposals integrating multiple reduction rules. RexBDDs also leverage (output) complement flags and (input) swap flags to potentially decrease the number of nodes by a factor of four. We define a reduced form of RexBDDs that ensures canonicity, and use a set of benchmarks to demonstrate their superior storage and runtime requirements compared to previous alternatives.

KEYWORDS

Binary decision diagrams, canonicity

ACM Reference Format:
Gianfranco Ciardo, Andrew S. Miner, Lichuan Deng, and Junaid Babar. 2024. RexBDDs: Reduction-on-Edge Complement-and-Swap Binary Decision Diagrams. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3656533

1 INTRODUCTION

Binary decision diagrams (BDDs) have greatly helped a variety of industrial applications, from VLSI design to circuit and protocol verification. Ordered BDDs are directed acyclic graphs where each nonterminal node belongs to a level, corresponding to one of L input variables, and are often a compact option to encode a boolean function. To ensure *canonicity*, they forbid *duplicate* nodes, and enforce *reduction rules* to interpret level-skipping edges. The two most popular canonical forms are reduced ordered BDDs [4] (we call them FBDDs) and zero-suppressed BDDs [10] (we call them ZBDDs), which use different reduction rules and are best suited to different applications, depending on which reduction eliminates the most nodes, but this may not be easy to know *a priori*.

The recently proposed *tagged BDDs* (tBDDs) [12] and *chain reduced BDDs* (cBDDs) [5] combine these reductions; ESRBDDs [3] offer even greater flexibility by explicitly labeling each edge with one of three reductions, while CESRBDDs [2] allow five reductions and complement flags [1].

We extend these ideas by using further symmetric reductions on edges, and incorporating complement and swap [11] flags. We show that the resulting *rule-on-edge, complement and swap flags BDDs* (RexBDDs) can be made canonical, and confirm their efficiency through an extensive set of benchmarks.

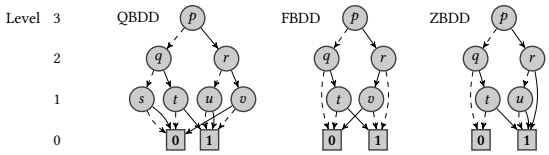

Figure 1: BDDs encoding $(\neg x_3 \wedge x_2 \wedge x_1) \vee (x_3 \wedge \neg(x_2 \wedge x_1))$.

2 BACKGROUND AND RELATED WORK

2.1 Quasi/Fully-reduced, Zero-suppressed BDDs

Definition 1. An L-variable *ordered BDD* (OBDD) is an edge-labeled directed acyclic graph where the terminal nodes are **0** and **1**, at level 0, while each nonterminal node p belongs to a level $p.lvl = k \in \{1, ..., L\}$, corresponding to the domain of boolean variable x_k, and has an edge to a 0-child, $p[0]$, and an edge to a 1-child, $p[1]$, satisfying $k > p[0].lvl$ and $k > p[1].lvl$ (this is the *ordered property*). The 0-child is also known as the "low" or "L" child, and the 1-child as the "high" or "H" child; graphically, these edges are shown using a dashed or solid line, respectively. □

If we give a meaning to *long edges* (from a node at level k to a node at level $h < k-1$, skipping levels $k-1, ..., h+1$) a BDD node encodes a boolean function of boolean variables. Three main options have been proposed: QBDDs [7], FBDDs [4], and ZBDDs [10]; in Fig. 1, the functions encoded by the nodes are $f_s = 0$, $f_t = x_1$, $f_u = 1$, $f_v = \overline{x_1}$, $f_q = x_1 \wedge x_2$, $f_r = \overline{x_1 \wedge x_2}$, and $f_p = (\overline{x_3} \wedge x_2 \wedge x_1) \vee (x_3 \wedge \overline{x_2 \wedge x_1})$.

To reflect the sharing of nodes and for consistency with our RexBDDs definition, we assume a BDD has a set $\mathcal{R}$ of *root edges*, each conceptually originating from level $L+1$ and encoding a function $\mathbb{B}^L \to \mathbb{B}$ of interest, where $\mathbb{B} = \{0, 1\}$. Thus, "the function encoded by BDD e" means that $\mathcal{R} = \{e\}$.

Definition 2. An L-variable OBDD is a *quasi-reduced BDD* (QBDD) if it has no *duplicates* (no nodes p and q at level $k > 0$ exist with $p[0] = q[0]$ and $p[1] = q[1]$) and no long edges: if node p is at level $k > 0$, then $p[0].lvl = p[1].lvl = k-1$. Function $f_p^{x_{[1,k]}} : \mathbb{B}^k \to \mathbb{B}$ encoded by QBDD node p at level k is:

$$f_p^{x_{[1,k]}} = \begin{cases} f_{p[x_k]}^{x_{[1,k-1]}} & \text{if } p.lvl > 0 \\ p & \text{if } p \text{ is node } \mathbf{0} \text{ or } \mathbf{1}. \end{cases} \quad \square$$

QBDD root edges are unlabeled and point to level-L nodes.

The work of the first three authors was supported in part by National Science Foundation under grant CCF-2212142.

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06
https://doi.org/10.1145/3649329.3656533

Definition 3. An L-variable OBDD is a *fully-reduced BDD* (FBDD) if it has no *duplicates* and no *redundant* nodes: no nonterminal node p has $p[0]=p[1]$. Function $f_p^{x_{[1,L]}}:\mathbb{B}^L\to\mathbb{B}$ encoded by FBDD node p at level k is:

$$f_p^{x_{[1,L]}} = \begin{cases} f_{p[x_k]}^{x_{[1,k-1]}} & \text{if } p.lvl > 0 \\ p & \text{if } p \text{ is node } \mathbf{0} \text{ or } \mathbf{1}. \ \square \end{cases}$$

FBDD root edges are unlabeled and point to nodes at any level; a long edge means that skipped variables are *don't-cares*. In Fig. 1, nodes s and u are redundant, thus removed, they encode the constants 0 and 1. FBDDs are effective for functions whose value often remains the same if some variable value flips.

Definition 4. An L-variable OBDD is a *zero-suppressed BDD* (ZBDD) if it has no *duplicates* and no *high-zero* nodes: no nonterminal node p has $p[1]=\mathbf{0}$. Function $f_p^{x_{[1,n]}}:\mathbb{B}^n\to\mathbb{B}$ encoded by ZBDD node p at level k w.r.t. a level $n\ge k$ is:

$$f_p^{x_{[1,n]}} = \begin{cases} \mathbf{0} & \text{if } n>k \wedge \exists h\in\{k+1,...,n\},\ x_h=1 \\ f_p^{x_{[1,k]}} & \text{if } n>k \wedge \forall h\in\{k+1,...,n\},\ x_h=0 \\ f_{p[x_k]}^{x_{[1,k-1]}} & \text{if } n=k>0 \\ p & \text{if } n=k=0. \ \square \end{cases}$$

ZBDD root edges are unlabeled and point to nodes at any level; a long edge means that the function has value 0 if any skipped variable has value 1. In Fig. 1, nodes s and v are high-zero, thus removed. ZBDDs are effective for *sparse* functions whose value often is 0 when some variable has value 1.

2.2 Canonicity, Efficiency, and Prior Proposals

QBDDs, FBDDs, and ZBDDs are *canonical*: for a fixed *variable order* mapping $x_1,...,x_L$ to levels, every function $f:\mathbb{B}^L\to\mathbb{B}$ has a *unique* representation in each of them.

A QBDD is always at least as large as the FBDD or ZBDD encoding the same function, but whether FBDDs or ZBDDs are best for an application largely depends on the number of redundant and high-zero nodes the QBDD would have. The size difference can be up to a factor of $L/2$, but the best choice may not be clear *a priori* and may even change as the computation evolves. Most importantly, choosing FBDDs means forgoing the savings of ZBDDs, and vice versa. This prompted BDD extensions integrating both reductions.

In 2017, van Dijk proposed two versions of *tagged BDDs* (tBDDs) [12], one where FBDD reductions are used from the source node to a "tag level" and ZBDD reductions after that until the target node, the other where ZBDD reductions are used first and FBDD reductions second; a node stores three levels, its own plus one for each outgoing edge. In 2018, Bryant proposed two versions of *chain-reduced BDDs* [5], cFBDDs, where FBDD nodes encode chains of high-zero nodes, and cZBDDs, where ZBDD nodes encode chains of redundant nodes; a node stores two levels, its own plus the one where the chain would have ended. For both chained and tagged BDDs, their two versions can result in different BDD sizes, and choosing between them is even more difficult (although less critical) than between FBDDs and ZBDDs. In 2019, Babar et al. proposed edge-specified reduction BDDs (ESRBDDs) [3], where long edges specify whether skipped levels mean redundant, high-zero, or *low-zero* nodes; in 2022 they extended them [2], to include *high-one* and *low-one* nodes and complement flags (discussed next).

2.3 Complement and Swap Flags

Further savings can be achieved using (output) *complement flags* [1] or (input complement) *swap flags* [11]. A complement flag c or a swap flag s is associated with each edge to node p, including root edges. If $c=1$, the value of f_p must be complemented; if $s=1$, the value of the variable x_k associated with p must be complemented prior to evaluating f_p, i.e., we swap the 0-child and 1-child of p.

If canonicity is enforced, complement or swap flags can each eliminate up to 50% of the nodes; together, up to 75% of the nodes. Canonicity conditions for complement flags were defined a decade later [6, 8], only for FBDDs: the complement flag of 0-edges must be 0 (thus, we only store the complement flag for 1-edges), terminal **1** is represented with a complement flag to terminal **0**, and only either p encoding f_p or $\overline{p}$ encoding $f_{\overline{p}}=\neg f_p$ may be present, specifically p if $f_p(0,...,0)=0$, $\overline{p}$ otherwise. Canonicity conditions for swap flags were also defined only for FBDDs: given a lexicographic order $\prec$ on edges, a nonterminal node p must satisfy $p[0]=\langle s_0,q_0\rangle \prec p[1]=\langle s_1,q_1\rangle$, i.e., either the children q_0 and q_1 are different and $q_0<q_1$ (e.g., based on their memory address) or $q_0=q_1=q$ and $s_0=0<s_1=1$. Note that node p is *not* redundant in the latter case even if its children coincide, because functions $f_{\langle 0,q\rangle}$ and $f_{\langle 1,q\rangle}$ are different: $f_{\langle 0,q\rangle}=f_q$ while $f_{\langle 1,q\rangle}=f_{\widetilde{q}}$, where $h=q.lvl$ and $f_{\widetilde{q}}(x_1,...,x_{h-1},x_h)=f_q(x_1,...,x_{h-1},\overline{x_h})$; finally, swap flags to terminal nodes are not meaningful, so they must be 0.

While not discussed in their original introduction, it is easy to see that complement or swap flags can also be used in QBDDs, with one caveat for the latter: if p is a redundant QBDD node with $p[0]=\langle s,q\rangle$ and $p[1]=\langle s,q\rangle$, then $p=\widetilde{p}$, and swap flags to p must be 0. FBDDs with *both* complement and swap flags were proposed in [11], but a subtle canonicity issue remained undiscovered for years [9]: consider a node p with $p[0]=\langle c_0,s_0,q\rangle$ and $p[1]=\langle c_1,s_1,q\rangle$, where c_i and s_i are the complement and swap flags on edge i. If $c_0=0$, $c_1=1$, and $s_0=s_1$, p is legal node, but complements or swap flags on edges to p have the same effect, i.e., $\overline{p}=\widetilde{p}$ and $p=\widetilde{\overline{p}}$. Thus, we arbitrarily forbid setting a swap flag to such nodes.

3 REXBDDS

3.1 Unreduced RexBDDs

We first introduce the semantics of unreduced RexBDDs.

Definition 5. An L-variable OBDD is a reduction-on-edge complement-and-swap BDD (RexBDD) with edges $\langle\rho,c,s,q\rangle$, where $\rho\in\{\mathsf{X},\mathsf{EL}_0,\mathsf{EL}_1,\mathsf{EH}_0,\mathsf{EH}_1,\mathsf{AL}_0,\mathsf{AL}_1,\mathsf{AH}_0,\mathsf{AH}_1\}$ is a *reduction rule*, $c\in\mathbb{B}$ is a *complement flag*, $s\in\mathbb{B}$ is a *swap flag*, and q is the target node. If $p[v]=\langle\rho,c,s,q\rangle$, we write $p[v].r=\rho$, $p[v].c=c$, $p[v].s=s$, $p[v].n=q$. Function $f_{\langle\rho,c,s,p\rangle}^{x_{[1,n]}}$ encoded by $\langle\rho,c,s,p\rangle$ w.r.t level $n\ge p.lvl$ is:

$$\begin{cases} \text{if } p.lvl=n=0, & c\oplus p \\ \text{if } p.lvl=n>0, & c\oplus f_{p[s\oplus x_n]}^{x_{[1,n-1]}} \\ \text{if } p.lvl=k<n\wedge\rho=\mathsf{X}, & f_{\langle\rho,c,s,p\rangle}^{x_{[1,k]}} \\ \text{if } p.lvl=k<n\wedge\rho=\mathsf{EL}_t, x_{k+1}\wedge\cdots\wedge x_n\ ?\ f_{\langle\rho,c,s,p\rangle}^{x_{[1,k]}}:t & \forall t\in\mathbb{B} \\ \text{if } p.lvl=k<n\wedge\rho=\mathsf{AL}_t, x_{k+1}\vee\cdots\vee x_n\ ?\ f_{\langle\rho,c,s,p\rangle}^{x_{[1,k]}}:t & \forall t\in\mathbb{B} \\ \text{if } p.lvl=k<n\wedge\rho=\mathsf{EH}_t, x_{k+1}\vee\cdots\vee x_n\ ?\ t:f_{\langle\rho,c,s,p\rangle}^{x_{[1,k]}} & \forall t\in\mathbb{B} \\ \text{if } p.lvl=k<n\wedge\rho=\mathsf{AH}_t, x_{k+1}\wedge\cdots\wedge x_n\ ?\ t:f_{\langle\rho,c,s,p\rangle}^{x_{[1,k]}} & \forall t\in\mathbb{B} \end{cases}$$

($\oplus$ means "exclusive-or", $\alpha?\beta:\gamma$ is β if α is true, else γ). $\square$

Thus, X means skipped variables are irrelevant along that path, EL_t means the function has value t if any skipped variable is 0, EH_t means the function has value t if any skipped variable is 1, AL_t means the function has value t if all skipped variables are 0, and AH_t means the function has value t if all skipped variables are 1 (see Fig. 2). Then, if $s=1$, the variable of the target nonterminal node q is complemented (s is irrelevant if q is a terminal node), and, if $c=1$, the value of the function encoded by q is complemented. All RexBDD root edges are also of the form $\langle \rho,c,s,q\rangle$.

3.2 Complementing a Function

In RexBDDs, complementing a function must extend to the meaning of the reduction rules attached to edges (including root edges).

Definition 6. The complement $\overline{\rho}$ of a rule ρ is defined as $\overline{\mathsf{X}}=\mathsf{X}$, $\overline{\mathsf{EL}_t}=\mathsf{EL}_{\overline{t}}$, $\overline{\mathsf{EH}_t}=\mathsf{EH}_{\overline{t}}$, $\overline{\mathsf{AL}_t}=\mathsf{AL}_{\overline{t}}$, $\overline{\mathsf{AH}_t}=\mathsf{AH}_{\overline{t}}$; that of an edge $e=\langle \rho,c,s,p\rangle$ as $\neg e=\langle \overline{\rho},\overline{c},s,p\rangle$. For convenience, we let $0\oplus\rho=\rho$, $1\oplus\rho=\overline{\rho}$, and $b\oplus e=\langle b\oplus\rho,b\oplus c,s,p\rangle$. □

The following theorem states that complementing an edge complements the function it encodes.

Theorem 1. For any edge $\langle \rho,c,s,p\rangle$ and any level $n\geq p.lvl$, $f^{x_{[1,n]}}_{\neg\langle \rho,c,s,p\rangle}=\neg f^{x_{[1,n]}}_{\langle \rho,c,s,p\rangle}$, thus $f^{x_{[1,n]}}_{b\oplus\langle \rho,c,s,p\rangle}=b\oplus f^{x_{[1,n]}}_{\langle \rho,c,s,p\rangle}$.

3.3 Challenges to RexBDD Canonicity

Interactions between RexBDD features pose canonicity challenges. We now examine each of them and its resolution.

Restrictions on terminal nodes, complement flags, swaps flags. With complement flags, canonicity allows only one terminal (we choose **0**, thus $\langle \rho,c,s,\mathbf{1}\rangle$ is changed into the equivalent $\langle \rho,\overline{c},s,\mathbf{0}\rangle$), and limits complement flags to one of the outgoing edges of a nonterminal node p (we require $p[0].c=0$). For FBDDs, if $p[0].c=1$, we store instead $\overline{p}$ i.e., we toggle the flags of p's two outgoing edges (and toggle the complement flag of any edge now pointing to $\overline{p}$, to preserve the value of any encoded function); for RexBDDs following the same idea is more complicated, as it must also take into account the use of swap flags on incoming edges and the presence of reduction rules on outgoing edges. With swap flags, canonicity forbids them on edges pointing to terminal nodes.

Meta-reductions for shorter edges. Function $f^{x_{[1,n]}}_{\langle \rho,c,s,q\rangle}$ encoded by $\langle \rho,c,s,q\rangle$ is independent of ρ if $q.lvl=n$. If nonterminal node p has a *short* edge $p[v]=\langle \rho,c,s,q\rangle$ to a nonterminal node q with $q.lvl=p.lvl-1>0$, we force ρ to be the *meta-rule* $\mathsf{N}=\{\mathsf{X},\mathsf{EL}_0,\mathsf{EL}_1,\mathsf{EH}_0,\mathsf{EH}_1,\mathsf{AL}_0,\mathsf{AL}_1,\mathsf{AH}_0,\mathsf{AH}_1\}$. If $p[v]$ skips instead one variable, i.e., $q.lvl=p.lvl-2>0$, then, for $t\in\mathbb{B}$, $\rho=\mathsf{EL}_t$ and $\rho=\mathsf{AL}_t$ have the same meaning and so do $\rho=\mathsf{EH}_t$ and $\rho=\mathsf{AH}_t$, thus we force $\rho\in\{\mathsf{X},\mathsf{L}_0,\mathsf{L}_1,\mathsf{H}_0,\mathsf{H}_1\}$, defining the meta-rules $\mathsf{L}_t=\{\mathsf{EL}_t,\mathsf{AL}_t\}$ and $\mathsf{H}_t=\{\mathsf{EH}_t,\mathsf{AH}_t\}$. Of course, the same restriction on using meta-reductions applies to root edges pointing to nodes at level L or $L-1$, and the definition of the complement $\overline{\rho}$ of a reduction rule ρ is extended to meta-rules by applying it to each element in the set of of equivalent rules.

Forbidden patterns. Just as FBDDs and ZBDDs, canonical RexBDDs forbid nodes representable by edges. Fig. 3 shows all such *node patterns* and the equivalent "replacement" edges, when $q\neq\mathbf{0}$, i.e., when at most one child of p is **0**. Having multiple rules in RexBDDs makes these replacements nontrivial. For example, if all edges

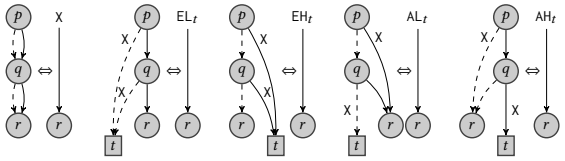

Figure 2: RexBDD rules when skipping two levels; $t\in\mathbb{B}$ and the missing reduction rules are N (explained later).

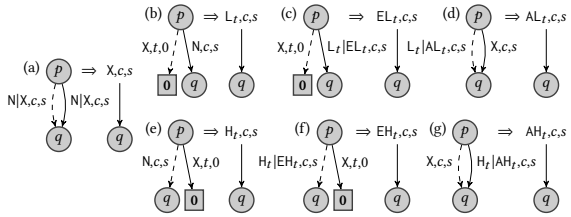

Figure 3: Forbidden patterns when at most one child of p is 0.

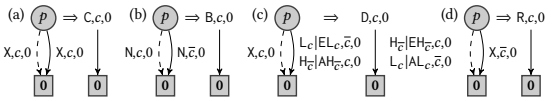

Figure 4: Forbidden patterns when both children of p are 0.

pointing to node p of Fig. 3(c) are of the form $\langle \mathsf{EL}_t,c',0,p\rangle$, we can replace them with edges $\langle \mathsf{EL}_t,c'\oplus c,s,q\rangle$ and remove node p, because edges $\langle \mathsf{EL}_t,c',0,p\rangle$ and $\langle \mathsf{EL}_t,c,s,q\rangle$ are "compatible". However, if even just one of them is, for example, of the form $\langle \mathsf{X},c',0,p\rangle$, then we cannot just bypass node p using a longer edge, because edges $\langle \mathsf{X},c',0,p\rangle$ and $\langle \mathsf{EL}_t,c,s,q\rangle$ are "incompatible": neither $\langle \mathsf{EL}_t,c'\oplus c,s,q\rangle$ nor $\langle \mathsf{X},c'\oplus c,s,q\rangle$ would encode the function encoded by $\langle \mathsf{X},c',0,p\rangle$.

Meta-reduction for patterns to terminal **0**. When both children of a node p at level k are **0**, the pattern describes a specific function of variables $x_{[1,k]}$, which may be equivalent to *multiple* long(er) edges, thus we represent it with special *meta-reductions* allowed only on edges pointing to **0**. Fig. 4 shows the four relevant patterns and their equivalent edges:

(a) encodes $c\in\mathbb{B}$, equivalent to meta-edge $\langle \mathsf{C},c,0,\mathbf{0}\rangle=\{\langle \mathsf{X},c,0,\mathbf{0}\rangle, \langle \mathsf{EL}_c,c,0,\mathbf{0}\rangle, \langle \mathsf{EH}_c,c,0,\mathbf{0}\rangle, \langle \mathsf{AL}_c,c,0,\mathbf{0}\rangle, \langle \mathsf{AH}_c,c,0,\mathbf{0}\rangle\}$, where "C" stands for "constant".

(b) applicable if $k=1$, encodes $c\oplus x_1$, equivalent to $\langle \mathsf{B},c,0,\mathbf{0}\rangle=\{\langle \mathsf{EL}_c,\overline{c},0,\mathbf{0}\rangle, \langle \mathsf{EH}_{\overline{c}},c,0,\mathbf{0}\rangle, \langle \mathsf{AL}_c,\overline{c},0,\mathbf{0}\rangle, \langle \mathsf{AH}_{\overline{c}},c,0,\mathbf{0}\rangle\}$, where "B" stands for "bottom variable".

(c) encodes $c\oplus\bigwedge_{h=1}^{k}x_h$, equivalent to $\langle \mathsf{D},c,0,\mathbf{0}\rangle=\{\langle \mathsf{EL}_c,\overline{c},0,\mathbf{0}\rangle, \langle \mathsf{AH}_{\overline{c}},c,0,\mathbf{0}\rangle\}$, where "D" stands for "and".

(d) encodes $c\oplus\bigvee_{h=1}^{k}x_h$, equivalent to $\langle \mathsf{R},c,0,\mathbf{0}\rangle=\{\langle \mathsf{EH}_{\overline{c}},c,0,\mathbf{0}\rangle, \langle \mathsf{AL}_c,\overline{c},0,\mathbf{0}\rangle\}$, where "R" stands for "or".

The restriction on meta-edges applies to root edges pointing to **0**, and the definition of complement extends to meta-edges.

Irrelevant swap flags. Unlike FBDDs, RexBDDs may have a nonredundant node p with $p[0]=p[1]=\langle \rho,c,s,q\rangle$, as long as $\rho\neq\mathsf{X}$; a swap flag on an edge to such p has no effect: $p=\widetilde{p}$, thus we require it to be 0.

Swap flags equivalent to complement flags As for FBDDs, a RexBDD node p with $p[0]=\langle \rho,0,s,q\rangle$ and $p[1]=\langle \overline{\rho},1,s,q\rangle$ satisfies $p=\widetilde{\overline{p}}$ and $\overline{p}=\widetilde{p}$, for *any* rule ρ; an edge to p may then have the complement flag set, but not the swap flag.

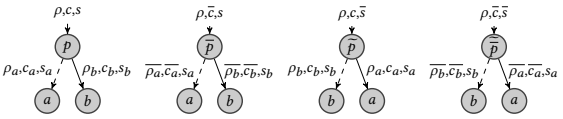

Figure 5: Four ways to represent the same function.

(a,s_a) vs (b,s_b)	c_a c_b	ρ_a vs ρ_b	Output	Notes
$<$	c any	any	$\langle$N,c,0,$c\oplus p\rangle$	
$>$	any c	any	$\langle$N,c,1,$c\oplus\widetilde{p}\rangle$	
$=$	c c	$c\oplus\rho_a \le c\oplus\rho_b$	$\langle$N,c,0,$c\oplus p\rangle$	if $\rho_a=\rho_b$, $p=\widetilde{p}$
$=$	c c	$c\oplus\rho_a > c\oplus\rho_b$	$\langle$N,c,1,$c\oplus\widetilde{p}\rangle$	if $\overline{\rho_a}=\overline{\rho_b}$, $\overline{p}=\widetilde{p}$
$=$	c $\overline{c}$	$c\oplus\rho_a \le \overline{c}\oplus\overline{\rho_b}$	$\langle$N,c,0,$c\oplus p\rangle$	if $\rho_a=\overline{\rho_b}$, $p=\widetilde{\overline{p}}$
$=$	c $\overline{c}$	$c\oplus\rho_a > \overline{c}\oplus\overline{\rho_b}$	$\langle$N,$\overline{c}$,1,$\overline{c}\oplus\widetilde{p}\rangle$	if $\overline{\rho_a}=\rho_b$, $\overline{p}=\widetilde{p}$

Figure 6: Decision logic to choose among p, $\overline{p}$, $\widetilde{p}$, and $\widetilde{\overline{p}}$.

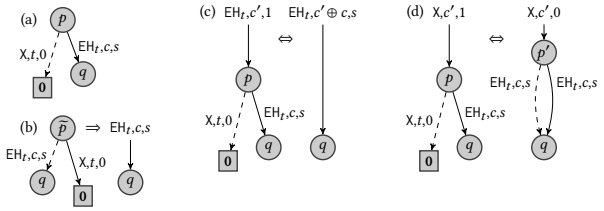

Figure 7: No swap flag s=1 on edges to p, if $\widetilde{p}$ is a pattern node.

Choosing between p, $\overline{p}$, $\widetilde{p}$, and $\widetilde{\overline{p}}$. RexBDDs must store only one of the four nodes p, $\overline{p}$, $\widetilde{p}$, and $\widetilde{\overline{p}}$ shown in Fig. 5, since the four edges $\langle\rho,c,s,p\rangle$, $\langle\rho,\overline{c},s,\overline{p}\rangle$, $\langle\rho,c,\overline{s},\widetilde{p}\rangle$, and $\langle\rho,\overline{c},\overline{s},\widetilde{\overline{p}}\rangle$ encode the same function $f^{x_{[1,n]}}_{\langle\rho,c,s,p\rangle}$ w.r.t. level $n \ge p.lvl$.

Fig. 6 shows how we choose: given $\langle$N,0,0,$p\rangle$, with $p[0]$=$\langle\rho_a,c_a,s_a,q_a\rangle$ and $p[1]$ = $\langle\rho_b,c_b,s_b,q_b\rangle$, we return canonical edge $\langle$N,c,s,$p'\rangle$ encoding the same function; the c and s flags identify the node p' representing the *normalization* of p. This includes cases where the four nodes collapse into two.

Swap flag to an edge pointing to the swap of a pattern node. Finally, we discuss a subtle exception to the decision logic of Fig. 6. Consider node p (legal, assuming $\mathbf{0} < q$ for any nonterminal node q) in Fig. 7(a) with $p[0]$ = $\langle$X,t,0,$\mathbf{0}\rangle$ and $p[1]$ = $\langle$EH$_t$,c,s,$q\rangle$, so that $\widetilde{p}$ is an EH$_t$-pattern node encoded by edge $\langle$EH$_t$,c,s,$q\rangle$, Fig. 7(b). Edge $\langle$EH$_t$,c',1,$p\rangle$ would then encode the same function as $\langle$EH$_t$,$c'\oplus c$,s,$q\rangle$, both cases shown in Fig. 7(c), and both would be legal given the restrictions listed so far. Thus, we forbid an edge with swap flag set to 1 from pointing to node p, if $\widetilde{p}$ is a pattern node.

If a node p is such that $\widetilde{p}$ is a pattern node, a further problem arises: it may be impossible to normalize it. For example, p in Fig. 7(a) is not normalized if $t = 1$, in which case we complement it by changing it into $\overline{p}$; then $p[0].n = \mathbf{0} < p[1].n = q$, thus $\overline{p}$ is normalized. However, had p been the swap of the EL$_t$ pattern node of Fig. 3(c), we could still complement it, but not normalize it, as $p[0].n = q > p[1].n = \mathbf{0}$, and we cannot swap it to put its outgoing edges in the right order because the result would be a pattern node. In other words, we can always have $p[0].c = 0$, but the full decision procedure of Fig. 6 must be applied only if $\widetilde{p}$ is not a pattern node.

3.4 Reduced RexBDDs and Their Canonicity

Reduced RexBDDs address all canonicity challenges just discussed. In addition to the nine reduction rules of unreduced RexBDDs, the

x_2	x_1	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

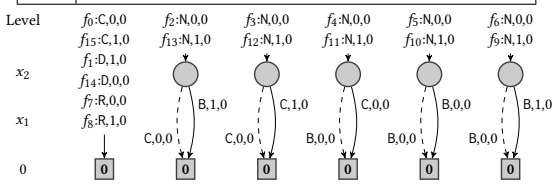

Figure 8: RexBDDs encoding all functions of two variables.

definition uses the meta-reductions N, L$_0$, L$_1$, H$_0$, and H$_1$ for *certain* edges pointing to nonterminal nodes, and the meta-reductions C, B, D, or R for *all* edges pointing to terminal **0**. However, an edge employing these meta-reductions is equivalent to a specific set of equivalent edges (one of these equivalent edges is consistently used in our implementation, which then requires only nine reduction rule "codes").

Definition 7. An L-variable RexBDD is reduced if it has:

- No terminal **1**.
- No duplicate nodes.
- For any edge pointing to **0**, rule C, B, D, or R.
- For any edge pointing to a nonterminal node,
  - rule N if the edge does not skip any level;
  - rule X, L$_t$, or H$_t$, $t\in\mathbb{B}$, if it skips one level;
  - rule X, EL$_t$, EH$_t$, AL$_t$, or AH$_t$, $t\in\mathbb{B}$, otherwise.
- No C, B, D, R, X, EL$_t$, EH$_t$, AL$_t$, or AH$_t$, $t\in\mathbb{B}$, pattern nodes.
- No swap flag $s = 1$ on edges to **0** or to a nonterminal p if $p = \widetilde{p}$, if $\overline{p} = \widetilde{p}$, or if $\widetilde{p}$ is a pattern node.
- No nonterminal node p with $p[0].c = 1$.
- No nonterminal node p with $p[0].n > p[1].n$, or $p[0].n = p[1].n$, $p[0].s$=1, and $p[1].s$=0, or $p[0].n = p[1].n$, $p[0].s = p[1].s$, and $p[0].r > p[1].c\oplus p[1].r$, for arbitrary orders on nonterminal nodes and terminal node **0**, and on the reduction rules, unless $\widetilde{p}$ is a pattern node. □

In RexBDD terminology, FBDDs and ZBDDs correspond to forbidding duplicate nodes and X or EH$_0$ pattern nodes, respectively. We now assume all RexBDDs are reduced.

Fig. 8 shows the canonical RexBDD encoding of all 2^{2^2}=16 functions of two boolean variables x_1 and x_2. Unlike QBDDs, FBDDs, and ZBDDs, RexBDDs have *no nodes at level* 1.

We now state the following theorems (proof omitted).

Theorem 2. RexBDDs are universal: given $g : \mathbb{B}^L \to \mathbb{B}$, there is a RexBDD edge $\langle\rho,c,s,p\rangle$ such that $f^{x_{[1,L]}}_{\langle\rho,c,s,p\rangle} = g$.

Theorem 3. RexBDDs are canonical: given level n and edges $\langle\rho,c,s,q\rangle$, $\langle\rho',c',s',q'\rangle$ s.t. $\max\{q.lvl, q'.lvl\} \le n$, if $f^{x_{[1,n]}}_{\langle\rho,c,s,q\rangle} = f^{x_{[1,n]}}_{\langle\rho',c',s',q'\rangle}$, then (1) $c=c'$, $s=s'$, $q=q'$ and (2) if $q.lvl < n$, then $\rho=\rho'$.

Theorem 4. Let the prefix C, S, or CS indicate QBDDs or FBDDs with complement flags, swap flags, or both. Then, the number $\overline{V}$ of nodes needed to encode any given function $f : \mathbb{B}^L \to \mathbb{B}$ using BDD variant VBDD satisfies: $\overline{\text{Q}} \ge \overline{\text{Z}} \ge \overline{\text{Rex}}$,
$\overline{\text{Q}} \ge \overline{\text{CQ}} \ge \overline{\text{CSQ}}$, $\overline{\text{Q}} \ge \overline{\text{SQ}} \ge \overline{\text{CSQ}}$, $\overline{\text{F}} \ge \overline{\text{CF}} \ge \overline{\text{CSF}}$, $\overline{\text{F}} \ge \overline{\text{SF}} \ge \overline{\text{CSF}}$,
$\overline{\text{Q}} \ge \overline{\text{F}}$, $\overline{\text{CQ}} \ge \overline{\text{CF}} \ge \overline{\text{Rex}}$, $\overline{\text{SQ}} \ge \overline{\text{SF}} \ge \overline{\text{Rex}}$, $\overline{\text{CSQ}} \ge \overline{\text{CSF}}$, $\overline{\text{CSQ}} \ge \overline{\text{Rex}}$.

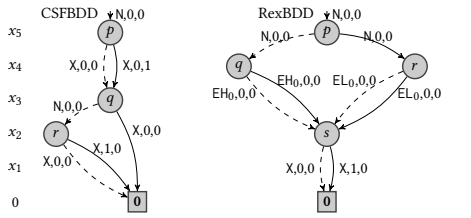

Figure 9: CSFBDD and RexBDD for $((\neg x_5 \wedge \neg x_3) \vee (x_5 \wedge x_3)) \wedge x_2$.

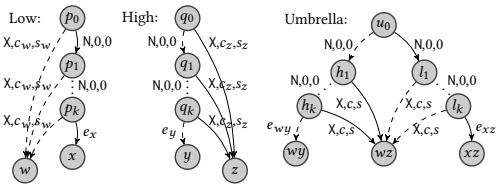

Figure 10: Patterns in a binary Apply operation on RexBDDs.

Note that we may have $\overline{\text{CSF}} < \overline{\text{Rex}}$: Fig. 9 shows how this can happen (such pathological case arises also in CESRBDDs).

3.5 RexBDD Elementwise Operations

Element-wise operations, such as *AND*, *OR*, and *NOT*, can be implemented using modified versions of the BDD Apply algorithm [4], for each of the BDD variants. Given any function f encoded by edge e, building $\bar{f}$ is an $O(1)$ operation for variants with complement flags: simply toggle the complement flag of e and, in the case of CESRBDDs and RexBDDs, also complement the reduction rule. For the other variants, we must instead build the complement of all nodes reached from e; letting their number be $|e|$, this requires $O(|e|)$ time. ZBDDs may require additional computation, as any long edges must be expanded into a chain of high-one nodes.

For binary operations, such as *AND*, the Apply algorithm simultaneously traverses both argument BDDs, and builds the result, using a memorization table to avoid duplicated computation. To fully exploit long edges in variants with different reduction rules, in particular RexBDDs, one must consider all possible pairs of edge reductions. This can be greatly simplified, as every long edge corresponds to either the *Low* or *High* pattern of Fig. 10: $\texttt{EL}_0$, $\texttt{EL}_1$, $\texttt{AH}_0$, and $\texttt{AH}_1$ fit the *Low* pattern, $\texttt{EH}_0$, $\texttt{EH}_1$, $\texttt{AL}_0$, and $\texttt{AL}_1$ fit the *High* pattern, and X fits both patterns (with $w = x$ or $y = z$). Then, we only need to consider three combinations: the *AND* of two *Low* patterns, the *AND* of two *High* patterns, and the *AND* of a *Low* and *High* pattern. The first two produce a pattern of the same type, where the bottom nodes and edges are obtained by recursively building the *AND* of the matching edges on the inputs; for example, the *AND* of two *Low* patterns builds $w = w_1 \wedge w_2$ and $e_x = e_{x_1} \wedge e_{x_2}$. The bottoms of the repeating pattern nodes (nodes p_k or q_k in the figure) must be at the same level; if necessary, patterns are broken to ensure this. The produced pattern may or may not correspond to a long edge; if not, the nodes in the pattern must be built (for example, if both w and x are non-terminal nodes). Building the *AND* of a *Low* and *High* pattern produces, instead, the "umbrella" pattern of Fig. 10, where the h nodes are a *High* pattern, and the l nodes are a *Low* pattern.

4 EXPERIMENTAL RESULTS

We implemented reduced RexBDDs in a prototype library, which we use to compare the size of (C/S/CS)QBDDs/FBDDs, ZBDDs, ESRBDDs, CESRBDDs (obtained by selectively restricting flags or reductions in our prototype), and RexBDDs when encoding all 2^{32} functions of five variables. We do not consider chained or tagged BDDs, as they never experimentally outperform CESRBDDs in [2]. Then, we compare these variants on a set of common benchmarks with many variables, to highlight the advantages of RexBDDs on larger applications. All required elementwise operations are implemented using AND and NOT, i.e., $f \vee g$ is computed as $\neg(\neg f \wedge \neg g)$.

We count BDD nodes to be implementation-independent. All nodes store "pointers" to their children and to the "next" node in the unique table; non-quasi-reduced variants store an integer level; complement, swap, or both flags require 1, 2, or 3 bits; n reductions require $\lceil \log_2 n^2 \rceil$ bits (4 for ESRBDDs, 5 for CESRBDDs, 7 for RexBDDs). Our index-based prototype "packs" all this information in 192 bits (24 bytes, the same as three 64-bit pointers), so the additional 10 bits required by RexBDDs w.r.t FBDDs or ZBDDs are "free".

4.1 Encoding All Functions of Five Variables

Fig. 11 reports the number of nonterminal nodes per level, and in total, for each BDD variant. Either complement or swap flags reduce the *total* number of nodes required to encode *all* 2^{32} functions by almost 50%, both flags reduce them by almost 75%, while the reduction choice (Q, F, Z, or edge-specified) has minimal impact. However, the *average* number of nodes to encode *one* generic function shows major differences: w.r.t. QBDDs, no-flag variants, FBDDs, ZBDDs, and ESRBDDs, require resp. 22%, 22%, and 36% fewer nodes; one-flag variants, CQBDDs, SQBDDs, CFBDDs, SFBDDs, and CESRBDDs, require resp. 18%, 11%, 33%, 33%, and 43% fewer nodes; both-flag variants, CSQBDD, CSFBDDs, and RexBDD, require resp. 23%, 38%, and 47% fewer nodes.

Fig. 13 reports more detailed distributional data: the number of functions encoded using 0,1,...,19 nonterminal nodes, for each BDD variant. RexBDDs are the best: only RexBDDs and CSFBDDs require at most 12 nodes, but more functions require 12 nodes with CSFBDDs than with RexBDDs.

4.2 Results from Common Benchmarks

First, we report results from encoding (together) all output bits for circuits in IWLS'93 benchmark `https://ddd.fit.cvut.cz/www/prj/Benchmarks/` using our 12 BDD variants and the variable order in the `blif` input file, on a MacBook Pro with an 8-core M1 Pro chip and 16GB RAM. Of these, we eliminated 5 "huge" circuits requiring >500M peak nodes for QBDDs (`C2670`, `C3540`, `C6288`, `C7552`, and `i10`) and 42 "small" circuits requiring <10K peak nodes for QBDDs (runtimes less than 0.06 seconds for QBDDs, 0.01 seconds for RexBDDs), leaving 27 "large" circuits (runtimes from 0.01 to 30 seconds for QBDDs, from 0.005 to 4.3 seconds for RexBDDs). Fig. 12 shows the geometric mean score for each variant:

$$score(V) = \sqrt[27]{\prod_{n=1}^{27} \rho_V(n) / \min\{\rho_{\text{Q}}(n), ..., \rho_{\text{Rex}}(n)\}},$$

where $\rho_V(n)$ is the final or peak number of nodes, or seconds, to generate all outputs of circuit n with variant V. $score(V) = 1$ if variant V is *always* the best. Fig. 12 also has results for huge circuit

Level	QBDD	CQBDD	SQBDD	CSQBDD	FBDD	CFBDD	SFBDD	CSFBDD	ZBDD	ESRBDD	CESRBDD	RexBDD
1	4	2	3	2	2	1	1	1	2	**0**	**0**	**0**
2	16	8	10	6	12	6	6	**4**	12	12	6	5
3	256	128	136	72	240	120	120	64	240	216	96	**56**
4	65,536	32,768	32,896	16,512	65,280	32,640	32,640	16,384	65,280	64,848	32,256	**16,206**
5	4,294,967,296	2,147,483,648	2,147,516,416	1,073,774,592	4,294,901,760	2,147,450,880	2,147,450,880	1,073,741,824	4,294,901,760	4,294,772,064	2,147,321,857	**1,073,677,827**
Total	4,295,033,108	2,147,516,554	2,147,549,461	1,073,791,184	4,294,967,294	2,147,483,647	2,147,483,647	1,073,758,277	4,294,967,294	4,294,837,140	2,147,354,215	**1,073,694,094**
Avg.	17.389	14.204	15.486	13.338	13.540	11.637	11.637	10.771	13.540	11.174	9.872	**9.305**

Figure 11: Number of nodes to encode all functions of 5 variables; average nodes to encode one of these functions. Bold is best.

	QBDD	CQBDD	SQBDD	CSQBDD	FBDD	CFBDD	SFBDD	CSFBDD	ZBDD	ESRBDD	CESRBDD	RexBDD
score final nodes	5.77	5.27	5.38	4.89	1.39	1.29	1.27	1.17	5.51	1.30	1.10	**1.01**
score peak nodes	8.59	7.19	8.00	6.74	1.40	1.18	1.27	1.08	8.31	1.32	1.08	**1.01**
score runtime	9.96	9.47	9.98	9.20	2.28	2.26	2.29	2.08	9.62	2.51	2.11	**1.00**
`C2670` final nodes	—	259,311,327	—	259,055,555	33,013,096	18,975,054	32,501,745	**18,719,282**	—	32,915,387	18,877,279	18,752,067
`C2670` peak nodes	>500M	407,427,217	>500M	407,164,516	43,218,281	23,728,393	42,699,096	**23,465,692**	>500M	43,105,063	23,615,049	23,495,218
`C2670` seconds	—	20,570	—	19,698	3,183	998	2,992	976	—	2,961	969	**862**
DictBinComp L =144	1,105,092	1,105,091	1,072,081	1,072,081	1,104,755	1,104,754	1,071,744	1,071,744	651,993	461,155	461,156	**460,971**
DictBinFull L =168	1,267,787	1,267,786	1,235,315	1,235,315	1,267,399	1,267,398	1,234,927	1,234,927	844,240	516,408	516,408	**516,231**
Dict1hotComp L =1,272	9,700,754	9,700,754	9,668,809	9,668,809	9,699,268	9,699,268	9,667,323	9,667,323	**300,271**	**300,271**	**300,271**	**300,271**
Dict1hotFull L =3,048	22,982,853	22,982,853	22,950,908	22,950,908	22,979,590	22,979,590	22,947,645	22,947,645	**300,271**	**300,271**	**300,271**	**300,271**

Figure 12: Scores for the 69 IWLS'93 combinational circuits. Detailed results for `C2670`. Dictionary results. Bold is best.

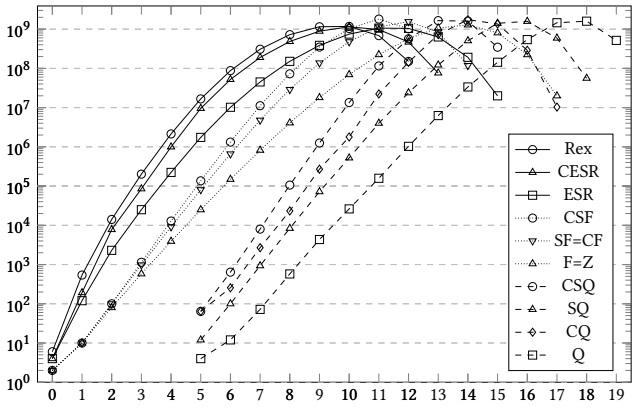

Figure 13: Number (y-axis, in logscale) of functions of five variablesencoded with 0,...,19 **(x-axis) nonterminal nodes.**

`c2670`; here, too, RexBDDs excel, requiring slightly more final (0.2%) and peak (0.1%) nodes than CSFBDDs, but 11.7% less runtime.

Then, we report on encoding the English words in MacOS file `/usr/share/dict/words`, as done in [5]: 235,886 words of length from 1 to 24, using either the full 128 ASCII alphabet, or a compact alphabet of 54 characters, and either a binary or a one-hot encoding. Fig. 12 shows the number of levels L and the number of nodes required by each variant. RexBDDs always require the fewest nodes. Interestingly, the one-hot encoding requires more levels but has ZBDDs, ESRBDDs, CESRBDDs, and RexBDDs all tied for the fewest nodes, regardless of the alphabet; however, when encoding the complement of this set, all variants require exactly the same number of nodes as before, except for ZBBDs and ESRBDDs, which become as bad as QBDDs (not shown).

5 CONCLUSIONS

We introduced RexBDDs, which canonically combine nine reductions (including those of the two most well-known BDD variants, FBDDs and ZBDDs) with complement and swap flags. Benchmark experiments confirm that RexBDDs encode many boolean functions using substantially fewer nodes than any other BDD variant, and tend to have much better runtimes. Once completed, our prototype RexBDD library will be a plug-and-play replacement to any BDD library, as users interact with it by simply creating and manipulating the boolean functions required by their specific application. Users can then enjoy the benefits of both FBDD and ZBDD reductions, without needing to decide between them *a priori*.

REFERENCES

[1] S. B. Akers. 1978. Functional testing using binary decision diagrams. In *Proc. 8th Int. Symp. on Fault-Tolerant Computing*. 75–82.

[2] Junaid Babar, Gianfranco Ciardo, and Andrew Miner. 2022. CESRBDDs: binary decision diagrams with complemented edges and edge-specified reductions. *Software Tools for Technology Transfer* 24 (Feb. 2022), 89–109. https://doi.org/10.1007/s10009-021-00640-0

[3] Junaid Babar, Chuan Jiang, Gianfranco Ciardo, and Andrew Miner. 2019. Binary decision diagrams with edge-specified reductions. In *Proc. TACAS*. Springer, 303–318. https://doi.org/10.1007/978-3-030-17465-1_17

[4] Randy E. Bryant. 1986. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comp.* 35, 8 (Aug. 1986), 677–691.

[5] Randal E. Bryant. 2018. Chain Reduction for Binary and Zero-Suppressed Decision Diagrams. In *Proc. TACAS*. Springer, 81–98.

[6] Kevin Karplus. 1988. *Representing boolean functions with if-then-else DAGs*. Technical Report. University of California at Santa Cruz.

[7] Shinji Kimura and Edmund M. Clarke. 1990. A parallel algorithm for constructing binary decision diagrams. In *Proc. ICCD*. IEEE CS Press, 220–223.

[8] Jean-Christophe Madre and Jean-Paul Billon. 1988. Proving circuit correctness using formal comparison between expected and extracted behaviour. In *Proc. ACM/IEEE DAC* (Atlantic City, New Jersey, USA). IEEE CS Press, 205–210.

[9] D.M. Miller and R. Drechsler. 1997. Negation and duality in reduced ordered binary decision diagrams. In *1997 IEEE Pacific Rim Conf. Communications, Computers and Signal Processing*, Vol. 2. 692–696. https://doi.org/10.1109/PACRIM.1997.620354

[10] S. Minato. 2001. Zero-suppressed BDDs and their applications. *Software Tools for Technology Transfer* 3 (2001), 156–170.

[11] S. Minato, N. Ishiura, and S. Yajima. 1990. Shared binary decision diagram with attributed edges for efficient boolean function manipulation. In *Proc. ACM/IEEE DAC*. IEEE CS Press, 52–57.

[12] Tom van Dijk, Robert Wille, and Robert Meolic. 2017. Tagged BDDs: combining reduction rules from different decision diagram types. In *Proc. FMCAD* (Vienna, Austria). 108–115.