Calculemus 2005 Preliminary Version

Symbolic Reasoning with Weighted and Normalized Decision Diagrams

Jörn Ossowski, Christel Baier

Universität Bonn, Institut für Informatik I,
Römerstrasse 164, 53117 Bonn, Germany
{ossowski,baier}@cs.uni-bonn.de

Abstract

Several variants of Bryant's ordered binary decision diagrams have been suggested in the literature to reason about discrete functions. In this paper, we introduce a generic notion of *weighted decision diagrams* that captures many of them and present criteria for canonicity. As a special instance of such weighted diagrams, we introduce a new BDD-variant for real-valued functions, called *normalized algebraic decision diagrams*. Regarding the number of nodes and arithmetic operations like addition and multiplication, these normalized diagrams are as efficient as factored edge-valued binary decision diagrams, while several other operators, like the calculation of extrema, minimum or maximum of two functions or the switch from real-valued functions to boolean functions through a given threshold, are more efficient for normalized diagrams than for their factored counterpart.

Key words: Weighted decision diagrams, factored edge-valued binary decision diagrams, normalized algebraic decision diagrams, minimum/maximum calculation, solving linear equation systems

1 Introduction

Ordered binary decision diagrams (BDDs) [5,6] are data structures to represent boolean functions $f : \{0,1\}^n \rightarrow \{0,1\}$ that rely on a compactification of binary decision trees. Several modifications of BDDs have been proposed to reason about discrete functions over a finite domain, such as real-valued matrices, weighted automata or graphs with distance or capacity functions [1]. Multi-terminal BDDs (also called algebraic DDs) [1,8,11] use sinks with arbitrary values (rather than just 0 and 1 as it is the case of ordinary BDDs), (factored) edge-valued BDDs [27,26] allow additional edge attributes, while multi-valued decision diagrams [17,24] use multiple branches to represent the possible values of the input variables. Although none

[1] Ordinary BDDs can also serve as data structure for functions $f : I \rightarrow O$ with finite domain I and finite range O by using appropriate binary encodings for the input and output values.

of the above mentioned BDD-variants can avoid an exponential-sized representation for certain functions, BDDs and their variants have been proven very successful for verification purposes [7,20,19], representation and analysis of probabilistic systems [12,2,4,18], combinatorial problems [16], integer linear programming [1,26], stochastic planning [15] and many more application areas (see e.g. the text books [13,16,21,10,9,28]). For many space critical applications where the explicit representation is not presentable anymore, the symbolic representation with BDDs can expand the presentable limit.

The purpose of this paper is twofold. First, we consider a generic notion of *weighted decision diagram* (WDD for short) for the representation of functions $f : \{0,1\}^n \to \mathbb{K}$ (that map bit-vectors into an arbitrary set $\mathbb{K}$) and present a general framework for WDDs to reason about *canonicity*. WDDs are variants of multi-terminal BDDs where the edges are augmented with certain "weights" which are formalized by bijections $\mathbb{K} \to \mathbb{K}$. Depending on the chosen type of the bijections Φ and the range $\mathbb{K}$, WDDs specialize to several known BDD-types.

In the second part of our paper, we introduce a new type of real-valued weighted decision diagrams, so called *normalized algebraic decision diagrams* (NADDs). As in factored edge-valued BDDs, the edges of a NADD are augmented with pairs (a,b) where a is a multiplicative weight and b an additive weight. NADDs and FEVBDDs differ in the underlying reduction criteria. Whereas FEVBDDs (with the so-called rational rule) rely on a representation where only the edges to the 1-successors can have a non-trivial weight, the inner nodes of a NADD stand for real-valued functions with minimum value 0 and maximum value 1. From our generic considerations for WDDs, we may conclude that NADDs and FEVBDDs for the same functions and variable ordering have the same size. FEVBDDs just need to store two parameters for both successors of a node because the edge-values for the 0-successor are fixed. Although none of the edge-attributes of NADDs are fixed, the normalization condition of NADDs enables a similar space-efficient representation of NADD-nodes, such that NADDs and FEVBDDs are almost of equal memory usage. Arithmetic operations, like addition and multiplication, can be realized with normalized and factored edge-valued diagrams by similar procedures. However, the calculation of minima and maxima can be performed in a NADD in constant time, while it requires a graph-traversal in the case of MTBDDs and FEVBDDs. Thus, NADDs support the calculation for the minimum or maximum of two functions or the switch from a real-valued function f to a boolean function that is obtained from f via a certain threshold in a more natural way. For instance, the latter is needed in the context of model checking probabilistic systems against formulas of a probabilistic branching-time logic like PCTL [14,3] where one first has to calculate a probability vector, regarded as a function $f : \{0,1\}^n \to [0,1]$, which is then replaced with a boolean vector respresenting, e.g., the set $\{s \in \{0,1\}^n : f(s) \geq p\}$ where $p \in]0,1[$ is a lower bound for the "acceptable" probabilities.

Organization of the paper. The basic concepts of binary decision diagrams and notations used in this paper are summarized in Section 2. Weighted decision diagrams are studied in Section 3. In Section 4, we introduce normalized algebraic decision diagrams and illustrate their implementation and efficiency, while Section 5 concludes the paper.

2 Binary decision diagrams

In this section, we briefly recall the basic concepts of multi-terminal BDDs and explain our notations. Further details can be found e.g. in the text books [21,28].
Variables. In the following, we fix a finite set $\mathcal{Z} = \{z_1, \ldots, z_n\}$ of boolean variables. An evaluation for $\mathcal{Z}$ denotes a function that assigns a boolean value to any variable $z_i \in \mathcal{Z}$. $Eval(\mathcal{Z})$ or $Eval(z_1, \ldots, z_n)$ denotes the set of all evaluations for $\mathcal{Z}$.
Switching functions. If $\mathbb{K}$ is a set then a $\mathbb{K}$-valued switching function means a function $f : Eval(\mathcal{Z}) \to \mathbb{K}$. Assuming a fixed enumeration $z_1, \ldots, z_n$ for the variables in $\mathcal{Z}$, any $\mathbb{K}$-valued switching function can be viewed as a function $f : \{0,1\}^n \to \mathbb{K}$. This allows simplified notations such as $f(\xi_1, \ldots, \xi_n)$ for the function value of the evaluation that assigns value $\xi_i \in \{0,1\}$ to variable z_i. The case $\mathbb{K} = \{0,1\}$ yields ordinary switching functions that return the boolean values 0 or 1 for any evaluation. In the sequel, we simply speak about "$\mathbb{K}$-valued functions" or just "functions" to denote functions of the type $f : Eval(\mathcal{Z}) \to \mathbb{K}$. If $z \in \mathcal{Z}$ then $f|_{z=0}$ and $f|_{z=1}$ denote the *cofactors* of f which arise by fixing the assignment $z \mapsto 0$ and $z \mapsto 1$ respectively. E.g., if $f = z_1 + 3z_2$ then $f|_{z_1=0} = 3z_2$ and $f|_{z_1=1} = 1 + 3z_2$. Variable z is called *essential* for f if $f|_{z=0} \neq f|_{z=1}$.
Binary decision diagrams (BDD) are a graph based data structure boolean functions which rely on the decomposition of boolean functions in their cofactors according to the *Shannon expansion* $f = (\neg z \wedge f|_{z=0}) \vee (z \wedge f|_{z=1})$. We shall consider here the multi-terminal variant [1,8,11] for representing e.g. real-valued functions in which case the Shannon expansion corresponds to $f = (1-z) \cdot f|_{z=0} + z \cdot f|_{z=1}$. Formally, a MTBDD for a $\mathbb{K}$-valued switching function is an acyclic rooted directed graph where every inner node v is labeled with a variable and has two children, called the 0-successor and 1-successor, denoted by $succ_0(v)$ and $succ_1(v)$. The terminal nodes are labeled by values in $\mathbb{K}$. In ordered (MT)BDDs [5], there is a variable ordering π which is preserved on any path from the root to a terminal node. That is, if v is an inner node labeled with variable z_i and w a child of v which is non-terminal and labeled with variable z_j then z_i appears in π before z_j, i.e., $i < j$ if $\pi = (z_1, \ldots, z_n)$. In the sequel, we shall use the notation π-MTBDD, or briefly MTBDD, to denote an ordered MTBDD relying on the ordering π and we refer to any inner node labeled with variable z as a z-node. The size $|\mathcal{B}|$ of $\mathcal{B}$ means the number of nodes in $\mathcal{B}$. The function represented by a terminal node agrees with the corresponding constant. The function f_v of a z-node v is $f_v = (1-z) \cdot f_{succ_0(v)} + z \cdot f_{succ_1(v)}$ (where we assume that $\mathbb{K} = \mathbb{R}$ or any other semi-ring). The function $f_{\mathcal{B}}$ represented by an MTBDD $\mathcal{B}$ agrees with the function for its root node. Thus, given an evaluation for $\mathcal{Z}$, the value under $f_{\mathcal{B}}$ is obtained by traversing $\mathcal{B}$ starting in its root and branching in any inner node according to the given evaluation. A MTBDD $\mathcal{B}$ is called reduced if the nodes in $\mathcal{B}$ represent pairwise distinct functions, i.e., $f_v = f_w$ implies $v = w$.

For fixed ordering π for the variable set $\mathcal{Z}$, π-MTBDDs yield a universal representation for functions $f : Eval(\mathcal{Z}) \to \mathbb{K}$. Moreover, the representation by π-MTBDDs is unique up to isomorphism. However, changing the variable ordering might result in a MTBDD of totally different structure and size [2].

3 Weighted decision diagrams

While MTBDDs yield a quite compact representation for functions with small domain and many symmetries, they cannot avoid an exponential-sized representation for injective functions. A more efficient representation can be obtained by allowing weights for the edges as it is the case for edge-valued BDDs (EVBDD) [27] and factored edge-valued BDDs (FEVBDD) [26]. EVBDDs attach additive weights to the edges, while the factored variants assign both multiplicative and additive weights to the edges. For such decision diagrams with attributed edges, canonicity and reducedness is less trivial than for ordinary MTBDDs and requires additional constraints. We will now consider a generic notion of weighted decision diagrams which (among others) subsumes MTBDDs and their (factored) edge-valued variants and normalized decision graphs that will be studied in Section 4.

Notation 3.1 In the sequel, let $\mathcal{Z}$ be a finite set of variables, $\mathbb{K}$ a set with at least two elements [3], and let $\mathbb{F}$ denote the set of functions $f : Eval(\mathcal{Z}) \to \mathbb{K}$. $\Phi_{\mathbb{F}}$ denotes a nonempty set of bijections $\mathbb{K} \to \mathbb{K}$ such that

(1) $\Phi_{\mathbb{F}}$ is closed under inversion and composition, i.e., if $\varphi, \psi \in \Phi_{\mathbb{F}}$ then $\varphi^{-1} \in \Phi_{\mathbb{F}}$ and $\varphi \circ \psi \in \Phi_{\mathbb{F}}$. (In particular, $\Phi_{\mathbb{F}}$ contains the identity id.)

(2) If $f \in \mathbb{F}$, $\varphi \in \Phi_{\mathbb{F}}$ and $f = \varphi \circ f$ then $\varphi = id$.

In our notion of weighted decision diagrams the edges will be augmented with functions $\varphi \in \Phi_{\mathbb{F}}$ that serve as transformations. The idea is that any φ-labelled edge to (a node for) a function f stands for the function $\varphi \circ f$. Condition (2) will be important for the uniqueness of the function representation. Note that if $\varphi, \psi \in \Phi_{\mathbb{F}}$, $f \in \mathbb{F}$ and $\varphi \circ f = \psi \circ f$ then $\varphi = \psi$ (as we have $f = (\varphi^{-1} \circ \psi) \circ f$, and hence, $\varphi^{-1} \circ \psi = id$ by (2)).
In some BDD-variants with weighted edges, the constant functions require a special treatment as there might be several possibilities to transform a constant $c \in \mathbb{K}$ into another constant $c' \in \mathbb{K}$ via the bijections $\varphi \in \Phi_{\mathbb{F}}$. Therefore, we split $\mathbb{F}$ into the sets $\mathbb{F}^{const}$ and $\mathbb{F}^{non\text{-}const}$ of constant and non-constant functions in $\mathbb{F}$, respectively, and assume that we are given sets of transformations $\Phi = \Phi_{\mathbb{F}^{non\text{-}const}}$ for the non-constant functions in $\mathbb{F}$ and Φ^{const} for the constant functions in $\mathbb{F}$, such that Φ and Φ^{const} fulfill conditions (1) and (2) in Notation 3.1.

[2] In fact, there are functions that have polynomial sized MTBDDs under "good" orderings and MTBDDs of exponential size for "bad" orderings.

[3] The requirement $\mathbb{K}$ to be a semi-ring as in [1] could be added, but it is irrelevant as long as we do not speak about operations on $\mathbb{K}$.

Definition 3.2 [Weighted decision diagram (WDD)] Let $\mathcal{Z}, \mathbb{K}, \Phi, \Phi^{const}$ be as above and π a variable ordering for $\mathcal{Z}$. A π-WDD for $(\mathcal{Z}, \mathbb{K}, \Phi, \Phi^{const})$ is a rooted, binary branching, acyclic graph $\mathcal{B}$ with several additional information. For the inner nodes, we have

- a function var that assigns a variable $var(v) \in \mathcal{Z}$ to any inner node v,
- functions $v \mapsto succ_0(v)$ and $v \mapsto succ_1(v)$ that specify the successors of v,
- functions $v \mapsto \phi_0(v)$ and $v \mapsto \phi_1(v)$ that specify the transformations associated with the outgoing edges from v.

For the terminal nodes, we have a function $v \mapsto value(v) \in \mathbb{K}$. If v is an inner node and $\xi \in \{0,1\}$ such that $succ_\xi(v)$ is an inner node then we require that $var(v)$ occurs in π before $var(succ_\xi(v))$ and $\phi_\xi(v) \in \Phi$. If $succ_\xi(v)$ is a terminal node then we require $\phi_\xi(v) \in \Phi^{const}$.
The root of $\mathcal{B}$ is a pair $r = \langle \phi_r, v_r \rangle$ consisting of a function $\phi_r \in \Phi \cup \Phi^{const}$ and a node v_r from which all other nodes in $\mathcal{B}$ are reachable. As for the edges we require $\phi_r \in \Phi$ if v_r is an inner node and $\phi_r \in \Phi^{const}$ if v_r is terminal [4]. $\square$

The semantics of a WDD $\mathcal{B}$ is formalized by associating a function $f_v \in \mathbb{F} = \mathbb{F}^{const} \cup \mathbb{F}^{non\text{-}const}$ to any node in $\mathcal{B}$ and a function for the root r. Intuitively, an incoming edge of node v labelled with φ stands for the function $\varphi \circ f_v$. Formally, the function $f_\mathcal{B}$ for WDD $\mathcal{B}$ is those induced by its root $r = \langle \phi_r, v_r \rangle$ which is given by $f_\mathcal{B} = \phi_r \circ f_{v_r}$ where the function f_v for the nodes is defined in a bottom-up fashion. The terminal nodes represent constant functions as expected, i.e., $f_v = value(v)$ for any terminal node v. If v is a z-node then $f_v : Eval(\mathcal{Z}) \to \mathbb{K}$ is defined as follows. Let $\eta \in Eval(\mathcal{Z})$ and $\eta(z) = \xi \in \{0,1\}$ then $f_v(\eta) = \phi_\xi(v) \circ f_{succ_\xi(v)}(\eta)$. That is, if $\mathbb{K} \subseteq \mathbb{R}$ then we may write f_v as

$$f_v \;=\; (1-z)\cdot(\phi_0(v) \circ f_{succ_0(v)}) \;+\; z\cdot(\phi_1(v) \circ f_{succ_1(v)}).$$

Before discussing the canonicity for WDDs, we observe that the notion of WDDs covers several types of known BDD-variants. For $\mathbb{K} = \{0,1\}$ our notion of a WDD specializes to an ordinary ordered BDD [5] when dealing with $\Phi = \{id\}$ and to ordered BDDs with complement bits for the edges [22] when dealing with $\Phi = \{id, \neg\}$. For $\mathbb{K} = \mathbb{R}$ (or $\mathbb{K} = \mathbb{N}$ or any other semi-ring) MTBDDs [1,8,11] are obtained through $\Phi = \{id\}$, while edge-valued BDDs [27] arise by taking $\Phi = \{x \mapsto x + b : b \in \mathbb{K}\}$. In these examples, the incoming edges of the terminal nodes do not play a special role and we may deal with $\Phi^{const} = \Phi$ in either case. Factored edge-valued BDDs (FEVBDDs) are obtained by taking $\Phi = \left\{x \mapsto ax + b : a, b \in \mathbb{K}, a \neq 0\right\}$ [5] and Φ^{const} the set of functions $x \mapsto x + b$ where $b \in \mathbb{K}$. Fig. 1 shows four FEVBDDs where we simply write (a,b) to denote the function $x \mapsto ax + b$ and use dashed lines for the edges to the 0-successors and solid lines for the edges to the 1-successors. The WDD in (1) represents the function $3y + 1$, while the WDDs in

[4] In this case, v_r is the only node in $\mathcal{B}$.
[5] $a \neq 0$ because elements of Φ must be bijections.

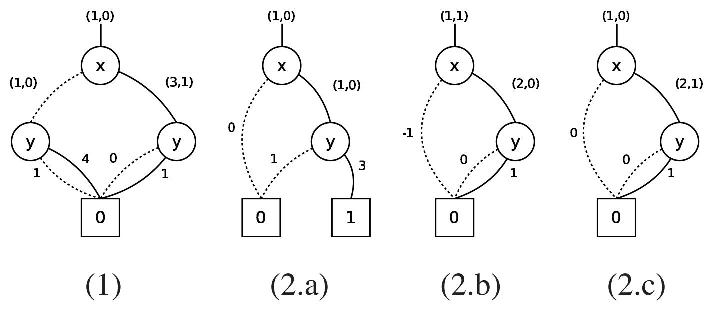


Fig. 1. Example for WDDs with multiplicative and additive edge-weights

(2.a), (2.b) and (2.c) represent the function $f = x(1+2y)$. The following example shows how the function f represented by the WDD in (2.a) can be calculated.

- $f_y = (1-y)\cdot(1+0)+y\cdot(3+1) = 3y+1$
- $f_x = (1-x)\cdot(0+0)+x\cdot(1\cdot f_y+0) = x\cdot(3y+1)$
- $f = 1\cdot f_x + 0 = x\cdot(3y+1)$

For any variable ordering π and any function $f : Eval(\mathcal{Z}) \to \mathbb{K}$ the decision tree for f with respect to ordering π yields a π-WDD for f when we attach the "trivial" weight $\varphi = id$ to all edges and the root node. Thus, π-WDDs provide a *universal* data structure for functions $f \in \mathbb{F}$. We now turn to the question how to formalize freedom of redundancies in WDDs, i.e., to provide a formal notion of reduced WDDs, and how to ensure the uniqueness of the representation of functions by π-WDDs.

Notation 3.3 [The equivalence $\equiv$] Let $\mathbb{F}'$ be $\mathbb{F}^{non\text{-}const}$ or $\mathbb{F}^{const}$ and Φ' the corresponding set of transformations, i.e., $\Phi' = \Phi$ if $\mathbb{F}' = \mathbb{F}^{non\text{-}const}$ and $\Phi' = \Phi^{const}$ if $\mathbb{F}' = \mathbb{F}^{const}$. Let $\equiv_{\Phi'}$ be the following equivalence on $\mathbb{F}'$ such that:

$$\text{If } f,g \in \mathbb{F}' \text{ then } f \equiv_{\Phi'} g \text{ iff there exists } \varphi \in \Phi' \text{ with } f = \varphi \circ g.$$ [6]

We write $\equiv$ for the coarsest equivalence on $\mathbb{F}$ which identifies all non-constant functions $f,g \in \mathbb{F}^{non\text{-}const}$ with $f \equiv_{\Phi} g$ and all constant functions $f,g \in \mathbb{F}^{const}$ with $f \equiv_{\Phi^{const}} g$. Capitol letters $F,G,\ldots$ will be used for the equivalence classes of $\mathbb{F}$ under $\equiv$. $\Box$

Definition 3.4 [Reduced WDDs] A π-WDD $\mathcal{B}$ is called reduced iff for all nodes v, w in $\mathcal{B}$ we have $f_v \equiv f_w$ implies $v = w$. $\Box$

For instance, the two π-WDDs shown in Fig. 1 (2.b) and (2.c) are reduced, while the ones in Fig. 1 (1) and (2.a) are not. In (2.a), the two sinks represent (constant) functions that can be transformed to each other via bijections in Φ^{const}. In (1), the two y-nodes represent (non-constant) functions that can be transformed to each other via bijections in Φ.
Our next goal is to establish criteria that ensure the canonicity of reduced π-WDDs.

[6] Note that $\equiv_{\Phi'}$ is in fact an equivalence as we have $f = \varphi\circ g$ implies $g = \varphi^{-1}\circ f$. Moreover, $\varphi\circ f = \psi\circ g$ implies $f = (\varphi^{-1}\circ\psi)\circ g \equiv_{\Phi'} g$.

We first observe that for the representation of a function by a reduced π-WDD there is still the freedom to choose the representatives in the $\equiv$-equivalence classes. Thus, reduced π-WDDs for the same function need not to be isomorphic. (We use the notion "isomorphism" for π-WDDs $\mathcal{B}$ and $\mathcal{C}$ in the sense that $\mathcal{B}$ and $\mathcal{C}$ agree up to renaming of the nodes.) Instead, as we will see in Theorem 3.6, reduced π-WDDs are *weakly isomorphic* by which we mean that they agree when abstracting away from the names of the nodes and ignoring the weights for the edges and the root. In particular, weakly isomorphic WDDs have the same size (number of nodes).

Lemma 3.5 *Let $f, g \in \mathbb{F}$ and $f \equiv g$. Then, f and g have the same essential variables and for all $z \in \mathcal{Z}$ we have $f|_{z=0} \equiv g|_{z=0}$ and $f|_{z=1} \equiv g|_{z=1}$.*

Proof. Obvious as $f = \varphi \circ g$ implies $f|_{z=\xi} = \varphi \circ g|_{z=\xi} \equiv g|_{z=\xi}$. □

Lemma 3.5 yields that cofactors can be built for $\equiv$-equivalence classes. That is, if $F \subseteq \mathbb{F}$ and $\xi \in \{0,1\}$ then we may write $F|_{z=\xi}$ to denote the unique $\equiv$-equivalence class that contains the functions $f|_{z=\xi}$ for all $f \in F$. A similar notation $F|_{z_1=\xi_1,\ldots,z_k=\xi_k}$ is used if we consider cofactors for several variables.
Let $\pi = (z_1, \ldots, z_n)$ be a variable ordering and $\mathcal{B}$ a reduced π-WDD. If $F_{\mathcal{B}} = [f_{\mathcal{B}}]_{\equiv}$ then there is a 1-1-correspondence between the cofactors $F_{\mathcal{B}}|_{z_1=\xi_1,\ldots,z_k=\xi_k}$ and the nodes in $\mathcal{B}$. (Note that some of these cofactors might agree.) To see why, we may use an inductive argument starting in the root node. If v is an inner node in $\mathcal{B}$ such that $[f_v]_{\equiv} = F_{\mathcal{B}}|_{z_1=\xi_1,\ldots,z_k=\xi_k}$ then v is labelled with a variable z_ℓ where $\ell > k$ and z_ℓ is the first essential variable for f_v (and all functions in $[f_v]_{\equiv}$). Moreover, the function f_{v_0} for the 0-successor v_0 of v is in the equivalence class

$$[f_v]_{\equiv}\big|_{z_\ell=0} = F_{\mathcal{B}}|_{z_1=\xi_1,\ldots,z_k=\xi_k,\ldots,z_\ell=0}$$

for arbitrary assignments of the variables $z_{k+1}, \ldots, z_{\ell-1}$. A similar condition holds for the 1-successor of v. Hence, if we ignore the edge-weights then all reduced π-WDDs for the same function have the same structure. We obtain:

Theorem 3.6 [Weak canonicity of reduced WDDs] *Let π be a variable ordering and $\mathcal{B}$, $\mathcal{C}$ reduced π-WDDs. Then: If $f_{\mathcal{B}} \equiv f_{\mathcal{C}}$ then $\mathcal{B}$ and $\mathcal{C}$ are weakly isomorphic. In particular, $|\mathcal{B}| = |\mathcal{C}|$.*

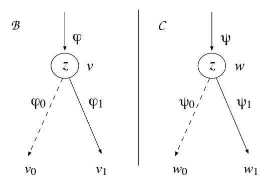
By a selection function for $\mathbb{F}$, we mean a function $\mathcal{S} : \mathbb{F}/\equiv \longrightarrow \mathbb{F}$ which "selects" in any equivalence class $F \in \mathbb{F}/\equiv$ a representative $\mathcal{S}(F) \in \mathbb{F}$. For $f \in \mathbb{F}$, we simply write $\mathcal{S}(f)$ rather than $\mathcal{S}([f]_{\equiv})$. Thus, $f \equiv \mathcal{S}(f)$ for all $f \in \mathbb{F}$.

Definition 3.7 [$\mathcal{S}$-reduced WDDs] A π-WDD $\mathcal{B}$ is called $\mathcal{S}$-reduced if $\mathcal{B}$ is reduced and $f_v = \mathcal{S}(f_v)$ for all nodes v in $\mathcal{B}$. □

Theorem 3.8 [Canonicity of $\mathcal{S}$-reduced WDDs] *Let π be a variable ordering, $\mathcal{S}$ a selection function and let $\mathcal{B}$, $\mathcal{C}$ be $\mathcal{S}$-reduced π-WDDs with $f_{\mathcal{B}} = f_{\mathcal{C}}$. Then, $\mathcal{B}$ and $\mathcal{C}$ are isomorphic.*

Proof. Our argument is by induction on $n = |\mathcal{Z}|$ (number of variables). If $n = 0$ then $f_{\mathcal{B}} = f_{\mathcal{C}}$ is constant. Let c be the value of $f_{\mathcal{B}} = f_{\mathcal{C}}$ and $c' = \mathcal{S}(c)$. Then, the root of $\mathcal{B}$ and $\mathcal{C}$ consists of a terminal node labelled with c' together with the unique transformation $\varphi \in \Phi^{const}$ such that $\varphi(c') = c$.

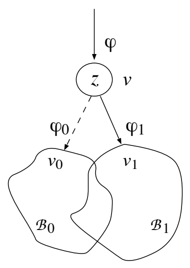
In the induction step, we assume that the root nodes of $\mathcal{B}$ and $\mathcal{C}$ are inner nodes, say z-nodes. That is, z is the first essential variable in $f_{\mathcal{B}} = f_{\mathcal{C}}$ according to the ordering π. Let $r_{\mathcal{B}} = \langle \varphi, v \rangle$ be the root of $\mathcal{B}$ and $r_{\mathcal{C}} = \langle \psi, w \rangle$ the root of $\mathcal{C}$.



Then, $\varphi \circ f_v = f_{\mathcal{B}} = f_{\mathcal{C}} = \psi \circ f_w$. Thus, $f_v \equiv f_w$. As $\mathcal{B}$ and $\mathcal{C}$ are $\mathcal{S}$-reduced we have $f_v = f_w$ and $\varphi = \psi$ (by condition (2) of Φ, cf. Notation 3.1). For $\xi \in \{0,1\}$, let $v_\xi = succ_\xi(v)$, $w_\xi = succ_\xi(w)$, $\phi_\xi(v) = \varphi_\xi$, $\phi_\xi(w) = \psi_\xi$. Then, $\varphi_\xi \circ f_{v_\xi} = f_v|_{z=\xi} = f_w|_{z=\xi} = \psi_\xi \circ f_{w_\xi}$. Hence, $f_{v_\xi} \equiv f_{w_\xi}$. Again, as $\mathcal{B}$ and $\mathcal{C}$ are $\mathcal{S}$-reduced, we get $f_{v_\xi} = f_{w_\xi}$ and $\varphi_\xi = \psi_\xi$ (by the conditions for Φ and Φ^{const}). By induction hypothesis, the sub-WDD with root nodes v_ξ and w_ξ are isomorphic. Hence, $\mathcal{B}$ and $\mathcal{C}$ are isomorphic. □

Theorem 3.9 [Universality and uniqueness of $\mathcal{S}$-reduced WDDs] *Let π be a variable ordering, $\mathcal{S}$ a selection function and $f : Eval(\mathcal{Z}) \to \mathbb{R}$. Then, there exists a unique $\mathcal{S}$-reduced π-WDDs $\mathcal{B}$ with $f = f_{\mathcal{B}}$. (Uniqueness is up to isomorphism.)*

Proof. It remains to provide the proof for the existence of a $\mathcal{S}$-reduced π-WDD for f. The construction is by induction on the number n of essential variables of f. If $n = 0$ then f is constant. Let $c = \mathcal{S}(f)$. There exists a $\varphi \in \Phi^{const}$ with $\varphi(c) = f$. Thus, we may use a WDD consisting of the root $\langle \varphi, v \rangle$ where v is a terminal node v labelled with c. In the induction step, we assume that f is not constant.



Let z be the first essential variable of f according to π and let $g = \mathcal{S}(f)$ and $f = \varphi \circ g$ where $\varphi \in \Phi$. For $\xi \in \{0,1\}$, let $g_\xi = \mathcal{S}(g|_{z=\xi})$ and $g|_{z=\xi} = \varphi_\xi \circ g_\xi$ where $\varphi_\xi \in \Phi$ if $g|_{z=\xi}$ is not constant and $\varphi_\xi \in \Phi^{const}$ if $g|_{z=\xi}$ is constant. By induction hypothesis there exist $\mathcal{S}$-reduced π-WDDs $\mathcal{B}_0$ and $\mathcal{B}_1$ for g_0 and g_1 respectively. We may assume w.l.o.g. that $\mathcal{B}_0$ and $\mathcal{B}_1$ share the same nodes for common cofactors. More precisely, we may assume that if w_0 is a node in $\mathcal{B}_0$ and w_1 a node in $\mathcal{B}_1$ such that $f_{w_0} = f_{w_1}$ then $w_0 = w_1$. (Otherwise the nodes in $\mathcal{B}_1$ can be renamed as there is an isomorphism between the sub-WDDs with root nodes w_0 and w_1, cf. Theorem 3.8.) We then may compose $\mathcal{B}_0$, $\mathcal{B}_1$ to a $\mathcal{S}$-reduced π-WDD for f as shown in the picture on the left. □

Theorem 3.9 applied to $\Phi = \Phi^{const} = \{id\}$ yields the known results that reduced MTBDDs are a canonical data structure for functions $f : Eval(\mathcal{Z}) \to \mathbb{K}$. As we have here $f \equiv g$ iff $f = g$ the selection function is obvious.

For reduced BDDs with negated edges [22], Theorem 3.9 has to be applied with the selection function $\mathcal{S}$ which is defined inductively on the number n of variables. $\mathcal{S}$ chooses 1 as representative for the constant functions. For f to be a boolean function with first essential variable z, $\mathcal{S}(f)$ is the unique function $g \equiv f$ such that $g|_{z=1} = \mathcal{S}(g|_{z=1})$. The latter corresponds to the requirement that only the edges leading to the 0-successors might be negated.

To obtain reduced edge-valued BDDs as in [27], we may deal with the selection function $\mathcal{S}$ that selects 0 for the equivalence class of the constant functions and, for

f to be non-constant with the first essential variable z, $\mathcal{S}(f)$ is the unique function $g \equiv f$ where $g|_{z=0} = \mathcal{S}(g|_{z=0})$. The latter means that only the edges to the 1-successor might be associated with an additive weight.
For factored edge-valued BDDs with the rational reduction rule as in [26], $\mathcal{S}(c) = 0$ for all constants c. If z is the first essential variable of f then $\mathcal{S}(f)$ is the unique function $g \equiv f$ such that

- if $f|_{z=0}$ is not constant then $g|_{z=0} = \mathcal{S}(f|_{z=0})$,
- if $f|_{z=0}$ is constant, but $f|_{z=1}$ is not constant then $\mathcal{S}(f) = z \cdot (h+b)$ where $h = \mathcal{S}(f|_{z=1})$ and $b = \frac{1}{c}(d - f|_{z=0})$ with unique constants c and d such that $f|_{z=1} = ch + d$,
- if $f|_{z=0}$ and $f|_{z=1}$ are constant then $z = \mathcal{S}(f)$.

In either case, the choice of the selection function was made in such a way that it is easy to implement. However, also other selection function could have been used. Although reduced WDDs under different selection functions might be different, Theorem 3.6 yields that they have the same topological structure, and hence, equal size.

Multi-branching weighted decision diagrams (MWDDs). In a similar way, we can treat transformations of the input variables such as input inverters [22]. The idea is to augment the incoming edges of a z-node with pairs (φ, λ) where φ is a transformation for the output values as before and $\lambda : Dom(z) \to Dom(z)$ a permutation of the possible values for variable z. Here, $Dom(z)$ denotes the domain of z. For binary branching decision diagrams where all variables are of boolean-type, $Dom(z) = \{0,1\}$ for all variables z. In that case, id and $\neg$ are the only possible permutations. However, we can generalize our approach to multi-branching decision diagrams (in the style of MDDs [17]) where the domain of any variable z is an arbitrary finite set $Dom(z)$ with at least two elements. That is, we now consider functions of the type $f : Eval(\mathcal{Z}) \to \mathbb{K}$ where $Eval(\mathcal{Z})$ is the set of functions $\eta : \mathcal{Z} \to \cup_{z \in \mathcal{Z}} Dom(z)$ such that $\eta(z) \in Dom(z)$ for all variables z. In addition to Φ, Φ^{const}, we will deal with nonempty sets Λ_z of bijections $\lambda : Dom(z) \to Dom(z)$. For instance, if $Dom(z) = \{0, 1, \ldots, p-1\}$ where $p \geq 2$ then we may choose $\Lambda_z = \{\xi \mapsto (\zeta + \xi) \mod p : \zeta \in Dom(z)\}$.
A multi-branching weighted decision diagram MWDD is like a WDD except that any z-node v has a successor $succ_\xi(v)$ for each of the values $\xi \in Dom(z)$. The incoming edges of v are labelled with pairs (φ, λ) where $\varphi \in \Phi$ and $\lambda \in \Lambda_z$. As before, the incoming edges of the terminal nodes are augmented with a output-transformation $\varphi \in \Phi^{const}$. Instead of $\equiv$ we now deal with the finest equivalence $\sim$ which identifies all constant functions f, g with $f = \varphi \circ g$ for some $\varphi \in \Phi^{const}$ and such that for all non-constant functions f, g with first essential variable z:

$$f \sim g \text{ iff there exists } \varphi \in \Phi \text{ and } \lambda \in \Lambda_z \text{ with } f|_{z=\xi} = \varphi \circ g|_{z=\lambda(\xi)} \ \forall \xi \in Dom(z).$$

Note that $\sim$ depends on Φ, Φ^{const}, the sets Λ_z for $z \in \mathcal{Z}$ and the ordering π.

A reduced π-MWDD means a π-MWDD such that, for all nodes v, w, $f_v \sim f_w$

implies $v = w$. Similarly to the binary branching case, if $\mathcal{B}$ and $\mathcal{C}$ are reduced π-MWDDs with $f_{\mathcal{B}} = f_{\mathcal{C}}$ then $\mathcal{B}$ and $\mathcal{C}$ have the same underlying graph where the edge-relation is viewed as a multiset of node-pairs (in particular, $|\mathcal{B}| = |\mathcal{C}|$), but $\mathcal{B}$ and $\mathcal{C}$ might differ in the edge-attributes. To ensure *canonicity*, we may choose an arbitrary selection function $\mathcal{S}$, i.e., a function $\mathcal{S} : \mathbb{F}/\sim \longrightarrow \mathbb{F}$ where $f \sim \mathcal{S}([f]_\sim)$ for all $f \in \mathbb{F}$. MWDD $\mathcal{B}$ is called $\mathcal{S}$-reduced iff $\mathcal{B}$ is reduced and $f_v = \mathcal{S}([f_v]_\sim)$ for all nodes v. We then have that for any selection function $\mathcal{S}$, any ordering π and any $f \in \mathbb{F}$ there is a unique π-MWDD $\mathcal{B}$ with $f_{\mathcal{B}} = f$. Again, uniqueness is up to isomorphism. The proof for this follows the same lines as Theorem 3.8 and Theorem 3.9. (Note, however, that in contrast to Lemma 3.5, $f \sim g$ and $f|_{z=\xi} \not\sim g|_{z=\xi}$ is possible.)

The concept of *input inverters* for binary decision diagrams [22] turns out to be an instance of MWDDs when we deal with $\mathbb{K} = \{0,1\}$, $\Phi = \Phi^{const} = \{id\}$ and $Dom(z) = \{0,1\}$, $\Lambda_z = \{id, \neg\}$ for all $z \in \mathcal{Z}$. The easiest way to choose a selection function $\mathcal{S}$ is to define a total relation $\leq_\sim$ on the equivalence classes, such that $[g|_{z=0}]_\sim \leq_\sim [g|_{z=1}]_\sim$ for all functions $f \in \mathbb{F}$ with $g = \mathcal{S}(f)$.

4 Normalized algebraic decision diagrams

As a special instance of WDDs we introduce a normalized variant of factored edge-valued BDDs. In the sequel, we assume that $\mathbb{K} = \mathbb{R}$. The set Φ of edge-transformations consists of the functions $x \mapsto ax + b$ where $a, b \in \mathbb{K}$ with $a \neq 0$. For the constant function we use the set Φ^{const} consisting of the functions $x \mapsto x + b$ where $b \in \mathbb{R}$. Normalized algebraic decision diagrams (NADD for short) are $\mathcal{S}_{NADD}$-reduced WDDs where the selection function $\mathcal{S}_{NADD}$ (see below) chooses in any equivalence class $F \in \mathbb{F}/\equiv$ for non-constant functions one function g such that the maximum of g is 1 and the minimum is 0. In the sequel, we will call such a function g to be *normalized*.

Before presenting the formal definition of the selection function $\mathcal{S}_{NADD}$ for non-constant equivalence classes we first show that for all non-constant equivalence classes $F \in \mathbb{F}/\equiv$ there are exactly two normalized functions $g, h \in F$.

Lemma 4.1 *For each non-constant function $f : Eval(\mathcal{Z}) \to \mathbb{R}$ there is exactly one triple $\langle a, b, g \rangle$ where $a, b \in \mathbb{R}$, $a > 0$, $f = ag + b$ and $g : Eval(\mathcal{Z}) \to \mathbb{R}$ is normalized.*

Proof. Let $M = \max_\eta f(\eta)$ and $m = \min_\eta f(\eta)$ where η ranges over all evaluations for $\mathcal{Z}$. As f is non-constant we have $m < M$. If $f = ag + b$ where $a > 0$ and g is normalized then $M = a + b$ and $m = b$ which yields $(a, b) = (M - m, m)$. □

From Lemma 4.1 we may derive that any non-constant f has two normalized decompositions, namely $f = ag + b$ where $a > 0$ and $f = (-a)(1 - g) + (b + a)$. Thus, for any equivalence class $F \in \mathbb{F}/\equiv$ for non-constant functions there are *exactly two normalized* functions g and h in F and we have $h = 1 - g$. The selection function $\mathcal{S}_{NADD}$ for NADDs has to choose one of them.

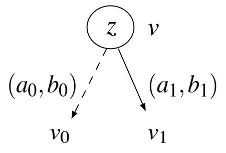
We attempt at a definition of $\mathcal{S}_{NADD}$ which allows for an efficient realization of the creation of new nodes in a NADD-package. This mainly concerns the find-or-add operation where we are given representations $\langle a_0, b_0, w_0\rangle$ and $\langle a_1, b_1, w_1\rangle$ for the cofactors $f|_{z=0} = a_0 f_{w_0} + b_0$ and $f|_{z=1} = a_1 f_{w_1} + b_1$ of a function f (with z the first essential variable of f) and have to insert a root for f, possibly creating a new z-node or reusing an already existing z-node for a normalized function $g \equiv f$.

The definition of $\mathcal{S}_{NADD}(F)$ is by induction on the number of essential variables of $F \in \mathbb{F}/\equiv$. $\mathcal{S}_{NADD}$ chooses 0 as representative for the constant functions. In particular, any NADD contains only the 0-sink, but no other terminal nodes.
Let $F \in \mathbb{F}/\equiv$ be an equivalence class for non-constant functions and z the first essential variable of F. Let $f_0 = \mathcal{S}_{NADD}(F|_{z=0})$ and $f_1 = \mathcal{S}_{NADD}(F|_{z=1})$ and let g and $1-g$ be the two normalized functions in F. Note that $g|_{z=0} \equiv f_0 \equiv 1-g|_{z=0}$ and $g|_{z=1} \equiv f_1 \equiv 1-g|_{z=1}$.

- If f_0 is not constant then we consider the (unique) transformations of f_0 into $g|_{z=0}$ and $1-g|_{z=0}$:
  $$g|_{z=0} = af_0 + b \text{ and } 1-g|_{z=0} = (-a)f_0 + (1-b) \text{ where } a, b \in \mathbb{R},\ a \neq 0.$$
  We then define $\mathcal{S}_{NADD}(F) = g$ if $a > 0$ and $\mathcal{S}_{NADD}(F) = 1-g$ if $a < 0$.
- If f_0 is constant, f_1 is not constant then we consider the (unique) transformations of f_1 into $g|_{z=1}$ and $1-g|_{z=1}$:
  $$g|_{z=1} = af_1 + b \text{ and } 1-g|_{z=1} = (-a)f_1 + (1-b) \text{ where } a, b \in \mathbb{R},\ a \neq 0.$$
  We then define $\mathcal{S}_{NADD}(F) = g$ if $a > 0$ and $\mathcal{S}_{NADD}(F) = 1-g$ if $a < 0$.
- If f_0 and f_1 are constant then we put $\mathcal{S}_{NADD}(F) = z$.[7]

By a π-NADD, we mean a $\mathcal{S}_{NADD}$-reduced π-WDD. The above definition of $\mathcal{S}_{NADD}$ can be reformulated as follows. Let $\mathcal{B}$ be a WDD for $\mathbb{K}, \Phi, \Phi^{const}$ as above, i.e., the edges to the inner nodes of $\mathcal{B}$ are augmented with pairs (a, b) that represent the transformation $x \mapsto ax+b$, while the incoming edges for the terminal nodes have labels of the form $(1, b)$ representing the transformation $x \mapsto x+b$. Then, $\mathcal{B}$ is a NADD iff (i) $\mathcal{B}$ has no terminal node labelled with a value different from 0, and (ii) for any inner node v the function f_v is normalized and if v is as shown in the following picture then we have:

- If v_0 is not the 0-sink then $a_0 > 0$.
- If $v_0 = 0$ and $v_1 \neq 0$ then $a_1 > 0$.
- If $v_0 = v_1 = 0$ then $b_0 = 0 \wedge b_1 = 1$.



From Theorem 3.9 we obtain:

Theorem 4.2 [Universality and uniqueness of NADDs] *Let π be a variable ordering and $f : Eval(\mathcal{Z}) \to \mathbb{R}$ a function. Then, there exists a unique π-NADD $\mathcal{B}$ with $f = f_{\mathcal{B}}$. (Uniqueness is understood up to isomorphism.)*

[7] The case $f_0 = f_1$ is not possible here as we require f to be non-constant and z to be an essential variable of f.

As NADDs and FEVBDDs are both reduced WDDs with the same transformation-sets Φ and Φ^{const}, we conclude from Theorem 3.6 that they always have the same structure and same size, only the weights might be different. In particular, all known results about the size of FEVBDDs in contrast to MTBDDs and EVBDDs carry over to NADDs. First, the size of a π-NADD is bounded above by the size of an equivalent π-MTBDD and (non-factored) π-EVBDDs. This simply follows from the observation that MTBDDs and EVBDDs can be viewed as (possibly non-reduced) WDDs. Second, there are examples where MTBDDs and EVBDDs are exponentially larger than NADDs and FEVBDDs.

Operations on NADDs. As for other BDD-variants, we can adapt the concept of shared BDDs with several roots and use appropriate hash techniques (unique table, computed table) to increase efficiency. We abstract here away from such details and sketch the basic ideas of how to realize operations on NADDs.

The selection function $\mathcal{S}_{NADD}$ can be realized by the find-or-add-operation shown in [23]. Although more complicated than for MTBDDs, "find-or-add" is a local operation and requires a *constant* number of arithmetic operations. Arithmetic operations like summation, multiplication can be realized for NADDs by similar algorithms as for MTBDDs or FEVBDDs, using the schema of Bryant's apply-algorithm [5], which relies on a traversal of the decision graph in a top-down manner. For addition and multiplication, NADDs and FEVBDDs share the same advantage of allowing more terminal cases than MTBDDs. In fact, our experimental studies showed the performance of such operations on NADDs and FEVBDDs is of roughly equal quality.

One major advantage of NADDs over FEVBDDs is the observation that NADDs are more suited for the computation of function minimum and maximum. In fact, extremal function values can be derived in *constant time* from a given NADD-representation [8], while FEVBDDs have to calculate extrema through a graph traversal. Moreover, NADDs simplify the computation of the representation for the minimum or maximum of two functions by allowing more terminal cases than their factored variants. For instance, if $a+b \leq d$ and $a,c > 0$ then $\max\{af+b, cg+d\} = cg+d$ can be performed in constant time.

In a similar way, NADDs support the switch from a real-valued function $f : Eval(\mathcal{Z}) \to \mathbb{R}$ to a boolean function $f' : Eval(\mathcal{Z}) \to \{0,1\}$ via a given threshold, say $f'(\eta) = 1$ if $f(\eta) \geq p$ and $f'(\eta) = 0$ otherwise. Although NADDs and FEVBDDs can realize the transformation $f \rightsquigarrow f'$ by similar algorithms (again, a top-down graph-traversal), early termination in the NADD-approach through the terminal cases "$\max f < p$" or "$\min f \geq p$" can lead to a major speed-up. The switch from a real-valued function f to a boolean function $f' = f \geq p$ is a crucial step for verying probabilistic systems against temporal logical specifications, see e.g. [14,3]. Another often used operation is the comparison of two functions via a threshold.

[8] If $\langle a,b,v\rangle$ is a root for $f = af_v + b$ then $(\min f, \max f) = (b, a+b)$ if $a > 0$ and $(\min f, \max f) = (a+b, b)$ if $a < 0$.

For instance, many iterative algorithms for solving linear equation systems use a termination criterion that checks whether the recently calculated vector f just differs from the prior received vector f' within a given tolerance ε, i.e. whether $|f - f'| < \varepsilon$. The latter is equivalent to the condition $(f < f' + \varepsilon) \wedge (f > f' - \varepsilon)$ which can be evaluated with NADDs or FEVBDDs by a traversal of the sub-graphs for f and f'. Here again, the knowledge of function minima and maxima often allows to skip certain sub-graphs in given NADDs for f and f', while the corresponding subgraphs for FEVBDDs still have to be explored.
Thus, although the size of NADDs and FEVBDDs agree, the former can lead to a speed up in computation time for several operations that allow to formulate terminal cases by means of function extrema.

Implementation and experimental results. For an efficient representation of FEVBDDs just the parameters for the 1-successors have to be stored. At first, it seems that NADDs gained their advantage for calculating the extrema of a function (compared with FEVBDDs) with additional amount of memory per node. As described in [23] and implemented in `http://www.jjs-bdd.de`, for the representation of the attributes (a_0, b_0) and (a_1, b_1) for the outgoing edges of an inner NADD-node, it is only necessary to store two parameters of the four values a_0, b_0, a_1, b_1 and the information how the others can be extracted from them. This follows from the fact that $\min\{b_0, a_0 + b_0, b_1, a_1 + b_1\} = 0$ and $\max\{b_0, a_0 + b_0, b_1, a_1 + b_1\} = 1$. From this point of view, NADDs need just a small amount of memory more than FEVBDDs.
In [23] several benchmarks have been considered to study the efficiency of NADDs. We report here only on two simple examples that illustrate the advantages of NADDs over FEVBDDs and MTBDDs. First, we consider the function $f = x_0 + 2x_1 + \ldots + 2^n x_n$ (the binary encoding of $(n+1)$-bit natural numbers) and the derived boolean function $f' = f < 4$. The results for this benchmark are shown in Table 1. Generating the representation for f by a NADD or FEVBDD requires roughly the same time, while the switch from f to $f < 4$ is much more costly for FEVBDDs than for NADDs (ca. 400-times slower for $n = 24$) and even for MTBDDs. At the first view, the latter is surprising since the MTBDD-representation for f grows exponentially in n. The reduced runtime for MTBDDs can be explained with the times needed to traverse the sub-graph which is much faster for MTBDDs than for FEVBDDs since the factored diagrams have to perform several calculations along the explored paths.
The second example, solving a linear equation system via the iterative Jacobi algorithm, serves to demonstrate that even in applications where function extrema are irrelevant NADDs are no worse than FEVBDDs. Symbolic DD-based representations of vectors and matrices rely on the view of vectors/matrices as switching functions that map (binary encodings for) the indices for the rows and columns to the corresponding entries of the given vector/matrix, see e.g. [8,11,25]. The linear equation system and the times to compute 1000 iterations of the Jacobi method for it can be found in Figure 2.

	MTBDD		FEVBDD		NADD	
n	[s]	$\lvert\mathcal{B}\rvert$	[s]	$\lvert\mathcal{B}\rvert$	[s]	$\lvert\mathcal{B}\rvert$
20	6.01	2097151	0.01	21	0.01	21
21	12.53	4194303	0.01	22	0.01	22
22	26.87	8388607	0.01	23	0.02	23
23	59.08	16777215	0.01	24	0.02	24
24	121.74	33554431	0.01	25	0.02	25

(a)

	MTBDD	FEVBDD	NADD
n	[s]	[s]	[s]
20	0.42	5.36	0.01
21	0.84	10.74	0.01
22	1.70	20.36	0.01
23	3.41	42.31	0.02
24	11.23	83.01	0.02

(b)

Table 1
Building times for f shown in (a) and for $f < 4$ in (b)

$$\begin{pmatrix} -10 & 6 & & & 4 \\ 4 & -10 & 6 & & \\ & \ddots & \ddots & \ddots & \\ & & 4 & -10 & 6 \\ 1 & \dots & \dots & \dots & 1 \end{pmatrix} x = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

n	MTBDD [s]	FEVBDD [s]	NADD [s]
4	1.65	1.42	1.57
8	6.03	4.99	5.74
16	22.29	16.76	21.40
32	91.69	65.17	77.10

Fig. 2. Linear equation system and the times for calculating 1000 iteration steps of the Jacobi method for solving it

As we deal here with a fixed bound for the number of iterations (rather than checking whether the current and previously obtained vector agree up to some tolerance), NADDs cannot profit from other terminal cases than FEVBDDs. Nevertheless the results in Figure 2 show that NADDs are almost as fast as FEVBDDs for solving linear equation systems. For all tested functions the numerical error for FEVBDDs and NADDs has the same dimension.
All results are obtained with the `JJS-BDD` library (`http://www.jjs-bdd.de`).

5 Conclusion

The concept of weighted decision graphs yields a generic framework to reason about the canonicity of various BDD-variants. We applied this framework for a new type of BDDs for the representation of real-valued functions, called normalized algebraic decision diagrams. Although NADDs share the same idea as factored edge-valued BDDs and always lead to a representation of the same size, the performance of several operations is better for NADDs than for the factored variant, such as minimum, maximum, the replacement of a real-valued function with a boolean function. Thus, NADDs are a serious alternative to FEVBDDs or MTBDDs for symbolic reasoning with weighted graphs, linear equations systems, linear optimization problems and other application areas.
In particular, with the above mentioned properties and the fact that the switch from a normalized function f to the function $1 - f$ can be realized with NADDs in constant time , NADDs appear as a promising data structure for symbolic calculations

with probabilities such as model checking for probabilistic automata.

References

[1] R. Bahar, E. Frohm, C. Gaona, G. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic decision diagrams and their applications. *Formal Methods in System Design*, 10(2-3):171–206, 1997.

[2] C. Baier, E. Clarke, V. Hartonas-Garmhausen, M. Kwiatkowska, and M. Ryan. Symbolic model checking for probabilistic processes. In *Proc. International Colloqium on Automata, Languages and Programming (ICALP)*, volume 1256 of *Lecture Notes in Computer Science*, pages 430–440, 1997.

[3] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proc. Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *Lecture Notes in Computer Science*, pages 499–513, 1995.

[4] M. Bozga and O. Maler. On the Representation of Probabilities over Structured Domains. In *Proc. International Conference on Computer Aided Verification (CAV)*, volume 1633 of *Lecture Notes in Computer Science*, pages 261–273, 1999.

[5] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35, 1986.

[6] R. Bryant. Symbolic Boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, 24(3):293–318, 1992.

[7] J. Burch, E. Clarke, K. McMillan, D. Dill, and L. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 1–33. IEEE Computer Society Press, 1990.

[8] E. Clarke, M. Fujita, and X. Zhao. Multi-terminal binary decision diagrams and hybrid decision diagrams. *in [25]*, pages 93–108, 1996.

[9] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[10] R. Drechsler and B. Becker. *Binary Decision Diagrams: Theory and Implementation*. Kluwer Academic Publishers, 1998.

[11] M. Fujita, P. McGeer, and J. Yang. Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. *Formal Methods in System Design*, 10(2-3):149–169, 1997.

[12] G. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Probabilistic Analysis of Large Finite State Machines. In *31st ACM/IEEE Design Automation Conference (DAC)*. San Diego Convention Center, 1994.

[13] G. Hachtel and F. Somenzi. *Logic Synthesis and Verification Algorithms*. Kluwer Academic Publishers, 1996.

[14] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[15] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the Fifteenth Conference on Uncertainty in Articial Intelligence*, pages 279–288, 1999.

[16] Shin ichi Minato. *Binary decision diagrams and applications for VLSI CAD*. Kluwer Academic Publishers, 1996.

[17] T. Kam, T. Villa, R. Brayton, and A. SagiovanniVincentelli. Multi-valued decision diagrams: Theory and applications. *Multiple-Valued Logic: An International Journal*, 4(1-2):9–62, 1998.

[18] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with prism: A hybrid approach. *International Journal on Software Tools for Technology Transfer (STTT)*, 2004.

[19] S. Malik, A. Wang, R. Brayton, and A. Sangiovanni-Vincentelli. Logic verification using binary decision diagrams in logic synthesis environment. In *Proceedings ICCAD*, pages 6–9, 1988.

[20] K. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[21] C. Meinel and T. Theobald. *Algorithms and Data Structures in VLSI Design*. Springer-Verlag, 1998.

[22] S. Minato, N. Ishiura, and S. Yajima. Shared binary decision diagram with attributed edges for efficient boolean function manipulation. In *Proceedings of the 27th ACM/IEEE Design Automation Conference (DAC)*, pages 52–57, 1990.

[23] J. Ossowski. Symbolic representation and manipulation of discrete functions. Master's thesis, University of Bonn, 2004.

[24] H. Sack, C. Meinel, and E. Dubrova. Mod-p decision diagrams: A data structure for multiple-valued functions. In *Proceedings of the 30th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2000)*, page 233. IEEE Computer Society, 2000.

[25] T. Sasao and M. Fujita, editors. *Representation of Discrete Functions*. Kluwer Academic Publishers, 1996.

[26] P. Tafertshofer and M. Pedram. Factored edge-valued binary decision diagrams. *Formal Methods in System Design*, 10(2-3):243–270, 1997.

[27] S. Vrudhula, M. Pedram, and Y. Lai. Edge-valued binary decision diagrams. *in [25]*, pages 109–132, 1996.

[28] I. Wegener. *Branching Programs and Binary Decision Diagrams. Theory and Applications*. Monographs on Discrete Mathematics and Applications, SIAM, 2000.