Available online at www.sciencedirect.com
SCIENCE DIRECT®
Annals of Pure and Applied Logic ▮ (▮▮▮▮) ▮▮▮–▮▮▮
ANNALS OF PURE AND APPLIED LOGIC
www.elsevier.com/locate/apal

Zero, successor and equality in BDDs

Bahareh Badban*, Jaco van de Pol

Department of Software Engineering, Centrum voor Wiskunde en Informatica, P.O. Box 94079, 1090 GB, Amsterdam, The Netherlands

Abstract

We extend BDDs (binary decision diagrams) for plain propositional logic to the fragment of first order logic, consisting of quantifier free logic with zero, successor and equality. We allow equations with zero and successor in the nodes of a BDD, and call such objects $(0, S, =)$-BDDs. We extend the notion of *Ordered* BDDs in the presence of zero, successor and equality. $(0, S, =)$-BDDs can be transformed to equivalent *Ordered* $(0, S, =)$-BDDs by applying a number of rewrite rules until a normal form is reached. All paths in these ordered $(0, S, =)$-BDDs represent satisfiable conjunctions. The major advantage of transforming a formula to an equivalent Ordered $(0, S, =)$-BDD is that on the latter it can be observed in constant time whether the formula is a tautology, a contradiction, or just satisfiable.

1. Introduction

We investigate the satisfiability and tautology problem for Boolean combinations over the equational theory of zero and successor in the natural numbers. The atoms are equations between terms built from variables, zero (0) and successor (S). Formulas are built from atoms by means of negation ($\neg$) and conjunction ($\wedge$). The formulas are quantifier-free, except for the implicit outermost quantifier ($\forall$ when considering tautology checking, and $\exists$ when considering satisfiability). The decision problem for plain equational theories in general is unsolvable already, so we must restrict to particular theories. The decision

* Corresponding author.
E-mail addresses: Bahareh.Badban@cwi.nl (B. Badban), Jaco.van.de.Pol@cwi.nl (J.C. van de Pol).

0168-0072/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.apal.2004.10.005

problem for Boolean combinations over equational theories can be approached in several ways. We shortly review what we will call the DNF-method, the plain BDD-method, the Encoding method and the EQ-BDD method.

In the DNF-method, the formula is transformed to a propositionally equivalent *Disjunctive Normal Form* (DNF). A formula in DNF is satisfiable if and only if at least one of its disjuncts is satisfiable. Such a disjunct is a conjunction of literals (equations and negated equations). For many theories, dedicated decision procedures for deciding satisfiability of conjunctions of (negated) equations exist. Examples include linear and integer programming for arithmetic over integers or reals, congruence closure algorithms to deal with uninterpreted functions (i.e. second order variables), and the Fourier–Motzkin transformation [8] for dealing with linear inequalities. This research was initiated by Shostak [26] and Nelson and Oppen [19]. See also [23,15]. Current research is devoted to combining decision procedures for different theories [25].

The DNF-method has a clear bottleneck, because the transformation to disjunctive normal form is not feasible: the resulting formula may be exponentially bigger than the original. This is improved by the plain BDD-method. In that method, a formula is transformed to a propositionally equivalent *Ordered Binary Decision Diagram* (OBDD [11]), which is a binary directed acyclic graph. Each node is labeled with an atomic proposition, and has a left and a right descendant. The leaves can be either $\top$ (true) or $\bot$ (false). A BDD can be viewed as an if-then-else (*ITE*) tree with shared subterms, where the tests are atomic propositions. In an ordered BDD, the order of the tests in each path of this tree is fixed by a total order on atoms. Although in principle OBDD representations are also exponentially big, it appears that in practice many formulas have a succinct OBDD-representation.

Two propositionally equivalent OBDDs are identical. This means that if all atoms are propositional symbols then OBDDs are unique representations of Boolean functions. However, in our case the atoms are equations, and uniqueness is lost. For instance, the OBDDs $ITE(x = y, \top, \bot)$ and $ITE(y = x, \top, \bot)$ are equivalent, although not propositionally equivalent. Similarly, in the propositional case all paths of an OBDD represent a satisfiable conjunction, but in the equational case this property is lost. For instance, the path to $\bot$ in $ITE(x = y, ITE(y = x, \top, \bot), \top)$ represents the inconsistent conjunction $x = y \wedge y \neq x$. As a result, OBDDs with $\bot$-leaves can still be a tautology.

In order to solve the satisfiability or tautology problem using OBDDs, it must be checked for each path in the OBDD whether it represents a consistent conjunction with respect to the underlying equational theory. This is done by applying the aforementioned decision procedures. If all consistent paths lead to $\top$-leaves, then the BDD is a tautology. If all consistent paths lead to $\bot$-leaves, then the BDD is a contradiction. Otherwise, it is just satisfiable. This procedure is both sound and complete, but due to sharing subterms, an OBDD can have exponentially many paths, so there is still a computational bottleneck. A typical example of this approach is the DDDs (difference decision diagrams) of [18], where all atoms are of the form $x < y + c$, for variables x and y and a constant c (known as separation predicates [22], or difference logic).

In both the DNF- and the plain BDD-method, the Boolean structure is flattened out immediately, and the arithmetic part is dealt with in a second step. In the *Encoding method* these steps are reversed. First the formula is transformed to a purely propositional

formula, which is satisfiable, if and only if the original formula is satisfiable in the equational theory. In this translation, facts from the equational theory (e.g. congruence of functions, transitivity of equality and orderings) are encoded into the formula. Then a *finite model property* is used to obtain a finite upperbound on the cardinality of the model. Finally, variables that range over a set of size n are encoded by $\log(n)$ propositional variables. The resulting formula can be checked for satisfiability with any existing SAT-technique, for instance based on resolution or on propositional BDDs. An early example is Ackermann's reduction [1], by which second order variables can be eliminated. More optimal versions can be found in [16,21,12]. Recently, this method is applied in [24] to Boolean combinations over successor, predecessor, equality and inequality over the integers, in [28] it is applied to separation predicates $x < y + c$, and in [27], Pressburger arithmetic for integers, and linear arithmetic for reals are translated into propositional logic.

In the last approach that we mention, called the EQ-BDD-method (*Binary Decision Diagrams extended with Equality* [17]), Boolean and arithmetic reasoning are not separated, but intertwined. Similar to the plain BDD-approach, an ordered EQ-BDD is constructed, but during this construction, facts from the equational theory are used to prune inconsistent paths at an earlier stage. The main technique is a substitution rule, which allows us to replace $ITE(s = t, \varphi(s), \psi)$ by $ITE(s = t, \varphi(t), \psi)$. It was shown that the resulting normal forms always exist, and have the desirable property that all paths in it represent consistent conjunctions. As a consequence, $\top$ and $\bot$ have a unique EQ-OBDD representation, so tautology, contradiction and satisfiability checking on EQ-OBDDs can be done in constant time. The resulting EQ-OBDDs are logically equivalent to the original formula (not just equi-satisfiable, as in the translations to propositional logic), so this technique can also be used to simplify a given formula. Finally, this technique does not depend on the finite model property. In [17] only the case of equational logic without any function symbols is covered.

Contribution and overview. In [17] BDDs have been extended with equality, resulting in EQ-BDDs. We follow this line of research and extend BDDs to propositional logic with zero, successor and equality, resulting in $(0, S, =)$-BDDs. Our goal is to find a terminating set of rewrite rules on $(0, S, =)$-BDDs, such that all paths in the normal forms represent satisfiable conjunctions. These normal forms are called Ordered $(0, S, =)$-BDDs. As a result, tautology checking and satisfiability checking on Ordered $(0, S, =)$-BDDs can be done in constant time.

In Section 2, we first shortly introduce binary decision diagrams, and then give a formal syntax and semantics of $(0, S, =)$-BDDs. In Section 3 a solution is presented, leading to the set of $(0, S, =)$-OBDDs (ordered BDDs). First any total and well-founded order on variables is extended to a total and well-founded order on atomic guards. Then the rewrite system is presented. Finally, we prove termination and satisfiability over all paths. Section 4 describes some alternative approaches, most of which are failed attempts. These are included in order to provide some insight in the subtleties of the method. Finally, Section 5 concludes with some remarks on implementation and possible applications.

2. Binary decision diagrams with equality

2.1. Binary decision diagrams

A *Binary Decision Diagram* [11] (BDD) represents a Boolean function as a finite, rooted, binary, ordered, directed acyclic graph. The leaves of this graph are labeled $\bot$ and $\top$, and all internal nodes are labeled with Boolean variables. A node with label p, left child L and right child R represents the formula *if p then L else R*.

Given a fixed total order on the propositional variables, a BDD can be transformed to an *Ordered* binary decision diagram (OBDD), in which the propositions along all paths occur in increasing order, redundant tests ($ITE(p, x, x)$) do not occur, and the graph is maximally shared. For a fixed order, each Boolean function is represented by a unique OBDD. Furthermore, Boolean operations, such as negation and conjunction, can be computed on OBDDs very cheaply. Together with the fact that (due to sharing) many practical Boolean functions have a small OBDD representation, OBDDs are very popular in verification of hardware design, and play a major role in symbolic model checking.

As it is described in [17], Ordered EQ-BDDs are not necessarily unique, so for our extension, we will not make any attempt to obtain unique representations.

2.2. Adding zero, successor and equality

In this section, we provide the syntax and semantics of BDDs extended with zero, successor and equality. For our purpose, the sharing information present in the graph is immaterial, so we formalize $(0, S, =)$-BDDs by terms (i.e. trees). We view $(0, S, =)$-BDDs as a restricted subset of formulas, and show that every formula is representable as BDD.

Assume V is a set of variables, and define $\bar{V} = V \cup \{0\}$. We define sets of terms, formulas, guards and BDDs as follows.

Definition 1. The sets of terms (W), formulas (Φ), guards (G) and $(0, S, =)$-BDDs (B) are defined as below:

$$W ::= 0 \mid V \mid S(W)$$
$$\Phi ::= \bot \mid \top \mid W = W \mid \neg\Phi \mid \Phi \wedge \Phi \mid ITE(\Phi, \Phi, \Phi)$$
$$G ::= \bot \mid \top \mid W = W$$
$$B ::= \bot \mid \top \mid ITE(G, B, B).$$

We now introduce some notational conventions. Throughout this paper $\equiv$ is used to denote syntactic equality between terms or formulas, in order to avoid confusion with the $=$-symbol in guards. Symbols $x, y, z, u, \ldots$ denote variables; $r, s, t, \ldots$ will range over W; $\varphi, \psi, \ldots$ range over Φ; f, g over guards. Furthermore, we will write $x \neq y$ instead of $\neg(x = y)$ and $S^m(t)$ for the m-fold application of S to t, so $S^0(t) \equiv t$ and $S^{m+1}(t) \equiv S(S^m(t))$. Note that each $t \in W$ is of the form $S^m(u)$, for some $m \in \mathbb{N}$ and $u \in \bar{V}$.

We will use a fixed interpretation of the above formulas throughout this paper. Terms are interpreted over the natural numbers ($\mathbb{N}$) and for formulas we use the classical interpretation over $\{0, 1\}$. In particular, *ITE* denotes the If-Then-Else function. Given a valuation $v : V \to \mathbb{N}$, we extend v homomorphically to terms and formulas in the

following way:

$$\begin{aligned} v(0) &= 0 \\ v(S(t)) &= 1 + v(t) \\ v(\bot) &= 0 \\ v(\top) &= 1 \\ v(s = t) &= 1, \text{ if } v(s) = v(t), 0, \text{ otherwise.} \\ v(\neg\varphi) &= 1 - v(\varphi) \\ v(\varphi \wedge \psi) &= \min(v(\varphi), v(\psi)) \\ v(ITE(\varphi, \psi, \chi)) &= v(\psi), \text{ if } v(\varphi) = 1, v(\chi), \text{ otherwise.} \end{aligned}$$

Given a formula φ, we say it is *satisfiable* if there exists a valuation $v : V \to \mathbb{N}$, such that $v(\varphi) = 1$; it is a *contradiction* otherwise. If for all $v : V \to \mathbb{N}$, $v(\varphi) = 1$, then φ is a *tautology*. Finally, if $v(\varphi) = v(\psi)$ for all valuations $v : V \to \mathbb{N}$, then φ and ψ are called *equivalent*.

Lemma 2. *Every formula defined above is equivalent to at least one* $(0, S, =)$*-BDD.*

Proof. First, we can eliminate all *ITE* symbols by using the equivalence $ITE(\varphi, \psi, \chi) \Leftrightarrow \neg(\neg(\varphi \wedge \psi) \wedge \neg(\neg\varphi \wedge \chi))$. We prove the lemma by induction over the remaining formulas. $ITE(g, \top, \bot)$ is a suitable representation of a formula g when it is a guard. Now suppose φ_1, φ_2 are two given formulas with representations T_1, T_2, respectively. Construct a first $(0, S, =)$-BDD from T_1 by substituting T_2 for its $\top$ symbols and call it T. Construct a second $(0, S, =)$-BDD from T_1 by swapping $\top$ and $\bot$ in T_1 and name it T'. Now T and T' represent $\varphi_1 \wedge \varphi_2$ and $\neg\varphi_1$, respectively. □

3. Ordered (0, S, =)-BDDs

We now introduce a total ordering on guards. It will be used in the definition of *Ordered* Binary Decision Diagrams ($(0, S, =)$-OBDDs). Next we prove that all $(0, S, =)$-BDDs (and hence all formulas) can be transformed to $(0, S, =)$-OBDDs by rewriting. Finally, we show that all paths in $(0, S, =)$-OBDDs represent consistent conjunctions, which make them well suited for deciding satisfiability and contradiction of propositional formulas over zero, successor and equality.

3.1. Definition of $(0, S, =)$-OBDDs

From now on, we use the term "BDD" as an abbreviation for "$(0, S, =)$-BDD". In this section, we define the set of *Ordered* BDDs. To this end an ordering on guards is needed. The latter is parameterized by a total ordering on the variables. In what follows, we consider a fixed total and well-founded order on V (for instance $x \prec y \prec z$).

Definition 3 (*Ordering Definition*). We extend $\prec$ to an order on W:

- $0 \prec u$ for each element u of V.
- $S^m(x) \prec S^n(y)$ iff $x \prec y$ or $(x \equiv y$ and $m < n)$ for each two elements $x, y \in \bar{V}$.

We use term rewriting systems (TRSs), being collections of rewrite rules, in order to specify reductions on guards and BDDs. The reduction relation induced by such a system is the closure of the rules under substitution and context. See [3] for a formal definition. A *normal form* is a term to which no rule applies. A TRS is *terminating* if all its reduction sequences are finite.

The first step to make a BDD ordered, is to simplify all its guards in isolation. Simplification on guards is defined by the following rules:

Definition 4. Suppose g is a guard. By $g\downarrow$ we mean the normal form of g w.r.t. the following rewrite rules:

$$\begin{array}{rcll} x = x & \to & \top & \\ S(y) = S(x) & \to & y = x & \\ S(x) = 0 & \to & \bot & \\ S^{m+1}(x) = x & \to & \bot & \text{for all } m \in \mathbb{N} \\ r = t & \to & t = r & \text{for all } r, t \in W \text{ such that } r \prec t. \end{array}$$

We call g simplified if it cannot be further simplified, i.e., $g \equiv g\downarrow$. A $(0, S, =)$-BDD T is called simplified if all guards in it are simplified. An immediate consequence of the last definition is the following:

Corollary 5. *Each simplified guard has exactly one of the following forms:*

- $x = S^m(0)$ *for some* $x \in V$.
- $y = S^m(x)$ *for some* $x, y \in V$, $x \prec y$.
- $S^m(y) = x$ *for some* $x, y \in V$, $x \prec y$, $m > 0$.
- $\top$, $\bot$.

Now consider the following formula: $\varphi := (S^3(y) = x) \wedge (y = S(x))$. Here we want that $\bot$ becomes the only OBDD which represents φ. So the question is how we can obtain $\bot$ from a BDD representation of φ in a systematic way. The first answer which comes to mind might be the substitution of x in $y = S(x)$ by $S^3(y)$, or y in $S^3(y) = x$ by $S(x)$. But none of these two solutions are satisfactory: both substitutions will yield bigger terms generally, while for our termination arguments we need smaller terms.

Here we solve it by a lifting process which raises the second equation by $S^3(.)$ to obtain $S^3(y) = S^4(x)$, then substitute $S^3(y)$ by x which is the right-hand-side part of the first equality, So it converts to $x = S^4(x)$ which can be simplified to $\bot$ (by Definition 4). Lifting and substitution are defined below, and we will show later that, in combination with simplification, these operations result in smaller guards.

Definition 6. Let $m \in \mathbb{N}$, terms $r, t \in W$, a variable $y \in V$ and a guard $g \in G$ be given. Then we define:

$$\begin{array}{rcl} (r = t)\uparrow^m & := & S^m(r) = S^m(t) \\ g|_{S^m(y)=r} & := & (g\uparrow^m [S^m(y) := r])\downarrow. \end{array}$$

As was mentioned before, to impose an ordering on BDDs, first we need a total ordering on guards. Since we are going to deal with simplified guards, we limit our definition to the simplified guards.

Definition 7 (*Order on Simplified Guards*). We define a total order $\prec$ on simplified guards as below.

- $\bot \prec \top \prec g$, for all guards g, different from $\top, \bot$.
- $(S^p(x) = S^q(y)) \prec (S^m(u) = S^n(v))$ iff:

  (i) $x \prec u$ or
  (ii) $x \equiv u \ \wedge y \prec v$ or
  (iii) $x \equiv u,\ y \equiv v,\ p < m$ or
  (iv) $x \equiv u,\ y \equiv v,\ p \equiv m,\ q < n$.

According to this definition $S^p(x) = S^q(y) \prec S^m(u) = S^n(v)$ iff $(x, y, p, q) \prec_l (u, v, m, n)$, in which $\prec_l$ is a lexicographic order on quadruples of the total, well-founded orders $(\bar{V}, \prec) \times (\bar{V}, \prec) \times (\mathbb{N}, <) \times (\mathbb{N}, <)$, and therefore it is well-founded and total.

Definition 8. An $(0, S, =)$-OBDD (ordered $(0, S, =)$-BDD) is an $(0, S, =)$-BDD which is simplified and a normal form w.r.t. to the following rewrite rules:

(1) $ITE(\top, T_1, T_2) \to T_1$.
(2) $ITE(\bot, T_1, T_2) \to T_2$.
(3) $ITE(g, T, T) \to T$.
(4) $ITE(g, ITE(g, T_1, T_2), T_3) \to ITE(g, T_1, T_3)$.
(5) $ITE(g, T_1, ITE(g, T_2, T_3)) \to ITE(g, T_1, T_3)$.
(6) $ITE(g_1, ITE(g_2, T_1, T_2), T_3) \to ITE(g_2, ITE(g_1, T_1, T_3), ITE(g_1, T_2, T_3))$
provided $g_1 \succ g_2$.
(7) $ITE(g_1, T_1, ITE(g_2, T_2, T_3)) \to ITE(g_2, ITE(g_1, T_1, T_2), ITE(g_1, T_1, T_3))$
provided $g_1 \succ g_2$.
(8) For any simplified $(0, S, =)$-BDD $C, g \in G, r \in W, y \in V$ and $m \in \mathbb{N}$:
$ITE(S^m(y) = r,\ C[g], T) \to ITE(S^m(y) = r,\ C[\, g|_{S^m(y)=r}], T)$
provided y occurs in g and $S^m(y) = r \prec g$.

Rules 1–7 are the normal rules for simplifying BDDs for plain propositional logic [31], which remove redundant tests, and ensure that guards along paths occur in increasing order. Rule 8 allows us to substitute equals for equals. This is needed to take care of transitivity of equality. Other properties of equality, such as reflexivity, symmetry, and injectivity of successor, are dealt with by the simplification rules. From now on we talk about OBDDs instead of $(0, S, =)$-OBDDs. We show an example of the application of rule 8, and an example of a larger derivation.

Example 9. Let $x \prec y \prec z$ (see Fig. 1)

$$\begin{aligned} & ITE(S^2(y) = x, ITE(z = y, \top, \bot), \bot) \\ \xrightarrow{8} \ & ITE(S^2(y) = x, ITE(\{(S^2(z) = S^2(y))[S^2 y := x]\}\downarrow, \top, \bot), \bot) \\ \equiv \ & ITE(S^2(y) = x, ITE(S^2(z) = x, \top, \bot), \bot). \end{aligned}$$

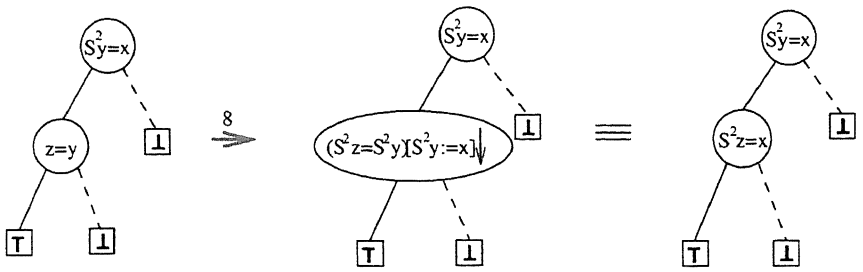

Fig. 1. Example 9.

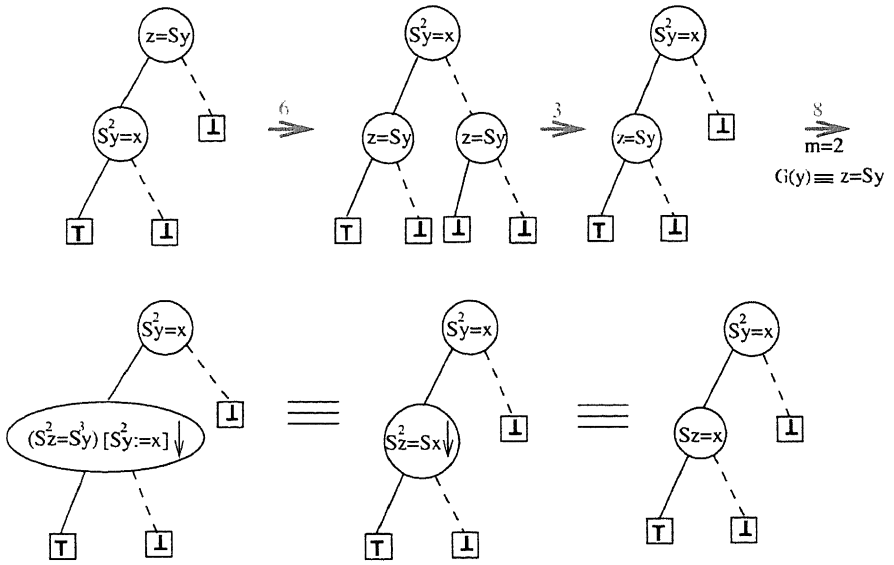

Fig. 2. Derivation of Example 10.

Example 10. Let $x \prec y \prec z$ (See Fig. 2)

$$
\begin{aligned}
& ITE(z = S(y), ITE(S^2(y) = x, \top, \bot), \bot) \\
\xrightarrow{6} \quad & ITE(S^2(y) = x, ITE(z = S(y), \top, \bot), ITE(z = S(y), \bot, \bot)) \\
\xrightarrow{3} \quad & ITE(S^2(y) = x, ITE(z = S(y), \top, \bot), \bot) \\
\xrightarrow{8} \quad & ITE(S^2(y) = x, ITE(\{S^2(z) = S^3(y)[S^2(y) := x]\}\downarrow, \top, \bot), \bot) \\
\overset{\text{substitution}}{\equiv} \quad & ITE(S^2(y) = x, ITE(\{S^2(z) = S(x)\}\downarrow, \top, \bot), \bot) \\
\equiv \quad & ITE(S^2(y) = x, ITE(S(z) = x, \top, \bot), \bot).
\end{aligned}
$$

3.2. Termination

Now we present the first main claim that every BDD with zero, successor and equality has a normal form with respect to the rewrite system of Definition 8, which implies that

each BDD has at least one equivalent OBDD. It suffices to prove termination: we apply TRS rules to a given BDD, until a normal form is reached after a finite number of steps, which is guaranteed by termination. The so-derived BDD is an equivalent OBDD.

We prove termination by means of a powerful tool, the *recursive path ordering* ($\prec_{\text{rpo}}$) [14,30]. This is a standard way to extend a (total) well-founded order on a set of labels to a (total) well-founded order on trees over these labels. To this end, we view guards as labels, ordered by Definition 7, and BDDs are viewed as *binary* trees, so $ITE(g, T_1, T_2)$ corresponds to the tree $g(T_1, T_2)$.

Definition 11 (*Recursive Path Order for BDDs*). $S \equiv f(S_1, S_2) \succ_{\text{rpo}} g(T_1, T_2) \equiv T$ if and only if

(I) $S_1 \succeq_{\text{rpo}} T$ or $S_2 \succeq_{\text{rpo}} T$; or
(II) $f \succ g$ and $S \succ_{\text{rpo}} T_1, T_2$; or
(III) $f \equiv g$ and $S \succ_{\text{rpo}} T_1, T_2$ and either $S_1 \succ_{\text{rpo}} T_1$, or ($S_1 \equiv T_1$ and $S_2 \succ_{\text{rpo}} T_2$).

Here $x \succeq_{\text{rpo}} y$ means that $x \succ_{\text{rpo}} y$ or $x \equiv y$, and $S \succ_{\text{rpo}} T_1, T_2$ is shorthand for $S \succ_{\text{rpo}} T_1$ and $S \succ_{\text{rpo}} T_2$.

This definition yields an order, as is shown in [30]. In order to prove termination, we will show that each rewrite rule (of Definition 8) is indeed a reduction rule regarding $\succ_{\text{rpo}}$. The next lemma will be helpful to show that this reduction property really holds.

Lemma 12. *Let* $f \equiv S^n(y) = S^m(x)$ *and* $g \equiv S^k(w) = S^l(v)$. *If* $f \prec g$ *and* $f \equiv f\!\downarrow$ *and* $g \equiv g\!\downarrow$ *and* $y \in \{v, w\}$ *then* $g|_f \prec g$.

Proof.

- Case I: $y \equiv v$. Therefore $x \prec y \equiv v \prec w$, since f and g are simplified guards. Now

$$\begin{aligned} g|_f &\equiv (g\!\uparrow^n [S^n(y) := S^m(x)])\!\downarrow \\ &\equiv (S^{k+n}(w) = S^{l+m}(x))\!\downarrow && v \equiv y \\ &\prec S^k(w) = S^l(v) && x \prec v, \text{ Definition 7(ii)} \\ &\equiv g. \end{aligned}$$

- Case II: $y \equiv w$. Hence $y \equiv w \succ v$, since g is a simplified guard. Now

$$\begin{aligned} g|_f &\equiv (g\!\uparrow^n [S^n(y) := S^m(x)])\!\downarrow \\ &\equiv (S^{k+m}(x) = S^{l+n}(v))\!\downarrow && w \equiv y. \end{aligned}$$

  And by Definition 7(i), $(S^{k+m}(x) = S^{l+n}(v))\!\downarrow \prec S^k(y) = S^l(v)$, irrespective of whether $x \prec v$ or $v \prec x$, because $x \prec y$ and $v \prec y$. □

Lemma 13. *Let* f, g *be two simplified guards, such that* $f \prec g$, *and* C *is a* $(0, S, =)$*-BDD. If* g *occurs at least once in* C *then* $C[g] \succ_{\text{rpo}} C[f]$.

Proof. Monotonicity of $\succ_{\text{rpo}}$ [30]. □

Lemma 14. *All simplified instances of all rewrite rules are contained in* $\succ_{\text{rpo}}$.

Proof.

(1) $\top(T_1, T_2) \succ_{\text{rpo}} T_1$ by (I)
(2) Similarly
(3) $g(T, T) \succ_{\text{rpo}} T$ by (I)
(4) $g(g(T_1, T_2), T_3) \succ_{\text{rpo}} g(T_1, T_3)$ by (III) and (I)
(5) Similarly
(6) Assume $g_1 \succ g_2$ and let $S \equiv g_1(g_2(T_1, T_2), T_3)$. Then
- $S \succ_{\text{rpo}} T_3$ by (I)
- $g_2(T_1, T_2) \succ_{\text{rpo}} T_1$ by (I)
- $S \succ_{\text{rpo}} T_1$ by (I)

hence $S \succ_{\text{rpo}} g_1(T_1, T_3)$ by (III). Similarly $S \succ_{\text{rpo}} g_1(T_2, T_3)$. And therefore $S \succ_{\text{rpo}} g_2(g_1(T_1, T_3), g_1(T_2, T_3))$ by (II)

(7) Similarly
(8) Let $f \equiv S^m(y) = S^n(x)$. Assume y occurs in g and $f \prec g$, and f and g are simplified. We have to show that $f(C[g], T) \succ_{\text{rpo}} f(C[\ g|_f\ ], T)$. Now using Lemma 12 we conclude $g \succ g|_f$, and so $C[g] \succ_{\text{rpo}} C[g|_f]$ by Lemma 13. Now, by using (I) twice and next (III) it is clear that this rule is also contained in $\succ_{\text{rpo}}$. □

Now we are able to prove our first main claim:

Theorem 15. *The rewrite system defined in Definition 8 is terminating on simplified* $(0, S, =)$*-BDDs.*

Proof. We showed in the previous lemma that all rewrite rules are contained in $\succ_{\text{rpo}}$. This implies termination, because $\succ_{\text{rpo}}$ is a reduction order, i.e. well-founded, and closed under substitutions and contexts [30]. □

This theorem says that by repeated applications of the rewrite rules on an arbitrary simplified BDD, after finitely many iterations we will obtain the normal form of it, which is its equivalent ordered form, so

Corollary 16. *Every* $(0, S, =)$*-BDD is equivalent to at least one OBDD.*

3.3. Satisfiability of paths in OBDDs

For a given formula, we can now construct a BDD representation (Lemma 2) and turn it to an OBDD by rewriting (Corollary 16). We now show the second main claim, stating that all paths in an ordered BDD represent satisfiable conjunctions. As a consequence it can be decided whether the formula is a tautology, a contradiction or a consistency.

NOTATION. Let α, β, γ range over finite sequences of guards and negations of guards. We write ε for the empty sequence, and $\alpha.\beta$ for the concatenation of sequences α and β. If the order of a sequence is unimportant, we sometimes view it as a set, and write $g \in \alpha$, or even $\alpha \cup \beta$. The latter denotes the set of all guards or negations of guards that occur somewhere on α or β.

Definition 17. *Literals* are guards or negations of guards. *Paths* are sequences of literals. We define the set of paths of a BDD T:

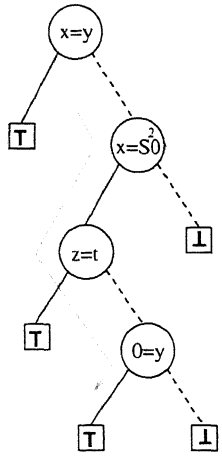

Fig. 3. A path from Example 18.

- $Pat(\top) = Pat(\bot) = \{\varepsilon\}$.
- $Pat(ITE(g, T_1, T_2)) = \{g.\alpha \mid \alpha \in Pat(T_1)\} \cup \{\neg g.\beta \mid \beta \in Pat(T_2)\}$.

A path α is *ordered* if it is a path in some OBDD. Valuation $v : V \to \mathbb{N}$ *satisfies* α if $v(g) = 1$ for all literals $g \in \alpha$. α is *satisfiable* if a valuation v that satisfies it exists.

Example 18. Let

$$T \equiv ITE(x = y, \top, ITE(x = S^2(0), ITE(z = t, \top, ITE(0 = y, \top, \bot)), \bot))$$

then $x \neq y.\ x = S^2(0).\ z \neq t.\ 0 = y$ is a path (Fig. 3). Furthermore, let $x \prec y \prec z$, then $y = x.\ z = x$ is an ordered path, because it is a path in $ITE(y = x, ITE(z = x, \top, \bot), \bot)$, which is an OBDD.

The following two lemmas give some syntactical properties on OBDDs, which can be used for proving satisfiability of each path in an OBDD.

Lemma 19. *Let α be an ordered path, of the form $\beta.(S^p(u) = S^q(y)).\gamma$ Then:*

(i) *u does not occur in γ.*
(ii) *u does not occur at the right-hand side of any literal in β.*
(iii) *u does not occur in a positive guard in β.*
(iv) *y does not occur at the left-hand side of any literal in γ.*

Proof.

(i) Since α is ordered, the rewrite rules should not be applicable. If u occurs in $g \in \gamma$, then either $S^p(u) = S^q(y) \prec g$, and hence rule 8 will be applicable, or $S^p(u) = S^q(y) \succeq g$, and one of rules 4–7 will be applicable.
(ii) Because otherwise, if $g \equiv S^k(v) = S^l(u)$ occurs in β, then $v \succ u$, so $g \succ S^p(u) = S^q(y)$, which will contradict the orderedness of α.

(iii) Regarding part (ii) above, u can possibly occur only in the left-hand side of a positive guard like $S^i(u) = S^j(z)$ in β. Therefore two paths β' and γ' will exist, such that $\alpha \equiv \beta'.(S^i(u) = S^j(z)).\gamma'$, and $S^p(u) = S^q(y)$ will belong to γ', but referring to part (i), this will never happen.

(iv) Similar to part (ii). □

Lemma 20. *Suppose that $S^l(u) = S^k(y)$ and $S^p(u) \neq S^q(y)$ are two literals on an ordered path δ. If v is a valuation on the path which satisfies $S^l(u) = S^k(y)$, then it will also satisfy $S^p(u) \neq S^q(y)$.*

Proof.

- If $S^l(u) = S^k(y) \prec S^p(u) = S^q(y)$, then, two paths β and γ will exist such that $\delta \equiv \beta.(S^l(u) = S^k(y)).\gamma$ in which $S^p(u) \neq S^q(y)$ belongs to γ, but according to Lemma 19(i), this will never happen.
- If $S^p(u) = S^q(y) \prec S^l(u) = S^k(y)$, then since δ is ordered, we can limit our inquiry to the two following cases:
  - $p < l$, and so $k = 0$:

$$\begin{aligned} v(S^p(u)) &= p + v(u) \\ &< l + v(u) \\ &= k + v(y) && v \text{ satisfies } S^l(u) = S^k(y) \\ &= v(y) && k = 0 \\ &\leq q + v(y) \\ &= v(S^q(y)). \end{aligned}$$

  - $p = l$ and $q < k$:

$$\begin{aligned} v(S^p(u)) &= p + v(u) \\ &= l + v(u) && p = l \\ &= k + v(y) && v \text{ satisfies } S^l(u) = S^k(y) \\ &> q + v(y) && q < k \\ &= v(S^q(y)). \end{aligned}$$

In both of these two cases $v(S^p(u)) \neq v(S^q(y))$. □

Definition 21. Suppose $s = t$ is a guard and α is a path. Define:

$$\begin{aligned} &\text{Reverse}(s = t) := t = s \\ &\bar{\alpha} := \alpha \cup \{ \text{Reverse}(g) \mid g \in \alpha \} \cup \{ \neg\text{Reverse}(g) \mid \neg g \in \alpha \}. \end{aligned}$$

Definition 22. Suppose α is an ordered path of the form $\beta.(S^m(z) = S^n(x)).\gamma$. We define a set $E_{x\alpha}$ as follows:

$$E_{x\alpha} = \{u \in \bar{V} \mid S^p(u) = S^q(x) \in \alpha \text{ for some } p, q \in \mathbb{N}\}.$$

Remark 23. According to Definition 22, 0 does not belong to $E_{x\alpha}$, because $S^p(u) = S^q(x)$ is a simplified guard on the ordered path α, therefore $u \succ x$, but we know that 0 does not have this property.

Intuitively, the set $E_{x\alpha}$ contains all variables from terms that are forced to be equal to x by path α. So if we want to raise the value of x, we must raise all values in $E_{x\alpha}$ as well. Note that the value of 0 can not be raised, and raising the value of x could inadvertently make some negated guards in α true. These considerations are captured by the following lemma, which shows how a given valuation of a path can be lifted to arbitrarily high values.

Lemma 24. *Let α be an ordered path of the form $\beta.(S^m(z) = S^n(x)).\gamma$, in which $x \in V$ (i.e. $x \not\equiv 0$). Let v be a valuation that satisfies this path. Then for each $k \in \mathbb{N}$ there exist $l > k$ and a valuation v', such that*

(i) v' *satisfies* α.

(ii) $v'(u) = v(u) + l$ *for each* $u \in E_{x\alpha} \cup \{x\}$.

(iii) $v'(y) = v(y)$ *for each* $y \notin E_{x\alpha} \cup \{x\}$.

Proof. Let us give some notes, before defining any valuation v'.

Note 1. Supposing $S^p(u) = S^q(y)$ is a positive guard on α and $y \not\equiv x$, then u will not belong to $E_{x\alpha} \cup \{x\}$.

Proof.

- $u \not\equiv x$, because otherwise α will be of the form $\mu.(S^p(x) = S^q(y)).\delta$ for some ordered paths μ and δ, and $S^m(z) = S^n(x) \in \mu \cup \delta$. If it is in μ, this contradicts Lemma 19(iii). If it is in δ, this contradicts Lemma 19(i).
- $u \notin E_{x\alpha}$, because otherwise $S^i(u) = S^j(x) \in \alpha$, for some $i, j \in \mathbb{N}$, and this guard will be different from $S^p(u) = S^q(y)$, since $y \not\equiv x$. Therefore $S^i(u) = S^j(x) \prec S^p(u) = S^q(y)$ or vice versa. In each case of these two, a contradiction will be derived, regarding Lemma 19(i). □

Note 2. y will not belong to $E_{x\alpha}$, if $S^p(u) = S^q(y)$ occurs positively or negatively in α, for some $u \in \bar{V}$ and $p, q \in \mathbb{N}$.

Proof. If $y \in E_{x\alpha}$ then $S^i(y) = S^j(x) \in \alpha$ for some $i, j \in \mathbb{N}$. $y \prec u$, since $S^p(u) = S^q(y)$ is a simplified guard on the ordered path α, therefore $S^i(y) = S^j(x) \prec S^p(u) = S^q(y)$. This means that the ordered path $\alpha \equiv \mu.(S^i(y) = S^j(x)).\delta$ for some μ and δ, in which $S^p(u) = S^q(y)$ or its negation will belong to δ, but this will contradict Lemma 19(i). □

Now define:

$$m' = Max\{q + v(y) \mid y \not\equiv x \text{ and } \exists u \in \{x\} \cup E_{x\alpha},\ \exists j \in \mathbb{N} :\ S^j(u) \not\equiv S^q(y) \in \bar{\alpha}\}.$$

Intuitively, m' is bigger than everything distinct from $E_{x\alpha}$. Using this m', we introduce a new valuation v' as below:

$$v'(u) := \begin{cases} v(u) + m' + k + 1 & \text{if } u \in \{x\} \cup E_{x\alpha} \\ v(u) & \text{otherwise} \end{cases}$$

$x \not\equiv 0$ by the assumption. Moreover, given $u \in E_{x\alpha}$, u will be nonzero by Remark 23. Therefore the given definition for v' is well-defined. Now define $l := m' + k + 1$. Then requirements (ii) and (iii) of the lemma are obviously met. Below we will show that requirement (i) holds, i.e. v' satisfies α. Suppose g is a literal on this path:

- If $g \equiv S^p(u) = S^q(y)$, then either of the two following cases applies:
  - $y \equiv x$. So that, $u \in E_{x\alpha}$, and hence $v'(u) = v(u) + m' + k + 1$. Now:

$$\begin{aligned} v'(S^p(u)) &= p + v'(u) \\ &= p + v(u) + m' + k + 1 && v'(u) = v(u) + m' + k + 1 \\ &= v(S^p(u)) + m' + k + 1 \\ &= v(S^q(y)) + m' + k + 1 && v \text{ satisfies } \alpha \\ &= q + v(x) + m' + k + 1 && y \equiv x \\ &= q + v'(x) && v'(x) = v(x) + m' + k + 1 \\ &= v'(S^q(y)) && y \equiv x. \end{aligned}$$

  - $y \not\equiv x$. Now according to the two given notes, u and y will both belong to the last case of the definition of v'. Therefore:

$$\begin{aligned} v'(S^p(u)) &= p + v'(u) \\ &= p + v(u) && v'(u) = v(u) \\ &= v(S^p(u)) \\ &= v(S^q(y)) && v \text{ satisfies } \alpha \\ &= q + v(y) \\ &= q + v'(y) && v'(y) = v(y) \\ &= v'(S^q(y)). \end{aligned}$$

- If $g \equiv S^p(u) \neq S^q(y)$, then $y \notin E_{x\alpha}$, by Note 2. We distinguish two cases:
  - $u \in E_{x\alpha}$. Therefore:

    If $y \equiv x$, then, since $u \in E_{x\alpha}$, so $S^i(u) = S^j(x) \in \alpha$ for some $i, j \in \mathbb{N}$, and according to the previous case, $v'(S^i(u)) = v'(S^j(x))$. Hence $v'(S^p(u)) \neq v'(S^q(x))$ by Lemma 20.

    If $y \not\equiv x$, then $v'(y) = v(y)$ because y also does not belong to $E_{x\alpha}$. Hence:

$$\begin{aligned} v'(S^p(u)) &= v'(u) + p \\ &= v(u) + m' + k + 1 + p && u \in E_{x\alpha} \\ &= v(S^p(u)) + m' + k + 1 \\ &> m' \\ &\geq v(S^q(y)) && \text{definition of } m' \\ &= v'(S^q(y)) && v'(y) = v(y). \end{aligned}$$

  - $u \notin E_{x\alpha}$. Thus:

    If $u \equiv x$, then $y \not\equiv x$, since g is simplified. So y belongs to the last case of the definition of v', because $y \notin E_{x\alpha}$ either, and hence $v'(y) = v(y)$. Now:

$$\begin{aligned} v'(S^p(u)) &= v'(S^p(x)) && u \equiv x \\ &= v'(x) + p \\ &= v(S^p(x)) + m' + k + 1 \\ &> m' \\ &\geq v(S^q(y)) && u \equiv x, \text{ definition on } m' \\ &= v'(S^q(y)) && v'(y) = v(y). \end{aligned}$$

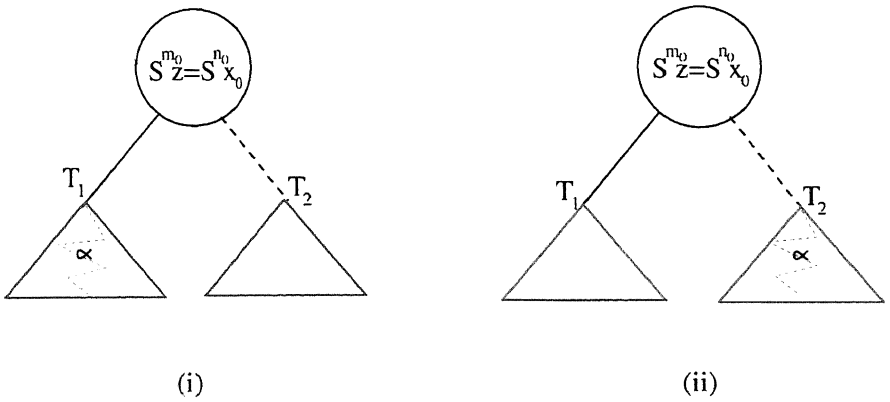

Fig. 4. Theorem 25.

If $u \not\equiv x$, then $v'(u) = v(u)$, since $u \notin E_{x\alpha}$ either. Therefore:

If $y \equiv x$, then

$$\begin{aligned} v'(S^p(u)) &= v'(u) + p \\ &= v(u) + p \\ &\leq m' && y \equiv x, \text{ definition of } m' \\ &< v(x) + m' + k + 1 \\ &= v'(x) \\ &\leq v'(S^q(x)) \\ &= v'(S^q(y)) && y \equiv x. \end{aligned}$$

If $y \not\equiv x$, then y will also belong to the last case of the definition of v', because $y \notin E_{x\alpha}$ either. Thus:

$$\begin{aligned} v'(S^p(u)) &= v'(u) + p \\ &= v(u) + p \\ &= v(S^p(u)) \\ &\neq v(S^q(y)) && v \text{ satisfies } \alpha,\ S^p(u) \neq S^q(y) \in \alpha \\ &= q + v(y) \\ &= q + v'(y) \\ &= v'(S^q(y)). \quad \square \end{aligned}$$

Finally, we come to the second main claim of this paper:

Theorem 25. *Each path in an OBDD is satisfiable.*

Proof. We prove this theorem by induction over OBDDs. Suppose $T \equiv ITE(S^{m_0}(z) = S^{n_0}(x_0), T_1, T_2)$ is an OBDD, and each path belonging to T_1 or T_2, is satisfiable. Then we will show that each path in T is satisfiable as well. Consider α is a satisfiable path, and v is a valuation which satisfies it.

Supposing α belongs to T_1 (Fig. 4(i)), we will provide a new valuation, which will satisfy $(S^{m_0}(z) = S^{n_0}(x_0)).\alpha$. Since T is ordered, z does not occur in any literal of α, by Lemma 19(i).

- If $x_0 \equiv 0$: then $m_0 = 0$, since $S^{m_0}(z) = S^{n_0}(x_0)$ is a simplified guard (Corollary 5). Define:

$$v'(u) := \begin{cases} n_0 & \text{if } u \equiv z \\ v(u) & \text{otherwise.} \end{cases}$$

  v' satisfies $(S^{m_0}(z) = S^{n_0}(x_0)).\alpha$ obviously.
- If $x_0 \not\equiv 0$: z does not occur on α, so that $(z = x_0).\alpha$ is still an ordered path, and without loss of generality, we can define $v(z) := v(x_0)$. Therefore, v will satisfy $(z = x_0).\alpha$. Using Lemma 24, there will be a valuation v' and a natural number $l > m_0$ such that v' satisfies $(z = x_0).\alpha$ and $v'(x_0) = v(x_0) + l$. Now define:

$$v''(u) := \begin{cases} v'(x_0) + n_0 - m_0 & \text{if } u \equiv z \\ v'(u) & \text{otherwise.} \end{cases}$$

  v'' is well-defined since

$$\begin{aligned} v''(z) &= v'(x_0) + n_0 + m_0 \\ &= v(x_0) + l + n_0 + m_0 \\ &= v(x_0) + n_0 + (l + m_0) \\ &\geq 0. \end{aligned}$$

  v'' satisfies α since v' does, moreover

$$\begin{aligned} v''(S^{m_0}(z)) &= m_0 + v''(z) && \text{definition of } v''(z) \\ &= v'(x_0) + n_0 \\ &= v''(x_0) + n_0 \\ &= v''(S^{n_0}(x_0)) \end{aligned}$$

  which means v'' satisfies $S^{m_0}(z) = S^{n_0}(x_0)$. Therefore $(S^{m_0}(z) = S^{n_0}(x_0)).\alpha$ is satisfiable.

Supposing α belongs to T_2 (Fig. 4(ii)), we will provide a new valuation, which will satisfy $(S^{m_0}(z) \neq S^{n_0}(x_0)).\alpha$. Define:

$$\begin{aligned} &H := \bar{\alpha} \cup \{S^{m_0}(z) \neq S^{n_0}(x_0)\} \\ &L_z := \{\, S^i(y) \mid \exists p \in \mathbb{N},\ \exists u \in E_{z\alpha} \cup \{z\}.\ S^p(u) \neq S^i(y) \in H \,\} \\ &k := Max\{\, i + v(y) \mid S^i(y) \in L_z \,\}. \end{aligned}$$

Either of the two following cases will hold:

- z does not occur at the left-hand side of any positive guard of α. If $E_{z\alpha} \neq \emptyset$ then there is a guard $S^p(u) = S^q(z) \in \alpha$ (recall that $S^{m_0}(z) \neq S^{n_0}(x_0)$ is a negative literal).

Applying Lemma 24, on the path $\alpha \equiv \beta.(S^p(u) = S^q(z)).\gamma$, with the defined k above and the supposed valuation v, there is a number $l \in \mathbb{N}$ and a valuation v', such that:

(i) v' satisfies α.
(ii) $v'(u) = v(u) + l$ for each $u \in E_{z\alpha} \cup \{z\}$.
(iii) $v'(y) = v(y)$ for each $y \notin E_{z\alpha} \cup \{z\}$.

Now define

$$l' := \begin{cases} l & \text{if } E_{z\alpha} \neq \emptyset \\ k+1 & \text{otherwise} \end{cases}$$

and

$$v''(y) := \begin{cases} v(y) + l' & \text{if } y \in E_{z\alpha} \cup \{z\} \\ v(y) & \text{otherwise.} \end{cases}$$

Below we will show that v'' satisfies $(S^{m_0}(z) \neq S^{n_0}(x_0)).\alpha$:

. If $E_{z\alpha} = \emptyset$, note that z occurs in negative guards only. Also

$$v''(y) \equiv \begin{cases} v(y) + k + 1 & \text{if } y \equiv z \\ v(y) & \text{otherwise.} \end{cases}$$

v'' satisfies each literal g which does not include z, since $v''(g) = v(g)$. Now we will show that it also satisfies every literal like $S^p(z) \neq S^q(y)$, which occurs on $\bar{\alpha} \cup \{S^{m_0}(z) \neq S^{n_0}(x_0)\}$:

$$\begin{aligned} v'(S^p(z)) &= p + v'(z) \\ &= p + v(z) + k + 1 \\ &> k \\ &\geq v(S^q(y)) && \text{since } S^q(y) \in L_z \\ &= v'(S^q(y)) && v'(y) = v(y). \end{aligned}$$

. If $E_{z\alpha} \neq \emptyset$, then

$$v''(y) \equiv \begin{cases} v(y) + l & \text{if } y \in E_{z\alpha} \cup \{z\} \\ v(y) & \text{otherwise.} \end{cases}$$

Therefore $v''(g) = v'(g)$, for each literal g in α, which means v'' satisfies α. Now for $S^{m_0}(z) \neq S^{n_0}(x_0)$: $x_0 \notin E_{z\alpha} \cup \{z\}$, because $x_0 \not\equiv z$, and also, by Lemma 19(ii), $x_0 \notin E_{z\alpha}$. Hence

$$\begin{aligned} v''(S^{m_0}(z)) &= v(S^{m_0}(z)) + l \\ &> k && (l > k) \\ &\geq v(S^{n_0}(x_0)) && S^{n_0}(x_0) \in L_z \\ &= v''(S^{n_0}(x_0)) && x_0 \notin E_{z\alpha} \cup \{z\}. \end{aligned}$$

- $S^m(z) = S^n(x)$ occurs positively on α, for some $x \in \bar{V}$ and some natural numbers m and n.

. If $x \equiv 0$: then $S^m(z) = S^n(x) \equiv z = S^n(0)$ since $S^m(z) = S^n(x)$ is a simplified guard (Corollary 5). $S^{m_0}(z) = S^{n_0}(x_0) \prec S^m(z) = S^n(x)$, therefore $S^{m_0}(z) = S^{n_0}(x_0) \equiv z = S^{n_0}(0)$ according to the Definition 7. v satisfies $z = S^n(0)$ so it also satisfies $z \neq S^{n_0}(0)$, by Lemma 20, so we are finished.

. If $x \not\equiv 0$:
  - If $x_0 \equiv x$ then, regarding Lemma 20, v will satisfy $S^{m_0}(z) \neq S^{n_0}(x_0)$, because it satisfies $S^m(z) = S^n(x)$.
  - If $x_0 \not\equiv x$: $\alpha \equiv \beta.(S^m(z) = S^n(x).\delta$ for some two ordered paths β and δ. Using Lemma 24, for α and the given number k, above, and the valuation v, there is a valuation v' and a natural number $l > k$, such that v' satisfies α, $v'(u) = v(u) + l$ if $u \in E_{x\alpha} \cup \{x\}$, and $v'(y) = v(y)$ if $y \notin E_{x\alpha} \cup \{x\}$. We will now show that v' is suitable.
  $x_0 \notin E_{x\alpha} \cup \{x\}$, because $x_0 \not\equiv x$, and forall $u \in E_{x\alpha}$, we have $u \succ x$ (by the definition of $E_{x\alpha}$) and $x \succ x_0$ (because $S^{m_0}(z) \neq S^{n_0}(x_0) \prec S^m(z) = S^n(x)$ by Definition 7). Therefore, if x_0 occurs on α, then, $v'(x_0) = v(x_0)$ already, otherwise we can define $v'(x_0) := v(x_0)$ without loss of generality, because v' still satisfies α. We will show that v' satisfies $S^{m_0}(z) \neq S^{n_0}(x_0)$ too:

$$\begin{aligned} v'(S^{m_0}(z)) &= v'(z) + m_0 \\ &= v(z) + l + m_0 && z \in E_{x\alpha} \\ &= v(S^{m_0}(z)) + l \\ &> k && l > k \\ &\geq v(S^{n_0}(x_0)) && S^{n_0}(x_0) \in L_z \\ &= v'(S^{n_0}(x_0)). \quad \square \end{aligned}$$

Corollary 26. *An immediate consequence of Theorem 25 is*

- $\top$ *is the only tautological OBDD.*
- $\bot$ *is the only contradictory OBDD.*
- *Every other OBDD is satisfiable (only).*

Proof. Each path in a tautological OBDD should end in a $\top$, because if T is a tautological OBDD, containing a path α which ends in a $\bot$, then according to Theorem 25, there is a valuation v which satisfies α, but then $v(T) = 0$, which is impossible since T is a tautology. Therefore, if T has more than one leaf, rule 3 of Definition 8 will be applicable on a tautological OBDD which is not $\top$, and this contradicts the orderedness. So $T \equiv \top$. Similarly, for a contradictory one. $\square$

4. Alternative solutions and failed attempts

As shown in Section 3, our main method is to extend a given ordering on variables to terms, and then lexicographically to guards, in such a way that we can prove *termination* (Theorem 15), which guarantees existence of OBDDs as normal forms, and *satisfiability* of paths (Theorem 25), which guarantees that contradictions and tautologies have unique OBDDs. The lexicographic extension of the term-ordering to the guard-ordering, as well as rules 1–8, are familiar from [17]. The creative parts are finding a good ordering on the terms and guards, and the idea of lifting equations. In this section we mention another approach, and two failed attempts, the first of which has non-terminating rewrite sequences, and the second one has multiple contradictory OBDDs.

The variables come with a total order, say $y \succ x$. If we know that $y = x$, then y will be eliminated in the $\top$-branch, by substituting the representant x for it. The solution that we

have described orders the guards by grouping together the variables to be eliminated. In the alternative solution, the representant variables are grouped together. This solution is closer to [17]. More precisely, the order on guards becomes: $S^p(x) = S^q(y) \prec S^m(u) = S^n(v)$ iff $(y, q, x, p)(\prec, <, \prec, <)_{lex}(v, n, u, m)$. For this ordering, the same results can be proved, as we did in a separate technical report [4].

We started our investigations with the ordering of Example 27. It was based on the observation that terms of the form $y = S^n(x)$ are easier to handle than $S^n(y) = x$. In the former case, all y's can be replaced by $S^n(x)$, while in the second case, replacing occurrences of $S^n(y)$ does not remove all occurrences of y (later we solved this by lifting the equation). So we wanted to make terms with S-symbols smaller than terms without S-symbols. Obviously, the resulting ordering on guards is not well-founded. We tried to give an upperbound of the number of S-symbols that occurs in a derivation, but this cannot be done.

Example 27. Consider the following total ordering on variables and their successors:

$$\ldots \prec S^2(x) \prec S^2(y) \prec \ldots \prec S(x) \prec S(y) \prec \ldots \prec x \prec y \prec \ldots$$

and its lexicographic extension to guards: $S^p(x) = S^q(y) \prec S^m(u) = S^n(v)$ iff $(q, y, p, x)(>, \prec, >, \prec)_{lex}(v, n, u, m)$. Then consider the rewrite system of Definition 8, over this new ordering. Now look at the formula below:

$$(y = S^2(x) \wedge z = S(y)) \vee (y \neq S^2(x) \wedge (S^2(z) = y \vee (S^2(z) \neq y \wedge z = S(y)))).$$

In Fig. 5 we show the first steps in a non-terminating rewrite sequence starting from this term. We conjecture that this BDD has no normal form at all.

So, unfortunately, this ordering can not be used, because it leads to non-termination, and the existence of OBDDs cannot be guaranteed. The first repair that comes into mind is reversing this order, so that it becomes well-founded. This led to our second try, in which terms without successors are smaller than terms having S-symbols.

Example 28. Consider an alternative ordering on variables and their successors as below:

$$x \prec y \prec \ldots \prec S(x) \prec S(y) \prec \ldots \prec S^2(x) \prec S^2(y) \prec \ldots \prec S^3(x) \prec \ldots.$$

This order is extended lexicographically on guards: $S^p(x) = S^q(y) \prec S^m(u) = S^n(v)$ iff $(q, y, p, x)(<, \prec, <, \prec)_{lex}(v, n, u, m)$. Next, we take rewrite rules 1–8 of Definition 8 w.r.t. to this new ordering. Now look at this formula:

$$\varphi := S(y) \neq x \wedge S(x) = z \wedge S^2(y) = z.$$

φ is equivalent to $\bot$, but it has an ordered BDD (w.r.t. the new order) as drawn in Fig. 6. This shows that a contradictory OBDD different from $\bot$ exists. The picture shows a path to $\top$, which is *un*satisfiable, so for this ordering, Theorem 25 would not hold.

Apparently, the occurrences of x in $S(y) = x$ and $S(x) = z$ are closely related, and should be treated in the same way. So we decided to change the ordering, so that all terms with x are smaller than all terms with y, etc. This led to the successful definition in Section 3. The price for also allowing terms of the form $S^n(y) = x$ is that, in the

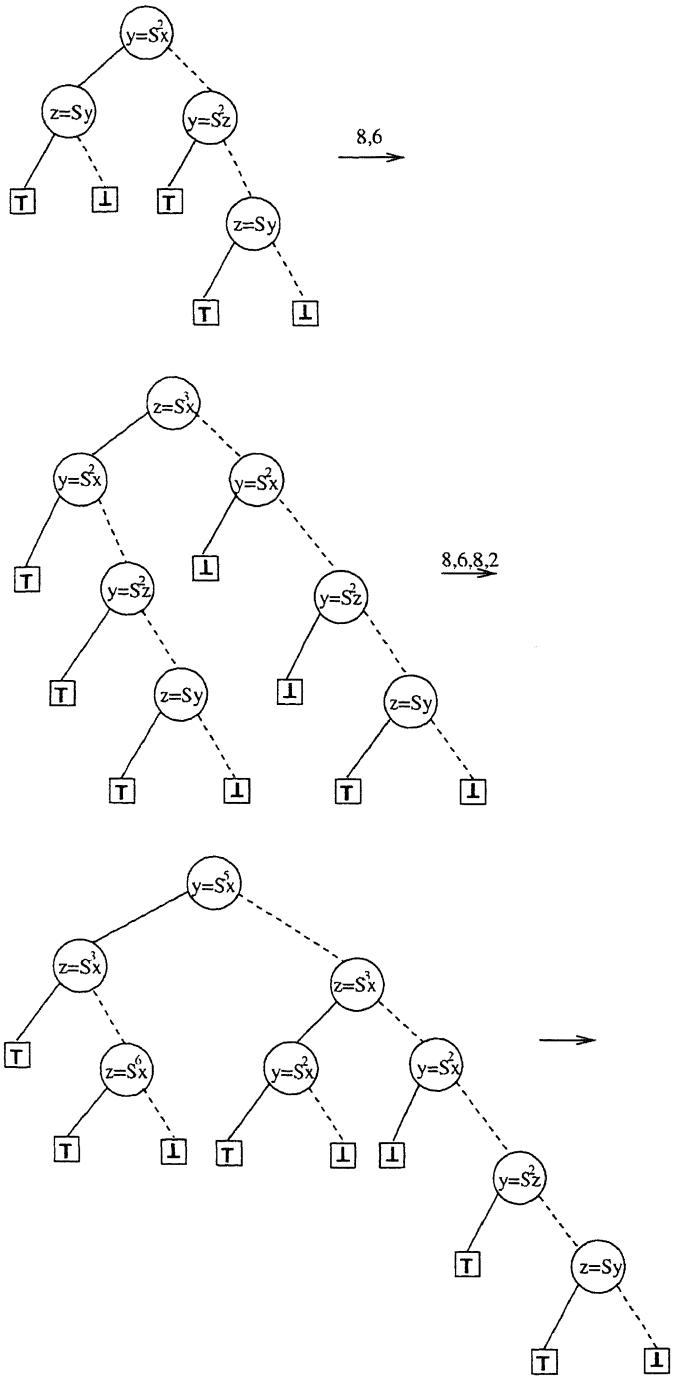

Fig. 5. Example 27.

Other interesting extensions are the incorporation of addition (+), or an investigation of other free algebras (such as LISP-list structures based on null and cons). Another useful extension with only unary constructors would be the binary encoding of the positive natural numbers, based on the free algebra over $(1 : \mathbb{N}, x2p0 : \mathbb{N} \to \mathbb{N}, x2p1 : \mathbb{N} \to \mathbb{N})$. Here $x2p0$ is interpreted as *times 2 plus 0* and $x2p1$ as *times 2 plus 1*. Our results do not immediately apply to such extensions; for each case, we have to define an appropriate order on atoms, a notion of ordered BDDs, and prove that paths in ordered BDDs represent satisfiable conjunctions.

Possible applications. Although the equational fragments that we considered are rather weak (in particular they do not even include addition), many proof obligations in hardware and software verification can be stated in these logics. In [22], Pratt already noticed the relevance of separation formulas of the form $x < y + c$. A similar fragment is also used in real-time model checking as in Uppaal [7,6].

Propositional logic with equality and uninterpreted functions (EUF) has been proposed for verifying the correctness of hardware designs [13]. Also the techniques of [12] are applied to proving equivalence of hardware designs. In [21], similar techniques are applied to the verification of the correctness of compiler optimization results.

These kind of decision procedures are built into many modern interactive theorem provers, such as PVS [20] and SVC [5]. In this interactive context, the ability to simplify a formula, without deciding it completely, may be a convenient feature. The automated theorem prover of the μCRL toolset [9,29] is based on EQ-BDD ideas, and applied in the verification of distributed systems [10].

Acknowledgement

We would like to thank Hans Zantema for his helpful idea to use valuations to prove satisfiability and showing us a preliminary version of his paper [30] on termination of term rewriting.

References

[1] W. Ackermann, Solvable cases of the decision problem, in: Studies in Logic and the Foundations of Mathematics, North-Holland, Amsterdam, 1954.

[2] H.R. Andersen, H. Hulgaard, Boolean expression diagrams, in: Twelfth Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, IEEE Computer Society, 1997, pp. 88–98.

[3] F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, 1998.

[4] B. Badban, J.C. van de Pol, Two solutions to incorporate zero, successor and equality in binary decision diagrams, Technical Report SEN-R0231, Centrum voor Wiskunde en Informatica, Amsterdam, December 2002.

[5] C.W. Barrett, D.L. Dill, J.R. Levitt, Validity checking for combinations of theories with equality, in: M.K. Srivas, A. Camilleri (Eds.), Proceedings of Formal Methods in Computer-Aided Design, FMCAD, LNCS, vol. 1166, Springer, 1996, pp. 187–201.

[6] G. Behrmann, K.G. Larsen, J. Pearson, C. Weise, W. Yi, Efficient timed reachability analysis using clock difference diagrams, in: Proc. of 11th Computer Aided Verification, 1999, pp. 341–353.

[7] J. Bengtsson, K.G. Larsen, F. Larsson, P. Pettersson, W. Yi, UPPAAL: a tool suite for the automatic verification of real-time systems, in: R. Alur, T.A. Henzinger, E.D. Sontag (Eds.), Hybrid Systems III, LNCS, vol. 1066, Springer, 1996, pp. 232–243.

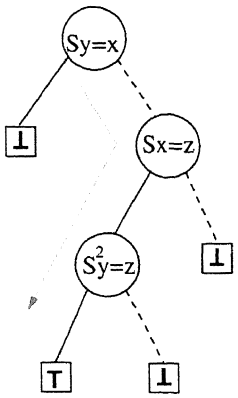

Fig. 6. Unsatisfiable path for Example 28.

substitution, we have to lift all occurrences of y to $S^n(y)$. This slightly complicates the formulation of rewrite rule 8.

5. Conclusion

We developed the theoretical basis for a decision procedure for Boolean combinations of equations with zero and successor. First, a formula is transformed into an $(0, S, =)$-BDD. A rewrite system on $(0, S, =)$-BDDs has been presented, which yields OBDDs (by definition). The system is proved to be terminating, and the normal forms have the desirable property that all paths are satisfiable. As a consequence, if a formula φ is a contradiction (i.e. equivalent to $\bot$), then it reduces to $\bot$. Similarly for tautologies. Therefore, our method can be used to decide tautology and satisfiability. Because the resulting OBDD is logically equivalent to the original formula, our method can also be used to simplify a formula. Although the resulting OBDDs are not unique, our method can also be used to check equivalence of formulas. In order to check whether φ and ψ are equivalent, we can check whether $\varphi \leftrightarrow \psi$ is a tautology.

Towards an implementation. The basic procedure is presented as a term rewrite system. This is still a non-deterministic procedure, because a term can have more than one redex. By proving termination, we established that every strategy will yield an OBDD. However, some strategies might be more effective than others. In [31] rewrite strategies are studied to compute OBDDs for plain propositional logic. In particular, it is shown how the usual efficient OBDD algorithms can be mimicked by a rewrite strategy. Already in [2], various strategies to normalize BEDs (Boolean Expression Diagrams) are described. In [17] a concrete algorithm for EQ-BDDs was presented and proved correct. We have not yet studied particular strategies in the presence of zero and successor, nor implemented the procedure. We view this as important future work. It could be used to evaluate various methods (such as various orderings) by practical examples.

Another line of future research would be the extension of our result to other algebras. An interesting extension would be the incorporation of uninterpreted functions directly (they can already be dealt with by first eliminating them by Ackermann's reduction [1,24]).

[8] A.J.C. Bik, H.A.G. Wijshoff, Implementation of Fourier–Motzkin elimination, Technical Report 94-42, Dept. of Computer Science, Leiden University, Leiden, The Netherlands, 1994.
[9] S.C.C. Blom, W.J. Fokkink, J.F. Groote, I. van Langevelde, B. Lisser, J.C. van de Pol, μCRL: A toolset for analysing algebraic specifications, in: G. Berry, H. Comon, A. Finkel (Eds.), Proc. of CAV 2001, July, LNCS, vol. 2102, Springer, 2001, pp. 250–254.
[10] S.C.C. Blom, J.C. van de Pol, State space reduction by proving confluence, in: E. Brinksma, K.G. Larsen (Eds.), Proc. of Computer-Aided Verification, CAV, LNCS, vol. 2404, Springer, 2002, pp. 596–609.
[11] R.E. Bryant, Symbolic boolean manipulation with ordered binary-decision diagrams, ACM Computing Surveys 24 (3) (1992) 293–318.
[12] R.E. Bryant, S. German, M.N. Velev, Processor verification using efficient reductions of the logic of uninterpreted functions to propositional logic, ACMTCL: ACM Transactions on Computational Logic 2 (2001).
[13] J.R. Burch, D.L. Dill, Automatic verification of pipelined micro-processors control, in: D.L. Dill (Ed.), Proceedings of Computer Aided Verification, CAV'94, LNCS, vol. 818, Springer, 1994, pp. 68–80.
[14] N. Dershowitz, Termination of rewriting, Journal of Symbolic Computation 3 (1–2) (1987) 69–115.
[15] H. Ganzinger, Shostak light, in: A. Voronkov (Ed.), Automated Deduction – CADE-18, LNCS, vol. 2392, Springer, 2002, pp. 332–346.
[16] A. Goel, K. Sajid, H. Zhou, A. Aziz, BDD based procedures for a theory of equality with uninterpreted functions, in: Proc. of Computer Aided Verification, CAV'98, LNCS, vol. 1427, Springer, 1998, pp. 244–255.
[17] J.F. Groote, J.C. van de Pol, Equational binary decision diagrams, in: M. Parigot, A. Voronkov (Eds.), Proc. of LPAR 2000, LNAI, vol. 1955, Springer, 2000, pp. 161–178.
[18] J. Møller, J. Lichtenberg, H.R. Andersen, H. Hulgaard, Difference decision diagrams, in: Computer Science Logic, Denmark, September, 1999.
[19] G. Nelson, D.C. Oppen, Fast decision procedures based on congruence closure, Journal of the ACM 27 (2) (1980) 356–364.
[20] S. Owre, S. Rajan, J.M. Rushby, N. Shankar, M.K. Srivas, PVS: Combining specification, proof checking, and model checking, in: R. Alur, T.A. Henzinger (Eds.), Proceedings of Computer Aided Verification, CAV'96, LNCS, vol. 1102, Springer, 1996, pp. 411–414.
[21] A. Pnueli, Y. Rodeh, O. Shtrichman, M. Siegel, Deciding equality formulas by small domains instantiations, in: N. Halbwachs, D. Peled (Eds.), Proc. of Computer Aided Verification, CAV'99, LNCS, vol. 1633, Springer, 1999.
[22] V. Pratt, Two easy theories whose combination is hard, Technical Report, Massachusetts Institute of Technology, Cambridge, MA, 1970.
[23] H. Ruess, N. Shankar, Deconstructing Shostak, in: 16th Ann. IEEE Symp. on Logic in Computer Science, LICS'01, IEEE, 2001, pp. 19–28.
[24] S.A. Seshia, S. Lahiri, R.E. Bryant, Modeling and verifying systems using a logic of counter arithmetic with lambda expressions and uninterpreted functions, in: E. Brinksma, K.G. Larsen (Eds.), Proc. of Computer-Aided Verification, CAV, LNCS, vol. 2404, Springer, 2002, pp. 78–92.
[25] N. Shankar, H. Rueß, Combining Shostak theories, in: S. Tison. (Ed.), Rewriting Techniques and Applications, RTA, LNCS, vol. 2378, Springer, 2002, pp. 1–18.
[26] R.E. Shostak, An algorithm for reasoning about equality, Communications of the ACM 21 (7) (1978) 583–585.
[27] O. Strichman, On solving Presburger and linear arithmetic with SAT, in: M.D. Aagaard, J.W. O'Leary (Eds.), Formal Methods in Computer-Aided Design, FMCAD, LNCS, vol. 2517, 2002, pp. 160–170.
[28] O. Strichman, S.A. Seshia, R.E. Bryant, Deciding separation formulas with SAT, in: E. Brinksma, K.G. Larsen (Eds.), Proc. of Computer-Aided Verification, CAV, LNCS, vol. 2404, Springer, 2002, pp. 209–222.
[29] J.C. van de Pol, A prover for the μCRL toolset with applications—Version 0.1, Technical Report SEN-R0106, CWI, Amsterdam, 2001.
[30] H. Zantema, Termination of term rewriting, in: M.A. Bezem, J.W. Klop, R.C. de Vrijer (Eds.), Term Rewriting Systems, Cambridge University Press, 2003 (Chapter 6).
[31] H. Zantema, J.C. van de Pol, A rewriting approach to binary decision diagrams, Journal of Logic and Algebraic Programming 49 (1–2) (2001) 61–86.