Two Solutions to Incorporate Zero, Successor and Equality in Binary Decision Diagrams

Bahareh Badban Jaco van de Pol
CWI
P.O.-box 94079, 1090 GB, Amsterdam, The Netherlands
{Bahareh.Badban,Jaco.van.de.Pol}@cwi.nl

ABSTRACT
In this article we extend BDDs (binary decision diagrams) for plain propositional logic to the fragment of first order logic, consisting of quantifier free logic with equality, zero and successor. We insert equations with zero and successor in BDDs, and call these objects $(0, S, =)$-BDDs. We extend the notion of *Ordered* BDDs in the presence of equality, zero and successor. $(0, S, =)$-BDDs can be transformed to equivalent Ordered $(0, S, =)$-BDDs by applying a number of rewrite rules. All paths in these extended OBDDs are satisfiable. 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.

2000 Mathematics Subject Classification: 03B20, 03B25, 03B35, 03B70, 68T15
Keywords and Phrases: Binary Decision Diagrams, Equality, Theorem proving, Decision Procedure, Natural numbers
Note: Research carried out in the NWO project 612.033.009 "Integrating Techniques for the Verification of Distributed Systems

1. Introduction

In this article we consider 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).

In general, the decision problem for plain equational theories is unsolvable already, so we must restrict to particular theories. The decision problem for boolean combinations over equational theories can be approached in several ways. We shortly review what we will call the DNF-method, the 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). This is satisfiable if and only if at least one of its disjuncts is satisfiable. 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 [7] for dealing with linear inequalities. This research was initiated by Shostak [26] and Nelson and Oppen [18]. See also [23, 14]. 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 is not feasible: the resulting formula may be exponentially bigger than the original. This is improved by the BDD-method. In that method, a formula is transformed to a propositionally equivalent *Ordered Binary Decision Diagram* (OBDD), which is a binary directed acyclic graph, that can be seen as a large if-then-else (ITE) tree with shared subterms (see Section 2.1 for more explanation). Although in principle also OBDD representations are exponentially big, it appears that in practice many formulas have a succinct OBDD-representation. In order to solve the satisfiability or tautology problem, each path in the OBDD has to be checked for consistency with respect to the underlying equational theory. A path represents a conjunction of (negated) equations, on which the aforementioned decision procedures can be applied. All inconsistent paths can be removed, resulting in an OBDD with only consistent paths. However, due

to sharing subterms, an OBDD can have exponentially many paths, so still there is a computational bottleneck. A typical example of this approach are the DDDs (difference decision diagrams) of [17], 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 BDD-method, the boolean structure is flattened out immediately, and the arithmetical 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 in propositional logic, 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 BDDs. An early example is Ackermann's reduction [1], by which second order variables can be eliminated. More optimal versions can be found in [15, 20, 11]. 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* [16]), boolean and arithmetic reasoning are not separated, but intertwined. Similar to the BDD-approach, an ordered EQ-BDD is built, but during this procedure, facts from the equational theory are used to prune inconsistent paths at an earlier stage. The main technique is a substitution rule, which allows to replace $ITE(s = t, \phi(s), \psi)$ by $ITE(s = t, \phi(t), \psi)$. It was shown that the resulting normal forms always exist, and have the desirable property that all paths in it are consistent by construction. 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 [16] only the case of equational logic without any function symbols is covered.

Contribution and overview. In [16] BDDs have been extended with equality, resulting in EQ-BDDs. We follow this line of research and extend BDDs to propositional logic with equality, zero and successor, 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 are satisfiable. These normal forms are called Ordered $(0, S, =)$-BDDs. As a result, tautology- and satisfiability checking on Ordered $(0, S, =)$-BDDs can be done in constant time. Actually, we give two different solutions, resulting in two slightly different rewrite systems, and two different sets of normal forms.

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 the first solution is presented, leading to the set of $(0, S, =)$-R-OBDDs (BDDs ordered by representants). First a total and well-founded order on variables is assumed, and extended to a total well-founded order on atomic guards. Then the rewrite system is presented. Finally, we prove termination and satisfiability over all paths. The other sections are devoted to variations on the solution. Section 4 is devoted to some failed attempts. These are included in order to provide some insight in the subtleties of the method. Finally, Section 5 presents the second solution, leading to $(0, S, =)$-E-OBDDs (BDDs ordered by eliminated variables). Intuitively, variables come with a total order, say $y \succ x$. If we know that $y = x$, then y can be eliminated, by substituting the representant x for it. The first solution orders the guards by grouping together the representant variables. In the second solution, the variables to be eliminated are grouped together. Finally, Section 6 concludes with some remarks on implementation and possible applications.

2. Binary Decision Diagrams with Equality

2.1 Binary Decision Diagrams

A *Binary Decision Diagram* [10] (BDD) represents a boolean function as a finite, rooted, binary, ordered, directed acyclic graph. The leafs 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 ($ITE(p, x, x)$) don't 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 [16], Ordered EQ-BDDs are not necessarily unique, so we will not make any attempt to obtain a unique representation in ordered form.

2.2 Adding Equality, Zero, Successor

In this section, we provide the syntax and semantics of BDDs extended with zero, successor and equality. For our purposes, the sharing information present in a graph is not important, 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:*

$$\begin{array}{rcl} W & ::= & 0 \mid V \mid S(W) \\ \Phi & ::= & \bot \mid \top \mid W = W \mid \neg\Phi \mid \Phi \wedge \Phi \mid ITE(\Phi, \Phi, \Phi) \\ G & ::= & \bot \mid \top \mid W = W \\ B & ::= & \bot \mid \top \mid ITE(G, B, B) \end{array}$$

We will use the following conventions: letters $x, y, z, u, \ldots$ denote variables; $r, s, t, \ldots$ will range over W; $\phi, \psi, \ldots$ range over Φ. 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 $S^0(t) = t$ and $S^{m+1}(t) = S(S^m(t))$. 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 define $Var(S^m(u)) \equiv u$. Finally, throughout this paper we will use $\equiv$ to denote syntactic equality between terms or formulas, in order to avoid confusion with the $=$-symbol in guards.

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 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:

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

Given a formula ϕ, we say it is *satisfiable* if there exists a valuation $v : V \to \mathbb{N}$, such that $v(\phi) = 1$; it is a *contradiction* otherwise. If for all $v : V \to \mathbb{N}$, $v(\phi) = 1$, then ϕ is a *tautology*. Finally, if $v(\phi) = v(\psi)$ 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, $ITE(\varphi, \psi, \chi)$ is equivalent to $\neg(\neg(\varphi \wedge \psi) \wedge \neg(\neg\varphi \wedge \chi))$. 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 $\phi_1 \wedge \phi_2$ and $\neg\phi_1$, respectively. ■

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

We now introduce a total ordering on guards. Sorting the guards along the paths of a BDD according to this order, leads to the set of *Representant-Ordered* Binary Decision Diagrams ($(0, S, =)$-R-OBDDs). Next we prove that all $(0, S, =)$-BDDs (and hence all formulas) can be transformed to $(0, S, =)$-R-OBDDs by rewriting. Finally, we show that all paths in $(0, S, =)$-R-OBDDs have a special property, which make them well suited for deciding satisfiability and contradiction of propositional formulas over zero, successor and equality.

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

From now on, we use the term "BDD" as an abbreviation for "$(0, S, =)$-BDD". In this section, we define the set of *representant-ordered* BDDs. Now, before applying an ordering on BDDs, we need some ordering on guards and to define the latter we use a total ordering on the variables. In the sequel, 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 (TRS), 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 obtained after applying the following TRS (Term Rewriting System) simplification rules on it:*

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

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 formula has one of the following forms:*

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

Now let us look at this formula: $\phi := (x = S^3(y)) \wedge (S(x) = y)$. Here we want that $\bot$ becomes the only OBDD which represents ϕ. So the question is how we can obtain $\bot$ through the ordering steps of each BDD representation of ϕ. The first answer which comes to mind might be the substitution of x in $S(x) = y$ by $S^3(y)$ or y in $x = S^3(y)$ by $S(x)$, but none of these two solutions are satisfactory: both substitutions will yield bigger terms, 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 $S^4(x) = S^3(y)$, then substitute $S^3(y)$ by x which is the left-hand-side part of the first equality, So it converts to $S^4(x) = x$ which can be reduced to $\bot$ (by Definition 4). Lifting and substitution are defined below, and we show that in combination with simplification, these operation result in smaller guards.

Definition 6 *Assume m is a natural number, define:*

$$(r = t)^m := S^m(r) = S^m(t) \qquad \text{for each } r, t \in W$$

Given a guard, after we have lifted it by Definition 6, it needs to be decreased through a substitution and simplification rule:

Definition 7 *Suppose g is a simplified guard and y is an element of V which occurs in g, and r is in W and define:*

$$g|_{r=S^m(y)} := (g^m[S^m(y) := r]) \downarrow$$

As it 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 8 *(order on simplified guards) We define total order $\prec$ on simplified guards as below*

- $\bot \prec \top \prec g$, *for all guards g, different from $\top$, $\bot$.*
- $(r_1 = t_1) \prec (r_2 = t_2)$, *iff either $r_1 \prec r_2$, or ($r_1 \equiv r_2$ and $t_1 \prec t_2$), where we use $\prec$ on $W \times W$.*

Definition 9 *A $(0, S, =)$-R-OBDD (representant-ordered $(0, S, =)$-BDD) is a simplified $(0, S, =)$-BDD which is a normal form regarding to the following term rewrite system:*

1. $ITE(\top, T_1, T_2) \rightarrow T_1$
2. $ITE(\bot, T_1, T_2) \rightarrow T_2$
3. $ITE(g, T, T) \rightarrow T$
4. $ITE(g, ITE(g, T_1, T_2), T_3) \rightarrow ITE(g, T_1, T_3)$
5. $ITE(g, T_1, ITE(g, T_2, T_3)) \rightarrow ITE(g, T_1, T_3)$
6. $ITE(g_1, ITE(g_2, T_1, T_2), T_3) \rightarrow ITE(g_2, ITE(g_1, T_1, T_3), ITE(g_1, T_2, T_3))$
   provided $g_1 \succ g_2$
7. $ITE(g_1, T_1, ITE(g_2, T_2, T_3)) \rightarrow ITE(g_2, ITE(g_1, T_1, T_2), ITE(g_1, T_1, T_3))$
   provided $g_1 \succ g_2$
8. *for every simplified $(0, S, =)$-BDD C, we have 8_C :*
   $ITE(S^n(x) = S^m(y),\ C[g], T\ ) \rightarrow ITE(S^n(x) = S^m(y),\ C[\ g|_{S^n(x)=S^m(y)}], T\ )$
   provided y occurs in g and $S^n(x) = S^m(y) \prec g$

In case 8 of the definition above always one of m or n is 0 (Corollary 5). Rules 1–7 are the normal rules for simplifying BDDs for plain propositional logic, which remove redundant tests, and ensure that guards along paths occur in increasing order. Rule 8 allows 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. In the sequel we talk about R-OBDDs instead of $(0, S, =)$-R-OBDDs. We now show an example.

Example 10 *Let $x \prec y \prec z$ (See Figure 1)*

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

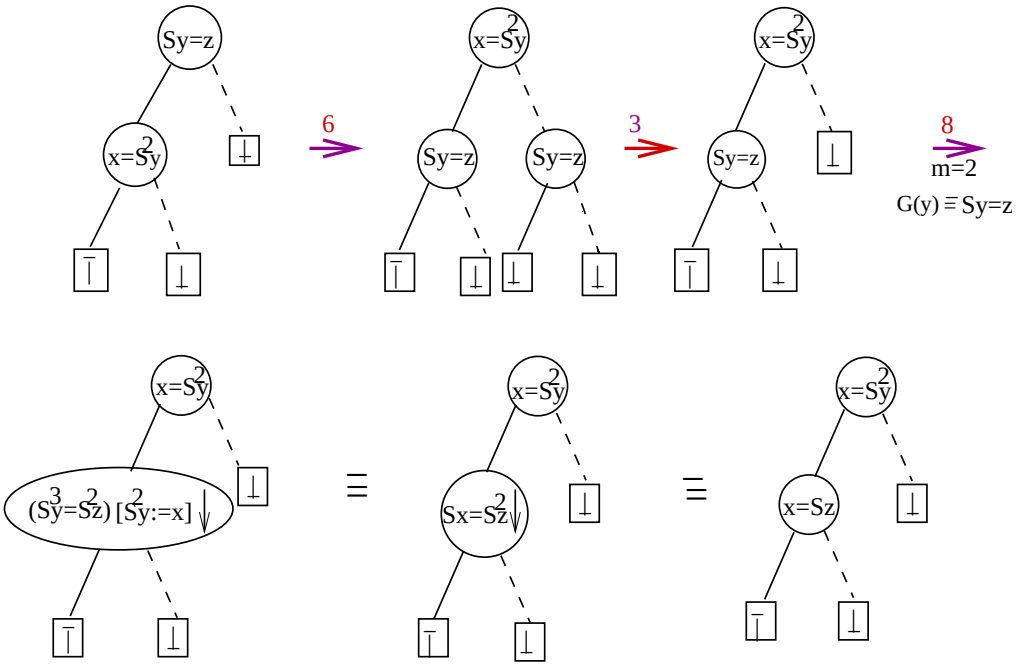

Figure 1: Derivation of Example 10

3.2 Termination

Now we come to the first main claim that every BDD with zero, successor and equality, has a normal form with respect to the TRS above, which means each given BDD has at least one equivalent R-OBDD. It suffices to prove termination: We apply TRS rules to a given BDD, until we reach a normal form after a finite number of steps, which is guaranteed by termination. The so derived BDD is the R-OBDD.

We prove termination by means of a powerful tool, the *recursive path ordering* ($\prec_{rpo}$) [13, 29]. 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 8, 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).* $S \equiv f(S_1, S_2) \succ_{rpo} g(T_1, T_2) \equiv T$ *if and only if*

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

Here $x \succeq_{rpo} y$ *means:* $x \succ_{rpo} y$ *or* $x \equiv y$, *also* $S \succ_{rpo} T_1, T_2$ *means:* $S \succ_{rpo} T_1$ *and* $S \succ_{rpo} T_2$.

This definition yields an order, as it is shown in [29]. In order to prove termination, we will show that each rewrite rule (of Definition 9) is indeed a reduction rule regarding $\succ_{rpo}$. The next lemma will be very helpful to show that this reduction property really holds.

Lemma 12 *Let* $f \equiv S^n(x) = S^m(y)$ *and* $g \equiv S^k(v) = S^l(w)$. *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: $y \equiv v$. *Then* $x \prec y (\equiv v) \prec w$ because f, g are simplified, $g|_f \equiv (g^m[S^m(y) := S^n(x)])\downarrow \equiv (S^{k+n}(x) = S^{l+m}(w))\downarrow$ so

$$g|_f \equiv x = S^{(l+m)-(k+n)}(w) \quad \prec g$$
$$\text{or} \quad g|_f \equiv S^{(k+n)-(l+m)}(x) = w \quad \prec g$$
$$\text{or} \quad g|_f \equiv \bot \quad \prec g$$

- Case II: $y \equiv w$
  $g|_f \equiv (g^m[S^m(y) := S^n(x)]) \downarrow \equiv (S^{k+m}(v) = S^{l+n}(x)) \downarrow$

  if $x \equiv v$ then $g|_f \in \{\top, \bot\} \prec g$
  else $x \prec v$ (because $f \prec g$), so
  $g|_f \equiv x = S^{(k+m)-(l+n)}(v) \prec g$
  or $g|_f \equiv S^{(l+n)-(k+m)}(x) = v \prec g$
  or $g|_f \equiv \bot \prec g$

■

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 $C[g] \succ_{rpo} C[f]$.*

Proof. Monotonicity of $\succ_{rpo}$ [29]. ■

Lemma 14 *Each rewrite rule is contained in $\succ_{rpo}$.*

Proof.

1. $\top(T_1, T_2) \succ_{rpo} T_1$ by (I)
2. Similarly
3. $g(T, T) \succ_{rpo} T$ by (I)
4. $g(g(T_1, T_2), T_3) \succ_{rpo} g(T_1, T_3)$ 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
   - $S \succ_{rpo} T_3$ by (I)
   - $g_2(T_1, T_2) \succ_{rpo} T_1$ by (I)
   - $S \succ_{rpo} T_1$ by (I)

   hence $S \succ_{rpo} g_1(T_1, T_3)$ by (III). Similarly $S \succ_{rpo} g_1(T_2, T_3)$. And therefore $S \succ_{rpo} g_2(g_1(T_1, T_3), g_1(T_2, T_3))$ by (II)
7. Similarly
8. Let $f \equiv S^n(x) = S^m(y)$. Assume y occurs in g and $f \prec g$, and f and g are simplified. We have to show that $f(C[g], T) \succ_{rpo} f(C[\ g|_f\ ], T)$. Now using Lemma 12 we conclude $g \succ g|_f$, and so $C[g] \succ_{rpo} C[g|_f]$ by Lemma 13. Now, by using (I) twice and next (III) it is clear that this rule is also contained in $\succ_{rpo}$.

■

Now we are able to prove our first main claim:

Theorem 15 *The rewrite system defined in Definition 9 is terminating.*

Proof. We showed in the previous Lemma that all rewrite rules are contained in $\succ_{rpo}$. This implies termination, because $\succ_{rpo}$ is a reduction order, i.e. well-founded, and closed under substitutions and contexts [29]. ■

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 R-OBDD.*

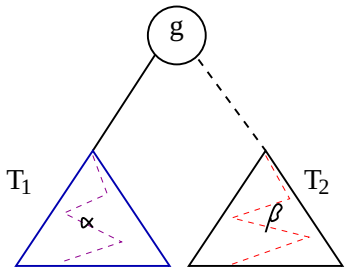

Figure 2: Definition 17

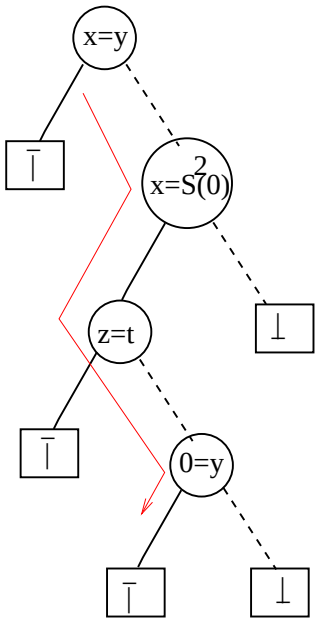

Figure 3: A path from Example 18

3.3 Satisfiability of paths in R-OBDDs

For a given formula, by constructing a BDD representation and then making it ordered by TRS rules then just looking at the result, we will be able to figure out whether the formula is a tautology, a contradiction or a consistency (satisfiable). Below we will explain the details:

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 (see Figure 2):*

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

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 (Figure 3).

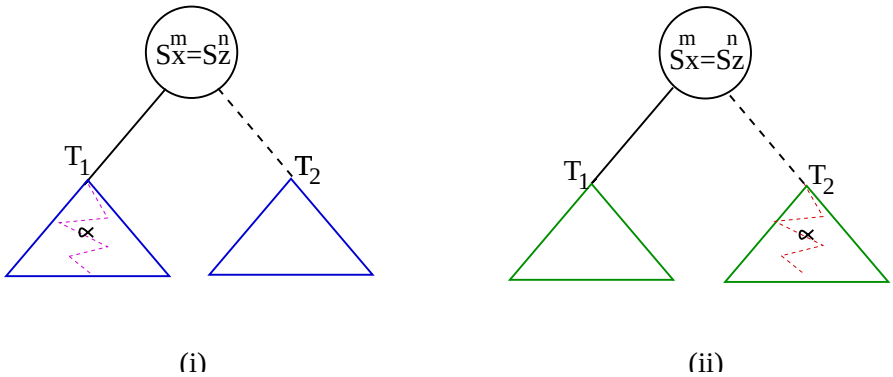

Figure 4: The two cases of Theorem 20

Lemma 19 *Suppose* $T \equiv ITE(S^m(x) = S^n(z), T_1, T_2)$ *is an R-OBDD. Let* α *be a path in* T_2*, let* $H = \{r = t \mid (r = t) \in \alpha \wedge Var(r) \equiv x\}$*, and let* $E = \{x\} \cup \{\, Var(t) \mid r = t \in H\}$*. Then for* $r_0 = t_0 \in \alpha$*, if* $Var(r_0) \in E$ *or* $Var(t_0) \in E$*, then* $r_0 = t_0 \in H$*.*

Proof. Write $H = \{\ S^{j_i}(x) = S^{l_i}(u_i) \mid 1 \leq i \leq k\ \}$. First note that $x \prec u_i$ for all $1 \leq i \leq k$, because H can only contain simplified guards.

- Guards $S^j(x) = S^l(u)$ are in H already, and guards $S^l(u) = S^j(x)$ cannot occur in α, because $u \prec x$ (guards are simplified), so this guard is smaller than $S^m(x) = S^n(z)$, contradicting orderedness of T.
- If there exists a guard like $t = r$ in α with $u_i \in Var(t) \cup Var(r)$ for some $1 \leq i \leq k$, then this guard can not occur below $S^{j_i}(x) = S^{l_i}(u_i)$ (rule 8 of Definition 9), so it should be above this guard
  - If $Var(t) = u_i$ then, because it is placed above $S^{j_i}(x) = S^{l_i}(u_i)$ and T is ordered, $u_i \preceq x$, which is in contradiction with $x \prec u_i$.
  - If $Var(r) = u_i$ then, because it is placed above $S^{j_i}(x) = S^{l_i}(u_i)$, rule 8 of Definition 9 is applicable, which contradicts orderedness of T.

■

The previous lemma gives a syntactical property on R-OBDDs, which can be used for proving satisfiability of each path in an R-OBDD.

Theorem 20 *Each path in an R-OBDD is satisfiable.*

Proof. We prove this theorem by induction over R-OBDDs. Suppose $T \equiv ITE(S^m(x) = S^n(z), T_1, T_2)$ is an R-OBDD, and each path which belongs to T_1 or T_2 is satisfiable, we will show that each path in T is satisfiable as well.

Suppose α is a satisfiable path in T_1 or T_2, so there is a valuation v which satisfies α. Let D be the set of those elements of $\bar{V}$ which occur in α. Now we are going to modify this valuation, in a way that it satisfies $S^m(x) = S^n(z)$ — or its negation, depending on whether α is in T_1 or T_2 — and also still satisfies α.

At first, suppose α belongs to T_1 (see Figure 4(i)). Then $z \notin D$, because T is ordered and rule 8 of Definition 9 would be applicable. Also, $x \prec z$ because all guards are ordered, so $z \not\equiv 0$. From Corollary 5 we obtain: either $n = 0$, or $m = 0$ and $x \not\equiv 0$.
a) if $n = 0$, define:

$$v'(u) = \begin{cases} v(x) + m & \text{if } u \equiv z \\ v(u) & \text{otherwise} \end{cases}$$

It can easily be derived that $v'(S^m(x) = z) = 1$ and since $z \notin D$, also for all $g \in \alpha$, we have $v'(g) = v(g) = 1$.

b) if $n > 0$ and $m = 0$, then according to Corollary 5, $x \not\equiv 0$ and hence $0 \notin D$ ($0 \prec x$, so one of rules 6,7 of Definition 9 would be applicable). Now define:

$$v'(u) = \begin{cases} v(x) & \text{if } u \equiv z \\ v(u) + n & \text{otherwise} \end{cases}$$

$v'(x) = v'(S^n(z))$ obviously, so this valuation satisfies $x = S^n(z)$. Now, let an arbitrary guard $g \equiv S^p(u) = S^q(w)$ be given, such that $g \in \alpha$ (or $\neg g \in \alpha$, respectively). Since $u, w \in D$, and $0, z \notin D$, we have:

$$\begin{aligned} v'(S^p(u)) = v'(S^q(w)) &\iff v'(u) + p = v'(w) + q \\ &\iff v(u) + n + p = v(w) + n + q \\ &\iff v(u) + p = v(w) + q \\ &\iff v(S^p(u)) = v(S^q(w)) \end{aligned}$$

So v' satisfies g (or $\neg g$, respectively). Hence v' will also satisfy the path.

Next, suppose α belongs to T_2 (see Figure 4(ii)). Note that $S^p(y) = S^q(x)$ can not occur in α (for any y, p, q), because then $y \prec x$ and rules 6,7 of Definition 9 will be applicable. So define

$$H = \{\ S^{j_i}(x) = S^{l_i}(u_i)\ \mid 1 \leq i \leq k\ \}$$

being the set of all (positive) guards in α in which x occurs (here x can be 0), and define

$$m' = Max\{j \mid S^j(x) \neq t \text{ occurs in } \alpha \text{ for some t}\}$$

(we set $m' = 0$ if this set is empty). Next, define for $y \in V$:

$$v'(y) = \begin{cases} v(y) & \text{if } y \in \{x, u_1, ..., u_k\} \\ v(y) + v(x) + m + m' + 1 & \text{otherwise} \end{cases}$$

We will now show that v' satisfies both $S^m(x) \neq S^n(z)$ and the path α.

- We first prove that $v'(S^m(x) \neq S^n(z)) = 1$. Note that $v'(S^m(x)) = m + v(x)$, regardless of whether $x \equiv 0$ or not. Also note that $z \not\equiv 0$. We distinguish the following cases:
  - if $z \not\equiv u_i$ for all $1 \leq i \leq k$ then $v'(S^n(z)) = v(z) + v(x) + m + m' + 1$ which is clearly greater than $v'(S^m(x))$ and therefore $v'(S^m(x)) \neq v'(S^n(z))$.
  - if $z \equiv u_i$ for some $1 \leq i \leq k$ then $S^{j_i}(x) = S^{l_i}(z) \in \alpha$. Since our BDD is ordered, either $m < j_i$ or $(m = j_i \wedge n < l_i)$
    - If $m < j_i$, then by Corollary 5, $l_i = 0$:

$$\begin{aligned} v'(S^m(x)) &= v(x) + m \\ &< v(x) + j_i \\ &= v(u_i) + l_i \\ &= v(u_i) \\ &\leq v(u_i) + n \\ &= v'(u_i) + n \\ &= v'(S^n(u_i)) = v'(S^n(z)) \end{aligned}$$

    - If $m = j_i \wedge n < l_i$:

$$\begin{aligned} v'(S^m(x)) &= v(x) + m \\ &= v(x) + j_i \\ &= v(u_i) + l_i \\ &> v(u_i) + n \\ &= v'(u_i) + n \\ &= v'(S^n(u_i)) = v'(S^n(z)) \end{aligned}$$

So in both cases $v'(S^m(x) \neq S^n(z)) = 1$

- Now on the path α. Note that for r, t such that $r = t \in \alpha$, or $r \neq t \in \alpha$, we have $Var(t) \not\equiv 0$ by Corollary 5. Furthermore, if $Var(r) \equiv 0$, then $x \equiv 0$, by the ordering rules. then
  - If $r = t \in \alpha$ then $v(r) = v(t)$
    - If $Var(r)$ or $Var(t) \in \{x, u_1, ...u_k\}$, then according to Lemma 19, $r = t \in H$, therefore $v'(r) = v(r)$ and $v'(t) = v(t)$, so $v'(r = t) = 1$.
    - Otherwise, note that $Var(r) \succ x$ and $Var(r) \succ x$, so both are non-zero. Hence $v'(r) = v(r) + v(x) + m + m' + 1$ and $v'(t) = v(t) + v(x) + m + m' + 1$ therefore $v'(r) = v'(t)$ and thus $v'(r = t) = 1$.
  - If $r \neq t \in \alpha$ then $v(r) \neq v(t)$
    - If neither $Var(r)$ nor $Var(t)$ belongs to $\{x, u_1, ...u_k\}$ then both are non-zero, and $v'(t) = v(t) + v(x) + m + m' + 1$ and $v'(r) = v(r) + v(x) + m + m' + 1$, so $v'(r) \neq v'(t)$ because $v(r) \neq v(t)$, and hence $v'(r \neq t) = 1$.
    - If $Var(r)$ and $Var(t)$ both belong to $\{x, u_1, ...u_k\}$, then $(r = t) \equiv (S^j(x) = S^l(u_i))$ for some $1 \leq i \leq k$ and $j, l \in N$. So by the definition and valuation properties $v'(r) = v(r)$ and $v'(t) = v(t)$. Hence $v'(r) \neq v'(t)$ because $v(r) \neq v(t)$, and $v'(r \neq t) = 1$.
    - If exactly one of $Var(r)$ or $Var(t)$ belongs to $\{x, u_1, ...u_k\}$, then one of these two cases holds:
      - This variable is x and since $Var(t) \neq x$ (because otherwise $r = t \prec S^m(x) = S^n(z)$, and rule 6,7 of Definition 9 would be applicable) then $Var(r) = x$, therefore $r \equiv S^l(x)$ for some $l \in \mathbb{N}$ and thus

$$\begin{aligned} v'(r) &= v'(S^l(x)) \\ &= v'(x) + l \\ &= v(x) + l \\ &\leq v(x) + m' \text{ by definition of } m' \\ &< v(t) + v(x) + m + m' + 1 \\ &= v'(t) \end{aligned}$$

        So $v'(r) \neq v'(t)$ and hence $v'(r \neq t) = 1$
      - This variable is u_i for some $1 \leq i \leq k$. u_i will not occur in any literal below the positive guard $S^{j_i}(x) = S^{l_i}(u_i)$ (rule 8 of Definition 9), so that $r \neq t$ should be placed somewhere above this guard. But now $r = t$ is placed between $S^m(x) = S^n(z)$ and $S^{j_i}(x) = S^{l_i}(u_i)$, so $Var(r) \equiv x$ (by rules 6,7 of Definition 9). This contradicts the fact that only one of r, t contains u_i or x.

■

Corollary 21 *An immediate consequence of Theorem 20 is*

- $\top$ *is the only tautological R-OBDD.*
- $\bot$ *is the only contradictory R-OBDD.*
- *Every other R-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 20, 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 9 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. ■

4. 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 20), 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 [16]. So the most creative part is finding a good ordering on the terms. In this section we present two failed attempts, the first one has non-terminating rewrite sequences, the second one has multiple contradictory OBDDs.

We started our investigations with the ordering of Example 22. It is based on the observation that terms of the form $S^n(x) = y$ are easier to handle than $x = S^n y$. In the former case, all y's can be replaced by $S^n x$, while in the second case, replacing occurrences of $S^n y$ doesn't remove all occurrences of y. 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 22 *Consider the following total ordering on variables and their successors:*

$$... \prec S^2x \prec S^2y \prec S^2z \prec ... \prec Sx \prec Sy \prec Sz \prec ... \prec x \prec y \prec z \prec ...$$

and its lexicographic extension to guards. Moreover consider the rewrite system of Definition 9, over this new ordering. Now look at ϕ below:

$$\phi := (S^2x = y \wedge Sy = z) \vee (S^2x \neq y \wedge (y = S^2z \vee (y \neq S^2z \wedge Sy = z)))$$

In Figure 5 and 6 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 to 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 23 *Consider an alternative ordering on variables and their successors as below:*

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

This order is extended lexicographically on guards. Next, we take rewrite rules 1–8 of Definition 9 w.r.t. to this new ordering. Now look at this formula:

$$\phi := x \neq Sy \wedge z = Sx \wedge z = S^2y$$

ϕ is equivalent to $\bot$, but it has an ordered BDD (w.r.t. the new order) as drawn in Figure 7. This shows that a contradictory OBDD different from $\bot$ exists. The picture shows a path to $\top$, which is unsatisfiable, *so for this ordering, Theorem 20 wouldn't hold.*

Apparently, the occurrences of x in $x = Sy$ and $z = Sx$ 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 allowing also terms of the form $x = S^n y$ is that in the substitution, we have to lift all occurences of y to $S^n y$. This slightly complicates the formulation of rewrite rule 8.

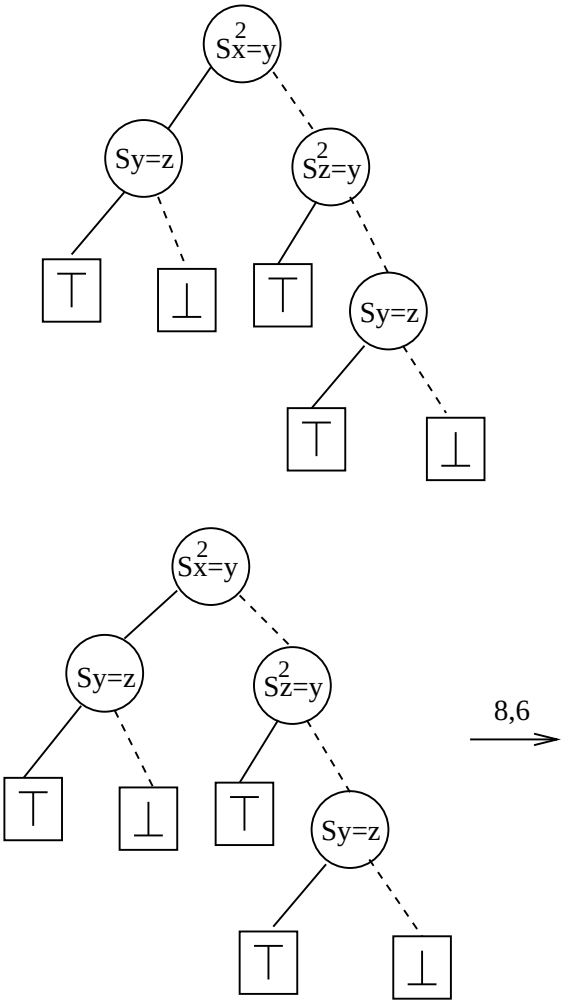

Figure 5: Example 22, see also Figure 6

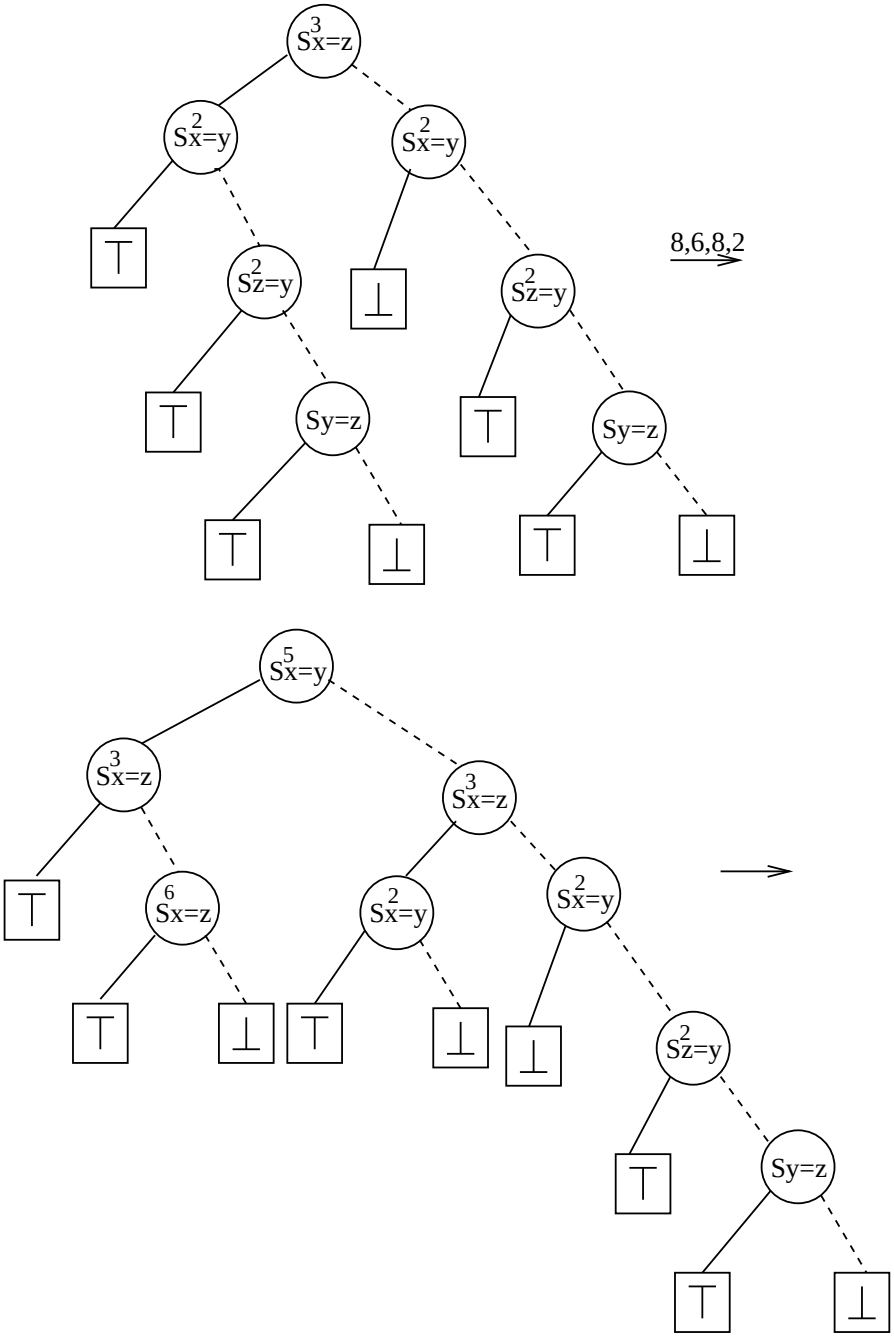

Figure 6: Example 22 (continued)

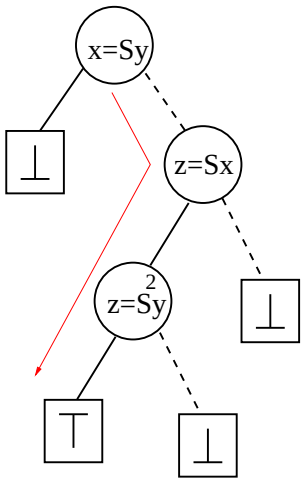

Figure 7: Unsatisfiable path for Example 23

5. Elimination-Ordered $(0, S, =)$-BDDs

5.1 Ordering Based on Eliminated Variables

Definition 24 *Suppose g is a guard. By $g\downarrow$ we mean the normal form of g obtained after applying the following TRS (Term Rewriting System) simplification rules on it:*

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

Corollary 25 *Consequently each simplified formula will be in 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* $x, y \in V,\ x \prec y$
- $\top, \bot$

Definition 26 *Suppose g is a simplified guard and y is an element of V which occurs in g, and r is in W and define:*

$$g|_{S^m(y)=r} := (g^m[S^m(y) := r])\downarrow$$

Regarding Definition 6.

Definition 27 *(order on simplified guards) We define a total order $\prec$ on simplified guards as below*

- $\bot \prec \top \prec g$, *for all guards* g.
- $(S^p(x) = S^q(y)) \prec (S^m(u) = S^n(v))$ *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) $x \equiv u,\ y \equiv v,\ p \equiv m,\ q < n$

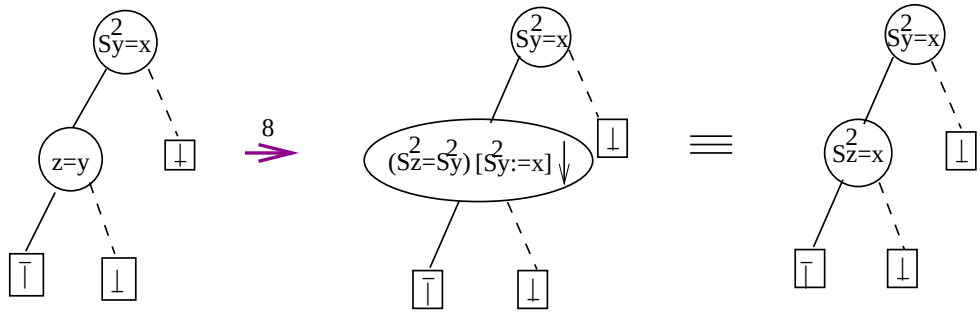

Figure 8: Example 29

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 total and well-founded.

Definition 28 *A $(0, S, =)$-E-OBDD is a simplified $(0, S, =)$-BDD, which is a normal form with respect to the following term rewrite system:*

- $1 - 7$. *Rules $1 - 7$ of Definition 9 (but with $\prec$ from Definition 27)*
- $8'$. *For every simplified $(0, S, =)$-BDD, C we have $8'_C$:*
  $ITE(S^m(y) = S^n(x),\ C[g], T\ ) \rightarrow ITE(S^m(y) = S^n(x),\ C[\ g|_{S^m(y)=S^n(x)}], T\ )$
  provided y occurs in g and $S^m(y) = S^n(x) \prec g$

Example 29 *Let $x \prec y \prec z$ (see Figure 8)*

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

5.2 Termination

Lemma 30 *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

$$\begin{aligned} g|_f &\equiv (g^n[S^n(y) := S^m(x)])\downarrow \\ &\equiv (S^{k+n}(w) = S^{l+m}(x))\downarrow && v \equiv y \\ &\prec S^k(w) = S^l(v) && x \prec v, \text{ Definition 8(ii)} \\ &\equiv g \end{aligned}$$

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

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

  And by Definition 8(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 $v \prec y$.

■

Theorem 31 *The rewrite system defined in Definition 28 is terminating.*

Proof. Definition 11, gives a unique ordering $\succ_{rpo}$ on BDDs, for each given order $\succ$ on guards. So by the new order of Definition 27 on guards, we will have a new $\succ_{rpo}$ on BDDs. Lemma 13 will remain true, regarding this new $\succ_{rpo}$ order, because of its monotonicity. Rewrite rule $8'$ of Definition 28 is contained in new $\succ_{rpo}$ order, because given a guard $f \equiv S^n(y) = S^m(x)$, and $g \succ f$ a literal containing y, then using Lemma 30, we will conclude $g \succ g|_f$. So $C[g] \succ_{rpo} C[g|_f]$ by Lemma 13, and then by Definition 11(I,II), $f(C[g], T) \succ_{rpo} f(C[g|_f], T)$. Rules 1–7 are also contained in $\succ_{rpo}$ by Lemma 14. So that the new rewrite system will be terminating, since $\succ_{rpo}$ is a reduction order (i.e. a well-founded, and closed under substitutions and contexts order), which contains each rule of this system [29]. ■

5.3 Satisfiability of Paths

Lemma 32 *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) With a similar reason as part (ii).

■

Lemma 33 *Supposing $S^l(u) = S^k(y)$ 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 32(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 34 *Suppose $s = t$ is a guard and α is a path. Define:*

$$Reverse(s = t) := t = s$$
$$\bar{\alpha} := \alpha \cup \{\ Reverse(g) \mid g \in\ \alpha\} \cup \{\ \neg Reverse(g) \mid \neg g \in\ \alpha\}$$

Definition 35 *α is an ordered path if it is a path in some E-OBDD.*

Example 36 *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 E-OBDD.

Definition 37 *Supposing α is an ordered path of the form $\beta.(S^m(z) = S^n(x)).\gamma$, we define a set $E_{x\alpha}$ as below:*

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

Remark 38 *According to Definition 37, 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 $E_{x\alpha}$ contain all variables that 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 39 *Given α, an ordered path of the form $\beta.(S^m(z) = S^n(x)).\gamma$, in which $x \in V$ (i.e. $x \not\equiv 0$), and given a valuation v which satisfies this path. Then for each $k \in \mathbb{N}$ exists $l > k$ and a valuation v', such that*

$$\begin{aligned} &(i)\ v' \text{ satisfies } \alpha \\ &(ii)\ v'(u) = v(u) + l && \text{for each } u \in E_{x\alpha} \cup \{x\} \\ &(iii)\ v'(y) = v(y) && \text{for each } y \notin E_{x\alpha} \cup \{x\} \end{aligned}$$

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 32(iii). If it is in δ, this contradicts Lemma 32(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 32(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 32(i). □

Now define:

$$m' = Max\{q + v(y) \mid y \neq x \text{ and } \exists u \in \{x\} \cup E_{x\alpha},\ \exists j \in \mathbb{N} :\ S^j(u) \neq S^q(y) \in \bar{\alpha}\}$$

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 38. 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) that 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 33.

∗ 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:

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

∗ 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:

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

■

Theorem 40 *Each path in an E-OBDD is satisfiable.*

Proof. We prove this theorem by induction over E-OBDDs. Suppose $T \equiv ITE(S^{m_0}(z) = S^{n_0}(x_0), T_1, T_2)$ is an E-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 (Figure 5.3(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 32(i).

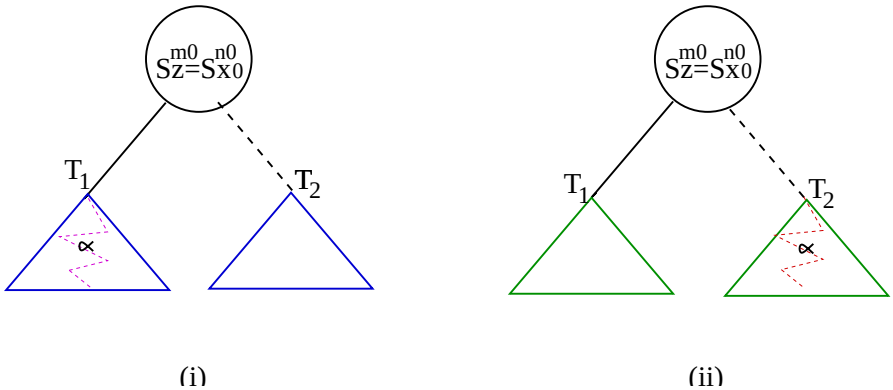

Figure 9: Theorem 40

- If $x_0 \equiv 0$: then $m_0 = 0$, since $S^{m_0}(z) = S^{n_0}(x_0)$ is a simplified guard (Corollary 25). 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 39, 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 (Figure 5.3(ii)), we will provide a new valuation, which will satisfy $(S^{m_0}(z) \neq S^{n_0}(x_0)).\alpha$. Define:

$$H := \bar{\alpha} \cup \{S^{m_0}(z) \neq S^{n_0}(x_0)\}$$

$$L_z := \{\ S^i(y) \mid \exists p \in \mathbb{N},\ \exists u \in E_{z\alpha} \cup \{z\}.\ S^p(u) \neq S^i(y) \in H\ \}$$

$$k := Max\{\ i + v(y) \mid S^i(y) \in L_z\ \}$$

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 $S^p(u) = S^q(z) \in \alpha$ ($S^{m_0}(z) \neq S^{n_0}(x_0)$ is a negative literal). Applying Lemma 39, 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:

  $$\begin{array}{ll} (i)\ v' \text{ satisfies } \alpha & \\ (ii)\ v'(u) = v(u) + l & \text{for each } u \in E_{z\alpha} \cup \{z\} \\ (iii)\ v'(y) = v(y) & \text{for each } y \notin E_{z\alpha} \cup \{z\} \end{array}$$

  Now define

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

  and

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

  Bellow 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 $\bar{\alpha} \cup \{S^{m_0}(z) \neq S^{n_0}(x_0)\}$:

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

  – 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 32(ii), $x_0 \notin E_{z\alpha}$. Hence

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

- $S^m(z) = S^n(x)$ 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 25). $S^{m_0}(z) = S^{n_0}(x_0) \prec S^m(z) = S^n(x)$, therefore $S^{m_0}(z) = S^{n_0}(x_0) \equiv z = S^{n_0}(0)$ according to the Definition 27. v satisfies $z = S^n(0)$ so it also satisfies $z \neq S^{n_0}(0)$, by Lemma 33.

  – If $x \not\equiv 0$:

    * If $x_0 \equiv x$ then, regarding Lemma 33, 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 39, 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 $y \notin E_{x\alpha} \cup \{x\}$.
$x_0 \notin E_{x\alpha} \cup \{x\}$, because $x_0 \not\equiv x$, and $u \succ x$ if $u \in E_{x\alpha}$, by the definition of $E_{x\alpha}$; on the other hand $x_0 \prec x$, because $S^{m_0}(z) \neq S^{n_0}(x_0) \prec S^m(z) = S^n(x)$ (Definition 27). Therefore, if x_0 occurs on α, then, $v'(x_0) = v(x_0)$, otherwise define (without loss of generality, because v' still satisfies α): $v'(x_0) := v(x_0)$. We will show that v' satisfies $S^{m_0}(z) \neq S^{n_0}(x_0)$ too:

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

■

6. Conclusion

We developed the basics for a decision procedure for boolean combinations of equations with zero and successor. First, a formula is transformed into an $(0, S, =)$-BDD. Two rewrite systems on $(0, S, =)$-BDDs are presented, which yield different normal forms. Both systems are proved to be terminating, and both kinds of 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 $\bot$ in both systems. 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 $\phi \leftrightarrow \psi$ is a tautology.

Towards an Implementation. The basic procedure is presented as a term rewrite system. This is still a highly 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 [30] 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 [16] 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, which may also give an indication which of the two methods that we introduced is preferable in practice.

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]). 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). It should be straightforward to extend our method in case all constructors are unary. This would yield a decision procedure for 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*.

Possible Applications. Although the equational fragments that we considered are rather weak (in particular they don't 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 [6, 5].

Propositional logic with equality and uninterpreted functions (EUF) has been proposed for verifying correctness of hardware designs [12]. Also the techniques of [11] are applied to proving equivalence of hardware designs. In [20], similar techniques are applied to the verification of the correctness of compiler optimization results.

Finally, these kind of decision procedures are built into many modern interactive theorem provers, such as PVS [19] and SVC [4]. In this interactive context, the ability to simplify a formula, without deciding it completely, may be a convenient feature. Finally, the automated theorem prover of the μCRL toolset [8, 21] is based on EQ-BDD ideas, and applied in the verification of distributed systems [9].

Acknowledgements. 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 [29] on termination of term rewriting.

Bibliography

[1] W. Ackermann. *Solvable Cases of the Decision Problem.* Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, 1954.

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

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

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

[5] G. Behrmann, K. G. Larsen, J. Pearson, C. Weise, and W. Yi. Efficient timed reachability analysis using clock difference diagrams. In *Proc. of 11th Computer Aided Verification*, pages 341–353, 1999.

[6] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and Wang Yi. UPPAAL: a tool suite for the automatic verification of real-time systems. In R. Alur, T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III*, LNCS 1066, pages 232–243. Springer, 1996.

[7] A.J.C. Bik and H.A.G. Wijshoff. Implementation of Fourier-Motzkin elimination. Technical Report 94-42, Dept. of Computer Science, Leiden University, Leiden, The Netherlands, 1994.

[8] S.C.C. Blom, W.J. Fokkink, J.F. Groote, I. van Langevelde, B. Lisser, and J.C. van de Pol. μCRL: A toolset for analysing algebraic specifications. In G. Berry, H. Comon, and A. Finkel, editors, *Proc. of CAV 2001*, LNCS 2102, pages 250–254. Springer, July 2001.

[9] S.C.C. Blom and J.C. van de Pol. State space reduction by proving confluence. In E. Brinksma and K.G. Larsen, editors, *Proc. of Computer-Aided Verification (CAV)*, LNCS 2404, pages 596–609. Springer, 2002.

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

[11] R.E. Bryant, S. German, and 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.

[12] Burch, J.R. and Dill, D.L. Automatic verification of pipelined microprocessors control. In D.L. Dill, editor, *Proceedings of Computer Aided Verification, CAV'94*, LNCS 818, pages 68–80. Springer, 1994.

[13] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1–2):69–115, 1987.

[14] H. Ganzinger. Shostak light. In A. Voronkov, editor, *Automated Deduction – CADE-18*, LNCS 2392, pages 332–346. Springer, 2002.

[15] A. Goel, K. Sajid, H. Zhou, and A. Aziz. BDD based procedures for a theory of equality with uninterpreted functions. In *Proc. of Computer Aided Verification, CAV'98*, LNCS 1427, pages 244–255. Springer, 1998.

[16] J.F. Groote and J.C. van de Pol. Equational binary decision diagrams. In M. Parigot and A. Voronkov, editors, *Proc. of LPAR 2000*, LNAI 1955, pages 161–178. Springer, 2000.

[17] J. Møller, J. Lichtenberg, H. R. Andersen, and H. Hulgaard. Difference decision diagrams. In *Computer Science Logic*, Denmark, September 1999.

[18] G. Nelson and D.C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, 1980.

[19] S. Owre, S. Rajan, J.M. Rushby, N. Shankar, and M.K. Srivas. PVS: Combining specification, proof checking, and model checking. In R. Alur and T.A. Henzinger, editors, *Proceedings of Computer Aided Verification CAV'96*, LNCS 1102, pages 411–414. Springer, 1996.

[20] A. Pnueli, Y. Rodeh, O. Shtrichman, and M. Siegel. Deciding equality formulas by small domains instantiations. In N. Halbwachs and D Peled, editors, *Proc. of Computer Aided Verification, CAV'99*, LNCS 1633. Springer, 1999.

[21] J.C. van de Pol. A prover for the μCRL toolset with applications – Version 0.1. Technical Report SEN-R0106, CWI, Amsterdam, 2001.

[22] V. Pratt. Two easy theories whose combination is hard. Technical report, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1970.

[23] H. Ruess and N. Shankar. Deconstructing Shostak. In *16th Ann. IEEE Symp. on Logic in Computer Science (LICS '01)*, pages 19–28. IEEE, 2001.

[24] S.A. Seshia, S. Lahiri, and R.E. Bryant. Modeling and verifying systems using a logic of counter arithmetic with lambda expressions and uninterpreted functions. In E. Brinksma and K.G. Larsen, editors, *Proc. of Computer-Aided Verification (CAV)*, LNCS 2404, pages 78–92. Springer, 2002.

[25] N. Shankar and H. Rueß. Combining Shostak theories. In Tison. S., editor, *Rewriting Techniques and Applications (RTA)*, LNCS 2378, pages 1–18. Springer, 2002.

[26] R.E. Shostak. An algorithm for reasoning about equality. *Communications of the ACM*, 21(7):583–585, 1978.

[27] O. Strichman. On solving Presburger and linear arithmetic with SAT. In M.D. Aagaard and J.W. O'Leary, editors, *Formal Methods in Computer-Aided Design (FMCAD)*, LNCS 2517, pages 160–170, 2002.

[28] O. Strichman, S.A. Seshia, and R.E. Bryant. Deciding separation formulas with SAT. In E. Brinksma and K.G. Larsen, editors, *Proc. of Computer-Aided Verification (CAV)*, LNCS 2404, pages 209–222. Springer, 2002.

[29] H. Zantema. Termination of term rewriting. In M.A. Bezem, J.W. Klop, and R.C. de Vrijer, editors, *Term Rewriting Systems*, chapter 6. Cambridge University Press, 2003 (to appear).

[30] H. Zantema and J.C. van de Pol. A rewriting approach to binary decision diagrams. *J. of Logic and Algebraic Programming*, 49(1-2):61–86, 2001.