Centrum voor Wiskunde en Informatica

REPORTRAPPORT

SEN

Software Engineering

Software ENgineering

A BDD-representation for the logic of equality and uninterpreted functions (a full version with proofs)

J.C. van de Pol, O. Tveretina

Report SEN-R0509 June 2005

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-369X

A BDD-representation for the logic of equality and uninterpreted functions (a full version with proofs)

ABSTRACT

The logic of equality and uninterpreted functions (EUF) has been proposed for processor verification. This paper presents a new data structure called Binary Decision Diagrams for representing EUF formulas (EUF-BDDs). We define EUF-BDDs similar to BDDs, but we allow equalities between terms as labels instead of Boolean variables. We provide an approach to build a reduced ordered EUF-BDD (EUF-ROBDD) and prove that every path to a leaf is satisfiable by construction. Moreover, EUF-ROBDDs are logically equivalent representations of EUF-formulae, so they can also be used to represent state spaces in symbolic model checking with data.

2000 Mathematics Subject Classification: 68T15 Theorem Proving
1998 ACM Computing Classification System: 1.2.3 Deduction and Theorem Proving
Keywords and Phrases: Binary Decision Diagrams, the logic of equality with uninterpreted functions, model checking
Note: This research was carried out in the IT-VDS project, granted by NWO as 612.033.009

A BDD-representation for the Logic of Equality and Uninterpreted Functions (a full version with proofs)

Jaco van de Pol[1,2], Olga Tveretina[2]

[1] Centrum voor Wiskunde en Informatica, Dept. of Software Engineering
P.O.-Box 94.079, 1090 GB Amsterdam, The Netherlands
Jaco.van.de.Pol@cwi.nl
[2] Department of Computer Science, TU Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
o.tveretina@tue.nl

Abstract. The logic of equality and uninterpreted functions (EUF) has been proposed for processor verification. This paper presents a new data structure called Binary Decision Diagrams for representing EUF formulas (EUF-BDDs). We define EUF-BDDs similar to BDDs, but we allow equalities between terms as labels instead of Boolean variables. We provide an approach to build a reduced ordered EUF-BDD (EUF-ROBDD) and prove that every path to a leaf is satisfiable by construction. Moreover, EUF-ROBDDs are logically equivalent representations of EUF-formulae, so they can also be used to represent state spaces in symbolic model checking with data.

2000 Mathematics Subject Classification: 68T15 Theorem Proving
1998 ACM Computing Classification System: 1.2.3 Deduction and Theorem Proving
Keywords: Binary Decision Diagrams, the logic of equality with uninterpreted functions, model checking
Note: This research was carried out in the IT-VDS project, granted by NWO as 612.033.009

1 Introduction

Binary Decision Diagrams (BDDs) are one of the biggest breakthroughs in computer-aided design. Reduced ordered BDDs [1] form a canonical representation of Boolean formulas, making testing of equivalence straightforward. Unfortunately, their power is mostly restricted to propositional logic, which is often not sufficiently expressive for verification. The equality logic with uninterpreted functions (EUF) has been proposed for verifying hardware [2]. EUF formulae have been successfully applied for the verification of pipelined processors [2], and translation validation [3].

Using uninterpreted functions simplifies proofs as the only retained information about a function is the property of *functional consistency*, i.e. if $x = y$

then $f(x) = f(y)$. The abstraction process does not preserve validity and may transform a valid formula into an invalid one, e.g. $x + y = y + x$ is valid but $f(x, y) = f(y, x)$ is not. However, in some application domains the process of abstraction is justified.

The original approach to decide this logic was to solve equalities while maintaining *congruence closure* with respect to the uninterpreted functions [4]. This is mainly applied to the conjunction of equalities. Disjunctions can be treated by case splitting [5]. Another approach is based on work of Ackermann [6], who has shown that deciding the validity of EUF formulae can be reduced to checking the satisfiability of pure *equality logic* formulae. Such reduction can be performed by replacing each application of an uninterpreted function symbol with a new variable and for each pair of function applications to add a constraint which enforces the property of functional consistency, i.e. while replacing any two subterms of the form $F(x)$ and $F(y)$ by new variables f_1 and f_2 , we have to add a constraint of the form $x = y \rightarrow f_1 = f_2$.

Due to the *finite domain property*, which states that an equality logic formula is satisfiable if and only if it is satisfiable over a finite domain, Pnueli et al. [3] find a small domain for each variable, which is large enough to maintain satisfiability. Goel et al. [7] proposed to decide equality logic formulae by replacing all equalities with new Boolean variables. Similarly, in [8] a BDD-based decision procedure for combinations of theories is presented. As a result of both approaches, BDDs are not a canonical representation for formulas anymore. Also, there can be paths to a leaf which are not satisfiable. Hence, all paths must be checked, for instance if they satisfy transitivity of equality. Therefore, the constraint solver can be invoked exponentially many times because of the Boolean structure of the formula.

Bryant et al. [9] reduce an equality formula to a propositional one by adding *transitivity constraints*. In that approach it is analyzed which transitivity properties may be relevant. Tveretina et al [10] proposed a resolution-based approach to check satisfiability of equality logic formulae.

In [11], equational BDDs (EQ-BDDs) are defined, in which all paths are satisfiable by construction. That approach extends the notion of orderedness to capture the properties of reflexivity, symmetry, transitivity, and substitutivity. The advantage of the method is that satisfiability checking for a given ordered EQ-BDD can be done immediately. However, it is restricted to the case when equalities do not contain function symbols. EQ-BDDs have been extended in [12]. Here some *interpreted* functions, viz. natural numbers with zero and successor were added. In [13] an alternative solution was provided, with a different orientation of the equations.

Contribution. We introduce EUF-BDDs, which are BDDs with internal nodes labelled by equalities between ground terms. We introduce reduced ordered EUF-BDDs, and prove that these have no contradictory paths. This makes them suitable for theorem proving and satisfiability checking. Moreover, contrary to the approaches to EUF mentioned above, we obtain a representation which is logically equivalent to the original formula. This method extends the approach introduced in [11]. However, the changed orientation of [13] is essential for the completeness of our method. So technically, the EQ-BDDs of [11] are not a special case of our EUF-BDDs, because the orientation of the guards is reversed.

Application. We have made a prototype implementation of our EUF-BDDs in the special purpose theorem prover for the μCRL toolset [14]. The prover is used to discharge proof obligations generated in protocol verifications, and it is also used in the symbolic model checker with data, proposed in [15]. In the latter application it is essential to have a concise representation of formulas, which is provided by EUF-BDDs.

2 Basic Definitions

2.1 Syntax

In this section we define a syntax for formulae. A *signature* is a tuple $\Sigma = (\mathsf{Fun}, \mathsf{ar})$, where $\mathsf{Fun} = \{f, g, h, \dots\}$ is an enumerable set of *function symbols* and $\mathsf{ar} : \mathsf{Fun} \to \mathbf{N}$ is a function describing the arity of the function symbols. Function symbols with the arity 0 are called constants (typically $a, b, c, \dots$). The set of constant symbols is denoted by Const. The set Term of terms is defined inductively: for $n \geq 0$, $f(t_1, \dots, t_n)$ is a term if $t_1, \dots, t_n$ are terms, $f \in \mathsf{Fun}$, and $\mathsf{ar}(f) = n$. For $n = 0$, we write a instead of $a()$. In the following, we use the lower case letters s, t, and u to denote terms. The set $\mathsf{SubTerm}(t)$ of *subterms* of a term t is defined inductively: for $n \geq 0$, $\mathsf{SubTerm}(f(t_1, \dots, t_n)) = \{f(t_1, \dots, t_n)\} \cup \bigcup_{i=1}^{n} \mathsf{SubTerm}(t_i)$. A subterm of a term t is called *proper* if it is distinct from t. The set of proper subterms of a term t is denoted by $\mathsf{SubTerm}_p(t)$.

Definition 1. (*Equalities*) *An* equality *is a pair of terms* $(s, t) \in (\mathsf{Term} \times \mathsf{Term})$*. We write an equality as* $s \approx t$*. The set of equalities over* Σ *is defined by* $\mathsf{Eq}(\Sigma)$ *or if it is not relevant by* Eq.

Here we write '$\approx$' for equality, and we use $\equiv$ to denote syntactical identity between two elements. We define the set of subterms occurring in an equality $s \approx t$ as $\mathsf{SubTerm}(s \approx t) = \mathsf{SubTerm}(s) \cup \mathsf{SubTerm}(t)$, and the set of proper subterms occurring in $s \approx t$ as $\mathsf{SubTerm}_p(s \approx t) = \mathsf{SubTerm}_p(s) \cup \mathsf{SubTerm}_p(t)$.

Definition 2. *Formulae (denoted by* $\mathsf{For}(\Sigma)$*) are expressions satisfying the following syntax.*

$$\varphi := \mathsf{true} \mid \mathsf{false} \mid \mathsf{Eq} \mid \mathsf{ITE}(\varphi, \varphi, \varphi)$$

In the following, the abbreviation $\neg\varphi$ *stands for* $\mathsf{ITE}(\varphi, \mathsf{false}, \mathsf{true})$*,* $\varphi \wedge \psi$ *stands for* $\mathsf{ITE}(\varphi, \psi, \mathsf{false})$*,* $\varphi \vee \psi$ *stands for* $\neg(\neg\varphi \wedge \neg\psi)$*,* $\varphi \to \psi$ *stands for* $\neg\varphi \vee \psi$*, and* $\varphi \leftrightarrow \psi$ *stands for* $(\varphi \to \psi) \wedge (\psi \to \varphi)$*.*

We write $s \not\approx t$ as an abbreviation of $\neg(s \approx t)$. For a given formula φ, the set of all equalities occurring in φ is denoted by $\mathsf{Eq}(\varphi)$.

We define the set of subterms occurring in a formula φ as $\mathsf{SubTerm}(\varphi) = \bigcup_{e \in \mathsf{Eq}(\varphi)} \mathsf{SubTerm}(e)$, and the set of proper subterms occurring in a formula

φ as $\mathsf{SubTerm}_p(\varphi) = \bigcup_{e \in \mathsf{Eq}(\varphi)} \mathsf{SubTerm}_p(e)$. We define the set Lit of *literals* as $\mathsf{Lit} = \{l \mid l \in \mathsf{Eq}\} \cup \{\neg l \mid l \in \mathsf{Eq}\}$. Given a conjunction of literals φ, by $\mathsf{Lit}(\varphi)$ we denote the set of all literals occurring in it.

2.2 Semantics

A *structure* $\mathcal{D}$ over a signature $\Sigma = (\mathsf{Fun}, \mathsf{ar})$ is defined to consist of a non-empty set D called the *domain*, and for every $f \in \mathsf{Fun}$, with $\mathsf{ar}(f) = n$, a map $f_D : D^n \to D$. The *interpretation* $[\![t]\!]_{\mathcal{D}} : \mathsf{Term}(\Sigma) \to D$ of a term t is inductively defined as follows. For $n \geq 0$, $[\![f(t_1, \ldots, t_n)]\!]_{\mathcal{D}} = f_D([\![t_1]\!]_{\mathcal{D}}, \ldots, [\![t_n]\!]_{\mathcal{D}})$, where $t_1, \ldots, t_n \in \mathsf{Term}$. The interpretation $[\![\varphi]\!]_{\mathcal{D}} : \mathsf{For}(\Sigma) \to \{\mathsf{true}, \mathsf{false}\}$ of a formula φ is defined as usual, i.e. $[\![\mathsf{true}]\!]_{\mathcal{D}} = \mathsf{true}$, $[\![\mathsf{false}]\!]_{\mathcal{D}} = \mathsf{false}$, $[\![s \approx t]\!]_{\mathcal{D}} = \mathsf{true}$, if $[\![s]\!]_{\mathcal{D}} = [\![t]\!]_{\mathcal{D}}$, and false otherwise.

$$[\![\mathsf{ITE}(\varphi, \psi, \chi)]\!]_{\mathcal{D}} = \begin{cases} [\![\psi]\!]_{\mathcal{D}} \text{ if } [\![\varphi]\!]_{\mathcal{D}} = \mathsf{true} \\ [\![\chi]\!]_{\mathcal{D}} \text{ otherwise} \end{cases}$$

Definition 3. *A structure $\mathcal{D}$* satisfies *a formula φ if $[\![\varphi]\!]_{\mathcal{D}} = \mathsf{true}$. A formula φ is called* satisfiable *if there exists a satisfying structure. Otherwise φ is called* a contradiction. *If each structure $\mathcal{D}$ satisfies φ then φ is a* tautology. *We say that a formula φ is* logically equivalent *to a formula ψ if for every structure $\mathcal{D}$, $[\![\varphi]\!]_{\mathcal{D}} = [\![\psi]\!]_{\mathcal{D}}$.*

3 Binary Decision Diagrams for EUF-logic

This paper presents a new data structure called an EUF-BDD for representing and manipulating formulas containing *equalities* and *uninterpreted functions*. We consider EUF-BDDs as a restricted subset of formulas.

Definition 4. *We define the set* B *of EUF-BDDs as follows.*

$$\mathsf{B} := \mathsf{true} \mid \mathsf{false} \mid \mathsf{ITE}(\mathsf{Eq}, \mathsf{B}, \mathsf{B})$$

It is straightforward to show that every formula defined above is equivalent to at least one EUF-BDD.

EUF-BDDs are nested ITE formulas which are represented in implementations as directed acyclic graphs. The difference between BDDs representing Boolean formulae and EUF-BDDs is, that in the latter case internal nodes are labelled with equalities. An EUF-BDD can be represented as a rooted, directed acyclic graph with nodes of out-degree zero labelled by true and false, and a set of nodes of out-degree two labelled by equalities between ground terms. For a node l the two outgoing edges are given by two functions $low(l)$ and $high(l)$.

Throughout the paper we use T and S to denote EUF-BDDs.

Example 5. The EUF-BDD representing the property of functional consistency $a \approx b \to f(a) \approx f(b)$ can be depicted as in Figure 1. The EUF-BDD can be written as $\mathsf{ITE}(a \approx b, \mathsf{ITE}(f(a) \approx f(b), \mathsf{true}, \mathsf{false}), \mathsf{true})$.

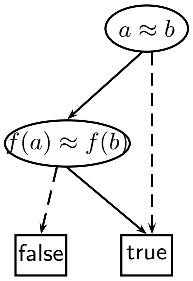

Fig. 1. Dashed lines represent low/false edges, and solid ones represent high/true edges.

In order to define *ordered* EUF-BDDs, we need a total well-founded order on equalities. This is built from a total well-founded order on terms. To ensure structural properties of ordered EUF-BDDs, this total order should satisfy certain properties.

Definition 6. (**Order on terms**) *We define a simplification order on the set* Term *as satisfying the following conditions:*

1. *For all* $s, t \in \mathsf{Term}$*,* $s \prec t$ *if* $s \in \mathsf{SubTerm}_p(t)$*.*
2. *For each* $f \in \mathsf{Fun}$ *and for all* $1 \leq i, j \leq n$*, and* $s_i, t \in \mathsf{Term}$*, if* $s_j \prec t$ *then* $f(s_1, \ldots, s_j, \ldots, s_n) \prec f(s_1, \ldots, t, \ldots, s_n)$*.*
3. *The order is total and well-founded.*
4. $\succ$ *is the reverse of* $\prec$*.*

In the sequel, we work with an arbitrary but fixed simplification order $\prec$. An example of such an order is the recursive path order [16], which we also used in our implementation.

Definition 7. (**Order on equalities**) *Given a simplification order* $\prec$ *on terms, the total well-founded order on the set* Eq *is defined as follows.*

$$(s \approx t) \prec (u \approx v) \text{ if either } s \prec u \text{ or } s \equiv u \text{ and } t \prec v.$$

We use terminology from *term rewrite systems* (TRS). In particular, by a *normal form* with respect to some TRS we mean a term to which no rules of the TRS are applicable. A system is *terminating* if no infinite rewrite sequence exists.

A first operation on EUF-BDDs is simplification of equalities as defined below.

Definition 8. (**Simplified equalities and EUF-BDDs**) *An equality* $s \approx t$ *is called* simplified*, if* $s \succ t$*. In order to simplify all equalities in a BDD, we introduce the following rewrite rules:*

- $s \approx t \to t \approx s$*, for all* $s, t \in \mathsf{Term}$ *such that* $s \prec t$*.*

– $\mathsf{ITE}(t \approx t, T_1, T_2) \to T_1$.

Suppose T is an EUF-BDD. By $T\downarrow$ we mean the normal form of T obtained after applying these rules. An EUF-BDD T is called simplified *if $T \equiv T\downarrow$.*

In the following, by $t[s]$ we mean a term t such that $s \in \mathsf{SubTerm}(t)$, by $e[s]$ we mean an equality such that $s \in \mathsf{SubTerm}(e)$, and by $T[s]$ we mean an EUF-BDD T such that there is a node l in T associated with an equality $e(l)$ and $s \in \mathsf{SubTerm}(e(l))$.

Given the order on equalities, we can define a system of reduction rules as in [11], but now the equations are oriented differently, as in [13]. Now starting with an arbitrary simplified EUF-BDD, we can transform it by repeatedly applying the following reduction rules.

Definition 9. (**Reduction rules on simplified EUF-BDDs**) *We define a TRS* Reduce-Order *as follows.*

1. $\mathsf{ITE}(e, T, T) \to T$
2. $\mathsf{ITE}(e, T_1, \mathsf{ITE}(e, T_2, T_3)) \to \mathsf{ITE}(e, T_1, T_3)$
3. $\mathsf{ITE}(e, \mathsf{ITE}(e, T_1, T_2), T_3)) \to \mathsf{ITE}(e, T_1, T_3)$
4. $\mathsf{ITE}(e_1, \mathsf{ITE}(e_2, T_1, T_2), T_3) \to \mathsf{ITE}(e_2, \mathsf{ITE}(e_1, T_1, T_3), \mathsf{ITE}(e_1, T_2, T_3))$, *if* $e_1 \succ e_2$.
5. $\mathsf{ITE}(e_1, T_1, \mathsf{ITE}(e_2, T_2, T_3)) \to \mathsf{ITE}(e_2, \mathsf{ITE}(e_1, T_1, T_2), \mathsf{ITE}(e_1, T_1, T_3))$, *if* $e_1 \succ e_2$.
6. $\mathsf{ITE}(s \approx t, T_1[s], T_2) \to \mathsf{ITE}(s \approx t, T_1[t]\downarrow, T_2)$, *if* $s \succ t$.

Rules 1−5 are the rules for simplifying BDDs for propositional logic, eliminating redundant tests and ensuring the right ordering. Rule 6 allows to substitute equals for equals. Note that we immediately apply simplification after a substitution. The transformation by the reduction rules yields a logically equivalent EUF-BDD.

Definition 10. (**EUF-ROBDDs**) *We define an EUF-ROBDD to be a simplified EUF-BDD which is a normal form with respect to the TRS* Reduce-Order.

It follows from Definition 10 that in a *reduced ordered* EUF-BDD (EUF-ROBDD) all equalities labelling the nodes are *oriented*, i.e. for a given order $\prec$ on terms, if a node l is associated with an equality $s \approx t$ then $s \succ t$; the equalities along a path appear only in a fixed order; and for each EUF-ROBDD of the form $\mathsf{ITE}(s \approx t, T_1, T_2)$, s doesn't occur in T_1.

Example 11. Consider $\varphi \equiv (x \approx y \wedge y \approx z) \to f(x) \approx f(z)$. For a given order $x \prec y \prec z$, the derivation of an EUF-ROBDD is depicted in Figure 2. The EUF-ROBDD consists of one node true. In the picture, we combined several steps in one arrow. Note that intermediate EUF-BDDs should always be kept simplified. In the middle arrow of the picture, we explicitly show a simplification step.

An EUF-ROBDD is a normal form with respect to the TRS Reduce-Order. The following theorem states that the system of reduction rules is terminating. As a consequence, for each EUF-BDD there exists a logically equivalent EUF-ROBDD.

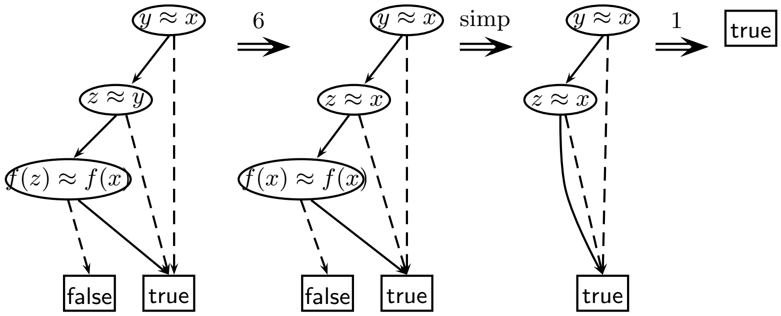

Fig. 2. The derivation of the EUF-ROBDD for $(x \approx y \wedge y \approx z) \rightarrow f(x) \approx f(z)$.

Theorem 12. *The rewrite system* Reduce-Order *is terminating.*

The proof is based on the *recursive path order* (RPO) [16] to prove termination. The details can be found in an appendix.

4 Satisfiability of Paths in EUF-ROBDDs

Checking equivalence of two Boolean functions can be done by comparing their ROBDD representation: equivalent formulas have identical ROBDDs. Unfortunately, the canonicity property of EUF-ROBDDs is violated as is shown for the plain equality case in [11]. In this section we prove that if the EUF-ROBDD corresponding to a formula φ consists of one node true then φ is a tautology, if the EUF-ROBDD consists of one node false then φ is a contradiction, and in all other cases φ is satisfiable. As a consequence, our approach allows to check whether φ and ψ are equivalent. It can be done by verifying whether $\varphi \leftrightarrow \psi$ is a tautology.

When BDDs are used to represent formulas including equalities and uninterpreted functions, a path to the true leaf in the BDD might not be consistent, i.e. the set of literals occurring along the path does not have a model. We show that each path in an EUF-ROBDD is satisfiable by construction.

For proving satisfiability of a path, we see it as a conjunction of literals occurring along the path, where $\wedge$ is considered modulo associativity and commutativity. We use letters α and β to denote finite sequences of literals, ϵ for the empty sequence and $\alpha.\beta$ for the concatenation of sequences α and β.

Definition 13. (**EUF-BDD paths**)

- *We define the set* $\mathsf{Path}(T)$ *of all paths contained in an EUF-BDD T inductively as follows.*
  - $\mathsf{Path}(\mathsf{true}) = \mathsf{Path}(\mathsf{false}) = \epsilon$,
  - $\mathsf{Path}(\mathsf{ITE}(e, T_1, T_2)) = \{e.\alpha \mid \alpha \in \mathsf{Path}(T_1)\} \cup \{\neg e.\alpha \mid \alpha \in \mathsf{Path}(T_2)\}$.

- *For a given path $\alpha \equiv l_1 \ldots . l_n$, we use an abbreviation φ_α to denote a formula $l_1 \wedge \cdots \wedge l_n$.*
- *The formula φ_α* corresponds *to a path α.*
- *We say that α is a* satisfiable path *if φ_α is satisfiable.*

Example 14. Consider an EUF-BDD $\mathsf{ITE}(a \approx b, \mathsf{true}, \mathsf{ITE}(f(a) \approx g(c), \mathsf{ITE}(b \approx c, \mathsf{false}, \mathsf{true})), \mathsf{false})$. The EUF-BDD and the path $a \not\approx b.f(a) \approx g(c).b \not\approx c$ are depicted in Figure 3.

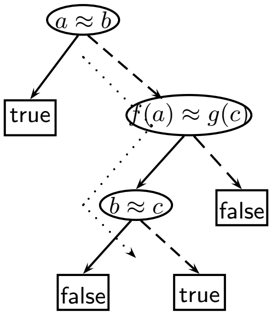

Fig. 3. The path $a \not\approx b.f(a) \approx g(c).b \not\approx c$.

4.1 Satisfiability of Reduced Formulas

To prove satisfiability of paths in EUF-ROBDDs we use a satisfiability criterium from [17]. Before turning to a proof that every path in an EUF-ROBDD is satisfiable, we need to give a definition of a non-propagated equality and a definition of a reduced formula. In [17] the definition of non-propagated equalities is given for CNFs. Here, for sake of simplicity, we rather speak of formulas, but actually we are interested in the case when a formula is a conjunction of literals. Since we see a path as a conjunction of literals, where $\wedge$ is considered modulo associativity and commutativity, this corresponds to a set of unit clauses, as in [17].

Definition 15. (**Non-propagated equality**) *An equality $s \approx t$ is called* non-propagated *in a formula φ if the following holds.*

- *$\varphi \equiv (s \approx t) \wedge \psi$ for some formula ψ.*
- *$s, t \in \mathsf{SubTerm}(\psi)$, and*

The set of all non-propagated equalities in φ is denoted by $\mathsf{NPEq}(\varphi)$.

Definition 16. (**Reduced formula**) *We say that $\varphi \equiv l_1 \wedge \cdots \wedge l_n$, where $l_i \in \mathsf{Lit}$, for all $1 \leq i \leq n$, is* reduced *if the following holds.*

- $\mathsf{NPEq}(\varphi) = \emptyset$, *and*
- *for each* $t \in \mathsf{Term}$, $(t \not\approx t) \notin \mathsf{Lit}(\varphi)$.

In the following Red *is used to denote the set of reduced formulas.*

Theorem 17. *Every* $\varphi \in \mathsf{Red}$ *is satisfiable.*

Proof. See [17]. □

4.2 Satisfiability of EUF-ROBDD Paths

In this section we prove that every path in an EUF-ROBDD is satisfiable. For a given path α, we transform φ_α into the logically equivalent reduced formula φ_α^{red}.

The idea of the proof is that a path in a (simplified) EUF-ROBDD contains segments of the form $s \not\approx t_0. \cdots .s \not\approx t_n.s \approx t$. The term s doesn't occur as a subterm in any t_i, nor in any of the other segments. We obtain a path corresponding to $\varphi_\alpha^{red} \in \mathsf{Red}$ by propagating $s \approx t$, i.e. replacing the segment by $t \not\approx t_0. \cdots .t \not\approx t_n.s \approx t$. Note that this operation doesn't introduce new subterms, so propagated equalities in other segments remain propagated. The result is an equivalent formula in Red, hence it is satisfiable.

Example 18. For a given order $a \prec b \prec c \prec f(d) \prec g(a)$, the EUF-ROBDD representation of $\mathsf{ITE}(f(d) \approx a, \mathsf{true}, \mathsf{ITE}(f(d) \approx b, \mathsf{true}, \mathsf{ITE}(f(d) \approx c, \mathsf{ITE}(g(a) \approx c, \mathsf{true}, \mathsf{false}), \mathsf{false})))$ is depicted in Figure 4.

Consider the path $f(d) \not\approx a.f(d) \not\approx b.f(d) \approx c.g(a) \approx c$. The formula corresponding to the path contains one non-propagated equality $f(d) \approx c$. By propagating this equality, i.e. replacing $f(d) \not\approx a \wedge f(d) \not\approx b$ with $c \not\approx a \wedge c \not\approx b$, we obtain a reduced formula $c \not\approx a \wedge c \not\approx b \wedge f(d) \approx c \wedge g(a) \approx c$.

The formula corresponding to the path and the reduced formula are logically equivalent. By Theorem 17, the reduced formula is satisfiable. Therefore, the path is also satisfiable.

Theorem 19. *Every path in an EUF-BDD is satisfiable.*

The full proof can be found in an appendix.

Corollary 20. *From Theorem 19*

- *The only tautological EUF-ROBDD is* true.
- *The only contradictory EUF-ROBDD is* false.
- *All other EUF-ROBDDs are satisfiable.*

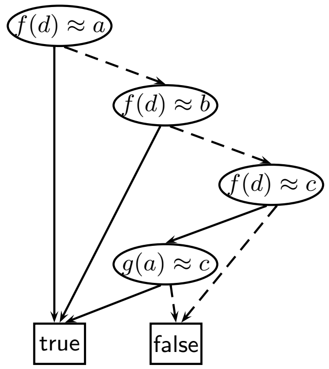

Fig. 4. The EUF-ROBDD representation of a formula $\mathsf{ITE}(f(d) \approx a, \mathsf{true}, \mathsf{ITE}(f(d) \approx b, \mathsf{true}, \mathsf{ITE}(f(d) \approx c, \mathsf{ITE}(g(a) \approx c, \mathsf{true}, \mathsf{false}), \mathsf{false})))$.

5 Implementation and Applications

We implemented our proposal for EUF-BDDs within the special purpose theorem prover for the μCRL toolset [14]. The language μCRL combines abstract data types with process algebra. The prover is used to discharge proof obligations generated in protocol verifications, in particular to prove process invariants and confluence of internal computation steps [18]. It is also used in the symbolic model checker with data, proposed in [15]. In the latter application it is essential to have a concise representation of formulas, which is provided by EUF-BDDs.

Usually, symbolic model checking uses ordered binary decision diagrams to provide a compact representation of the transition system. BDD-based model checking performs an exhaustive traversal of the model by considering all possible behaviors in a compact way. Such exhaustive exploration allows BDD based model checking algorithms to conclude whether a given property is satisfied.

In a similar way, the symbolic model checker with data represents a possibly infinite state space by BDDs extended with equalities and function symbols. In this case, the main operation for the model checker is to compute the sequence of EUF-ROBDDs $\Phi^n(\bot)$ [for some operator Φ]. A fixed point has been reached as soon as the formula $\Phi^n(\bot) \iff \Phi^{n+1}(\bot)$ is a tautology, which can be checked by our method.

As input, the prover takes a data specification consisting of a signature of constructor and defined symbols, and a set of equations. It also takes a quantifier-free formula as input, and it returns a logically equivalent EUF-ROBDD. If the result is either true or false, we know for sure that the formula is a tautology or a contradiction, respectively. For the other cases, we would like to conclude that both the formula and its negation are satisfiable. However, this is only possible for certain fragments. We call the prover complete for such fragments.

The previous implementation of this theorem prover [19] was based on EQ-BDDs [11], and consequently it was only complete for the case of equality logic with equations between variables. The implementation also used plain term rewriting with equations from the abstract datatype. It was sound for any data specification, but not complete.

The current implementation is based on the observations in this paper. Consequently, it is now complete for the theory with equality and uninterpreted function symbols. It is sound – but incomplete – for the case that functions denote constructors, or when they are specified by means of equations.

In the current implementation, we use the reversed equation order as introduced in [13]. Moreover, we use the lexicographic path order to compare terms; this order satisfies the conditions of Definition 6. Given a formula φ, we find the smallest equation $t \approx s$ in it, then recursively compute the EUF-BDDs A of $\varphi[t := s]$ and B of $\varphi[t \approx s := \mathsf{false}]$, and return $\mathsf{ITE}(t \approx s, A, B)$. This procedure must be repeated in order to obtain an EUF-ROBDD.

The new prover was applied to many existing case studies (see for instance [18, 14] for a description). It is confirmed that the new prover can handle more formulas. Moreover, it was never slower than the version of [19].

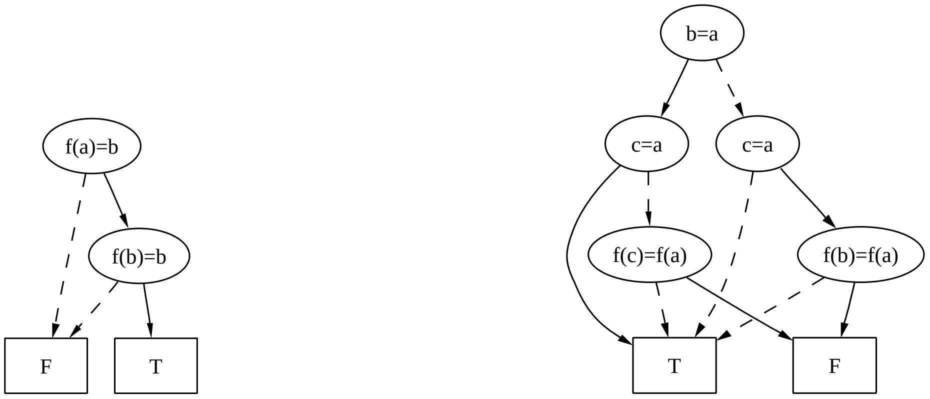

Fig. 5. EUF-ROBDDs obtained by the implementation

The resulting EUF-ROBDD can be visualized (using graphviz/dot) for small formulas. Figure 5 shows the EUF-ROBDDs for the formulas $f(f(f(f(f(a))))) \approx b \wedge f(f(a)) \approx f(a)$ and $(f(b) \approx f(c) \Rightarrow a \approx b) \Leftrightarrow (f(b) \approx f(c) \Rightarrow a \approx c)$, respectively.

6 Conclusions

We have extended the approach from [11] in the presence of uninterpreted function symbols, and the changed orientation of [13] is essential for the completeness of our method. Starting from the EUF-BDD representing an arbitrary EUF formula and applying rewrite rules of a rewrite system Reduce-Order, a normal form, called an EUF-ROBDD, can be calculated. We proved that all paths in a EUF-ROBDD are satisfiable by construction. our approach is suitable for checking tautology, satisfiability, and equivalence of formulas. A prototype implementation of this method works within the special purpose theorem prover for the μCRL toolset [14].

Future work. We have not yet studied strategies for choosing an ordering on equalities. A good ordering is crucial since it yields a compact representation: for some Boolean functions, the ROBDD sizes are linear in the number of variables for one ordering, and exponential for another.

It is interesting to extend our methods beyond the EUF fragment. The current implementation handles any data specified by a TRS, but is in general incomplete. The study in [12] shows that finding complete extensions may be hard.

References

1. Bryant, R.: Symbolic boolean manipulation with ordered binary decision diagrams. ACM Computing Surveys **24** (1992) 293–318
2. Burch, J., Dill, D.: Automated verification of pipelined microprocesoor control. In Dill, D., ed.: Computer-Aided Verification (CAV'94). Volume 818 of LNCS., Springer-Verlag (1994) 68–80
3. Pnueli, A., Rodeh, Y., Shtrichman, O., Siegel, M.: The small model property: how small can it be? Information and Computation **178** (2002) 279 – 293
4. Nelson, G., Oppen, D.: Fast decision procedures based on congruence closure. Journal of the ACM **27 (2)** (1980) 356 – 364
5. Shostak, R.: An algorithm for reasoning about equality. Communications of the ACM **21** (1978) 583–585
6. Ackermann, W.: Solvable cases of the decision problem. Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam (1954)
7. Goel, A., Sajid, K., Zhou, H., Aziz, A., Singhal, V.: BDD based procedures for a theory of equality with uninterpreted functions. In Hu, A.J., Vardi, M.Y., eds.: Computer-Aided Verification (CAV'98). Volume 1427 of LNCS., Springer-Verlag (1998) 244–255
8. Fontaine, P., Gribomont, E.P.: Using BDDs with combinations of theories. In Baaz, M., Voronkov, A., eds.: Logic for Programming and Reasoning (LPAR'2002). Volume 2514 of LNCS., Springer-Verlag (2002) 190–201
9. Bryant, R., Velev, M.: Boolean satisfiability with transitivity constraints. ACM Transactions on Computational Logic **3** (2002) 604–627
10. Tveretina, O., Zantema, H.: A proof system and a decision procedure for equality logic. In Farach-Colton, M., ed.: LATIN 2004: Theoretical Informatics. Volume 2976 of LNCS. (2004) 530–539
11. Groote, J., van de Pol, J.: Equational binary decision diagrams. In Parigot, M., Voronkov, A., eds.: Logic for Programming and Reasoning (LPAR'2000). Volume 1955 of LNAI. (2000) 161–178
12. Badban, B., van de Pol, J.: Zero, sucessor and equality in BDDs. Annals of Pure and Applied Logic **133/1-3** (2005) 101–123
13. Badban, B., van de Pol, J.: An algorithm to verify formulas by means of (0,s,=)-BDDs. In: Proceedings of the 9th Annual Computer Society of Iran Computer Conference (CSICC 2004), Tehran, Iran (2004)

14. Blom, S., Groote, J., van Langevelde, I., Lisser, B., van de Pol, J.: New developments around the μCRL tool set. In: Proceedings of FMICS 2003. ENTCS volume 80 (2003)
15. Groote, J., Willemse, T.: Parameterised boolean equation systems (extended abstract). In Gardner, P., Yoshida, N., eds.: Proceedings of CONCUR 2004. LNCS 3170 (2004) 308–324
16. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
17. Tveretina, O.: A decision procedure for equality logic with uninterpreted functions. In Buchberger, B., Campbell, J., eds.: Artificial Intelligence and Symbolic Mathematical Computation. Volume 3249 of LNAI., Springer-Verlag (2004) 63–76
18. Blom, S., van de Pol, J.: State space reduction by proving confluence. In Brinksma, E., Larsen, K., eds.: Proceedings of CAV'02. LNCS 2404, Springer (2002) 596–609
19. van de Pol, J.: A prover for the μCRL toolset with applications – Version 0.1. Technical Report SEN-R0106, CWI, Amsterdam (2001)

A Proof of Theorem 12

We use *recursive path order* (RPO) [16] to prove termination. The main idea of recursive path order is that two terms are compared by first comparing their root symbols and then recursively comparing their subterms. We do it by viewing BDDs as binary trees, i.e. $\mathsf{ITE}(e, T_1, T_2)$ can be seen as the tree $e(T_1, T_2)$.

Definition 21. (*Recursive path order for BDDs*) *We say that for simplified EUF-BDDs S and T, $S \succ_{rpo} T$, where $S \equiv f(S_1, S_2), T \equiv g(T_1, T_2)$, if one of the following holds.*

1. *$S_1 \succeq_{rpo} T$ or $S_2 \succeq_{rpo} T$.*
2. *$f \succ g$, and $S \succ_{rpo} T_1$ and $S \succ_{rpo} T_2$.*
3. *$f \equiv g$, and $S \succ_{rpo} T_1$ and $S \succ_{rpo} T_2$, and either $S_1 \succ_{rpo} T_1$ or ($S_1 \equiv T_1$ and $S_2 \succ_{rpo} T_2$).*

Lemma 22. *Let $T[s]$ be a simplified EUF-BDD, and s and t be terms such that $s \succ t$, for some order satisfying Definition 6. Then $T[s] \succ_{rpo} T[t]\downarrow$.*

Proof. Suppose $T[s]$ contains an internal node labelled with en equality $e[s]$. Consider $e[t]\downarrow$. Then one of the following holds.

- $e[t]\downarrow \equiv \mathsf{true}$.
  Suppose $T[s] \equiv e[s](T_1, T_2)$ contains one internal node which is labelled by an equality $e[s]$.
  By Definition 8, $T[t]\downarrow \equiv T_1$. Therefore, taking into account Definition 21(1), $T[s] \succ_{rpo} T[t]$.
  For induction hypothesis assume that for each simplified EUF-BDD $T'[s]$ containing at most $n-1$ internal nodes, $T'[s] \succ_{rpo} T'[t]\downarrow$.
  Suppose $T[s] \equiv e(T_1, T_2)$ has n internal nodes.
  If $s \in \mathsf{SubTerm}(e)$ then by Definition 8, $T[t]\downarrow \equiv T_1$. Therefore, taking into account Definition 21(1), $T[s] \succ_{rpo} T[t]$.
  If $s \notin \mathsf{SubTerm}(e)$ then s occurs either in T_1 or in T_2. By induction hypothesis, if s occurs in T_1, then $T_1[s] \succ T_1[t]\downarrow$, and if s occurs in T_2, then $T_2[s] \succ T_2[t]\downarrow$. Hence, by 1 and 3 of Definition 21, $T[s] \succ T[t]\downarrow$.
- $e[t]\downarrow \not\equiv \mathsf{true}$.
  First we show that for each equality $e[s]$, $e[s]\downarrow \succ e[t]\downarrow$.
  Suppose $e[s]\downarrow \equiv u \approx v$ and $e[t] \equiv u' \approx v'$. Then one of the following holds.
  - $u' \succ v'$. Then $e[t]\downarrow \equiv u' \approx v'$, and one of the following holds.
    - $u' \prec u$. In this case by Definition 7, $e[s]\downarrow \succ e[t]\downarrow$.
    - $u' \equiv u$ and $v' \prec v$. By Definition 7, $e[s]\downarrow \succ e[t]\downarrow$.
  - $u' \prec v'$. Then $e[t]\downarrow \equiv v' \approx u'$. Since $v' \preceq v \prec u$, by Definition 7, $e[s]\downarrow \succ e[t]\downarrow$.

  Suppose $T[s]$ contains one internal node labelled by an equality $e[s]$. Since $T[s]$ is simplified, $e[s] \equiv e[s]\downarrow$. Taking into account that $e[s] \succ e[t]\downarrow$ and 1 and 2 of Definition 21, we obtain $T[s] \succ_{rpo} T[t]\downarrow$.

For induction hypothesis assume that for each simplified EUF-BDD $T'[s]$ containing at most $n-1$ internal nodes, $T'[s] \succ_{rpo} T'[t]\downarrow$.
Suppose $T[s] \equiv e(T_1, T_2)$ has n internal nodes.
If $s \in \mathsf{SubTerm}(e)$ then $e \succ e[t]\downarrow$. Therefore taking into account 1 and 2 of Definition 21, we obtain $T[s] \succ T[t]\downarrow$.
If $s \notin \mathsf{SubTerm}(e)$ then s occurs either in T_1 or in T_2. By induction hypothesis, if s occurs in T_1, then $T_1[s] \succ T_1[t]\downarrow$, and if s occurs in T_2, then $T_2[s] \succ T_2[t]\downarrow$. Hence, by 1 and 3 of Definition 21, $T[s] \succ T[t]\downarrow$.

□

Now we give a proof of Theorem 12.

Proof. We show that every rewrite rule is contained in $\succ_{rpo}$.

1. $e(T, T) \succ_{rpo} T$ by 1 of Definition 21.
2. $e(T_1, e(T_2, T_3)) \succ_{rpo} e(T_1, T_3)$ by 1 and 3 of Definition 21.
3. $e(e, T_1, T_2), T_3)) \succ_{rpo} e(T_1, T_3)$ by 1 and 3 of Definition 21.
4. Suppose $e_1 \succ e_2$. Then
   – $e_1(e_2(T_1, T_2), T_3) \succ_{rpo} e_1(T_1, T_3)$ by 1 and 3 of Definition 21.
   – $e_1(e_2(T_1, T_2), T_3) \succ_{rpo} e_1(T_2, T_3)$ by 1 and 3 of Definition 21.
   We conclude $e_1(e_2(T_1, T_2), T_3) \succ_{rpo} e_2(e_1(T_1, T_3), e_1(T_2, T_3))$ by 2.
5. Suppose $e_1 \succ e_2$. Then
   – $e_1(T_1, e_2(T_2, T_3)) \succ_{rpo} e_1(T_1, T_2)$ by 1 and 3.
   – $e_1(T_1, e_2(T_2, T_3)) \succ_{rpo} e_1(T_1, T_3)$ by 1 and 3.
   We conclude $e_1(T_1, e_2(T_2, T_3)) \succ_{rpo} e_2(e_1(T_1, T_2), e_1(T_1, T_3))$ by 2.
6. Suppose $s \succ t$. Then
   $T_1[s] \succ_{rpo} T_1[t]\downarrow$ by Lemma 22.
   Hence $(s \approx t)(T_1[s], T_2) \succ_{rpo} (s \approx t)(T_1[t]\downarrow, T_2)$ by 1 and 3 of Definition 21.

□

B Proof of Theorem 19

Definition 23. (**Proper formulas**) *We say that a formula $\varphi \equiv l_1 \wedge \cdots \wedge l_n$ is* proper *if the following holds.*

1. *for every $(s \approx t) \in \mathsf{Eq}(\varphi)$, $s \succ t$.*
2. *for every $s \approx t \in \{l_1, \ldots, l_n\}$ and for every $e \in \mathsf{Eq}(\varphi)$ if $e \succ (s \approx t)$ then $s \notin \mathsf{SubTerm}(e)$.*

Lemma 24. *A formula φ corresponds to a path in an EUF-ROBDD if and only if it is a proper formula.*

Proof. Assume $\varphi \equiv l_1 \wedge \cdots \wedge l_n$ and $l_1.\ldots.l_n$ is a corresponding path in an EUF-BDD.

($\Rightarrow$) Suppose a formula φ corresponds to a path in an EUF-ROBDD. We have to check Definition 23.

1. Definition 23(1) holds (otherwise not all equalities are simplified).
2. Definition 23(2) holds(otherwise Rule 6 would be applicable).

Hence, φ is a proper formula.

($\Leftarrow$) Suppose φ is a proper formula. W.l.o.g. we can assume that all $i \neq j$, $l_i \not\equiv l_j$.

Consider the path $l_1. \ldots .l_n$. Since we consider $\wedge$ modulo associativity, w.l.o.g. we can assume that for every $1 \leq i < j \leq n$, $e(l_i) \prec e(l_j)$.

All equalities are simplified by Definition 23(1). By Definition 23(2), for every $l_i \equiv s_i \approx t_i$, $1 \leq i < n$, $s_i \notin \mathsf{SubTerm}(l_{i+1}, \ldots, l_n)$.

Hence, no rule of Defenition 9 is applicable. Therefore, $l_1. \ldots .l_n$ is a path in an EUF-ROBDD. □

Lemma 25. *Let for $s, t \in \mathsf{Term}$, $\alpha.(s \approx t).\beta$ be a path in an EUF-ROBDD. Then the following holds.*

1. *For every $(s' \approx t') \in \alpha \cup \beta$, $s \notin \mathsf{SubTerm}(s' \approx t')$.*
2. *$s \notin \mathsf{SubTerm}_p(\alpha.(s \approx t).\beta)$.*

Proof. 1. *Since α is a path in EUF-ROBDD, then $s \notin \mathsf{SubTerm}(\beta)$ (otherwise Rule 6 of Definition 9 would be applicable.)*
Consider

$$(u \approx v) \in \alpha$$

Then the following holds (we use the first property of Definition 6).

- *$s \notin \mathsf{SubTerm}(u)$ since $u \prec s$ (if $u \equiv s$ then Rule 6 of Definition 9 would be applicable, and if $u \succ s$, Rule 4 or 5 would be applicable).*
- *$s \notin \mathsf{SubTerm}(v)$ since $v \prec u \preceq s$ (every EUF-ROBDD is simplified).*

Hence,

$$s \notin \mathsf{SubTerm}(u \approx v).$$

2.
- *$s \notin \mathsf{SubTerm}_p(\beta)$ (otherwise Rule 6 of Definition 9 would be applicable)*
- *For each $(u \approx v) \in \alpha$, $s \notin \mathsf{SubTerm}_p(u \approx v)$ by Lemma 25(1).*
- *For each $(u \not\approx v) \in \alpha$,*
  - *$s \notin \mathsf{SubTerm}_p(u)$ since $u \preceq s$ (otherwise Rule 4 or Rule 5 of Definition 9 would be applicable), and*
  - *$s \notin \mathsf{SubTerm}_p(v)$ since $v \prec u \preceq s$ (the EUF-ROBDD is a simplified EUF-BDD).*
- *$s \notin \mathsf{SubTerm}_p(t)$ because $t \prec s$ (the EUF-BDD is simplified).*

Hence,

$$s \notin \mathsf{SubTerm}_p(\alpha.s \approx t.\beta).$$

□

Suppose $s, t \in \mathsf{Term}$ and $s \notin \mathsf{SubTerm}(t)$. We denote by $\varphi[s := t]$ the formula obtained from φ, where all occurrences of s are repeatedly replaced by t until no occurrence of s is left. At first we prove a technical lemma which we use in a proof of Lemma 27.

Lemma 26. *Suppose $\alpha \equiv \alpha_1.(s \approx t).\alpha_2$ is a path in an EUF-ROBDD, $\beta \equiv \beta_1.(s \approx t).\beta_2$, where $\beta_1 \subseteq \alpha_2$ and $\beta_2 \subseteq \alpha_2$.*

Then $\mathsf{SubTerm}(\varphi_\beta) = \mathsf{SubTerm}(\varphi_{\beta_1}[s := t] \wedge (s \approx t) \wedge \varphi_{\beta_2}[s := t])$.

Proof. We use a trivial observation that for $s, t, u \in \mathsf{Term}$, if $s \notin \mathsf{SubTerm}(u)$ then $\mathsf{SubTerm}(u[s := t]) \equiv \mathsf{SubTerm}(u)$. By Lemma 25, $s \notin \mathsf{SubTerm}_p(\alpha_1 \cup \alpha_2)$. Since $(\beta_1 \cup \beta_2) \subseteq (\alpha_1 \cup \alpha_2)$, we conclude $s \notin \mathsf{SubTerm}_p(\beta_1 \cup \beta_2)$.

Hence,

$$\begin{aligned}\mathsf{SubTerm}(\varphi_\beta) &= \mathsf{SubTerm}(\varphi_{\beta_1} \wedge (s \approx t) \wedge \varphi_{\beta_2})\\ &= \mathsf{SubTerm}(\varphi_{\beta_1} \wedge \varphi_{\beta_2}) \cup \mathsf{SubTerm}(\{s \approx t\})\\ &= \mathsf{SubTerm}(\varphi_{\beta_1}[s := t] \wedge \varphi_{\beta_2}[s := t]) \cup \mathsf{SubTerm}(\{s \approx t\})\\ &= \mathsf{SubTerm}(\varphi_{\beta_1}[s := t] \wedge (s \approx t) \wedge \varphi_{\beta_2}[s := t])\end{aligned}$$

□

Lemma 27. *Let $\alpha \equiv \alpha_1.(s \approx t).\alpha_2$ be a path in an EUF-ROBDD, and $(s \approx t) \in \mathsf{NPEq}(\varphi_\alpha)$, where $s, t \in \mathsf{Term}$. Then the following holds.*

1. $|\mathsf{NPEq}(\varphi_{\alpha_1}[s := t] \wedge (s \approx t) \wedge \varphi_{\alpha_2}[s := t])| < |\mathsf{NPEq}(\varphi_\alpha)|$.
2. *$\varphi_{\alpha_1}[s := t] \wedge (s \approx t) \wedge \varphi_{\alpha_2}[s := t]$ is a proper formula.*

Proof. 1. By Lemma 26

$$\mathsf{SubTerm}(\varphi_{\alpha_1}[s := t] \wedge (s \approx t) \wedge \varphi_{\alpha_2}[s := t]) = \mathsf{SubTerm}(\varphi_\alpha).$$

Let us consider an arbitrary equality $(u \approx v) \in \alpha$ such that
- $(u \approx v) \not\equiv (s \approx t)$,
- $(u \approx v) \notin \mathsf{NPEq}(\varphi_\alpha)$.

By Lemma 25(1),

$$s \notin \mathsf{SubTerm}(u \approx v).$$

Taking into account a trivial observation that for $s, t, u \in \mathsf{Term}$, if $s \notin \mathsf{SubTerm}(u)$ then $\mathsf{SubTerm}(u[s := t]) \equiv \mathsf{SubTerm}(u)$, we obtain

$$(u \approx v)[s := t] \equiv (u \approx v).$$

We wil show that $(u \approx v) \notin \mathsf{NPEq}(\varphi_{\alpha_1}[s := t] \wedge (s \approx t) \wedge \varphi_{\alpha_2}[s := t])$. Consider $\alpha' \equiv \alpha \backslash \{u \approx v\}$. One can see that either $(u \approx v) \in \alpha_1$ or $(u \approx v) \in \alpha_2$. W.l.o.g. we can consider the case $(u \approx v) \in \alpha_1$. Let $\alpha'_1 \equiv \alpha_1 \backslash \{u \approx v\}$. By Lemma 26,

$$\mathsf{SubTerm}(\varphi_{\alpha'}) = \mathsf{SubTerm}(\varphi_{\alpha'_1}[s := t] \wedge (s \approx t) \wedge \varphi_{\alpha_2}[s := t]).$$

Taking into account Definition 15, we obtain that

$$(u \approx v) \notin \mathsf{NPEq}(\varphi_{\alpha_1}[s := t] \wedge (s \approx t) \wedge \varphi_{\alpha_2}[s := t]).$$

We can conclude that for each $(u \approx v) \in \alpha$ such that $(u \approx v) \notin \mathsf{NPEq}(\varphi_\alpha)$,

$$(u \approx v)[s := t] \notin \mathsf{NPEq}(\varphi_{\alpha_1}[s := t] \wedge (s \approx t) \wedge \varphi_{\alpha_2}[s := t]),$$

i.e. the size of the set $\mathsf{NPEq}(\varphi_{\alpha_1}[s := t] \wedge (s \approx t) \wedge \varphi_{\alpha_2}[s := t])$ cannot be bigger than the size of the set $\mathsf{NPEq}(\varphi_\alpha)$.
Since $(s \approx t) \in \mathsf{NPEq}(\varphi_\alpha)$ and $(s \approx t) \notin \mathsf{NPEq}(\varphi_{\alpha_1}[s := t] \wedge (s \approx t) \wedge \varphi_{\alpha_2}[s := t])$, we can conclude that

$$|\mathsf{NPEq}(\varphi_{\alpha_1}[s := t] \wedge (s \approx t) \wedge \varphi_{\alpha_2}[s := t])| < |\mathsf{NPEq}(\varphi_\alpha)|.$$

2. Taking into account Lemma 25 and a definition of an EUF-BDD, we obtain that there are $t_1, \ldots, t_k$, $k \geq 0$ such that

$$\alpha_1 \equiv \alpha.s \not\approx t_1.\ldots.s \not\approx t_k,$$

where $t_i \prec t$,$1 \leq i \leq k$, and $s \notin \mathsf{SubTerm}(\alpha' \cup \alpha_2)$.
We show that Definition 23 holds.
We denote $\overline{\alpha}$ $s \not\approx t_1.\ldots.s \not\approx t_k$, $\overline{\alpha}^* = s \not\approx t_1[s := t].\ldots.s \not\approx t_k[s := t] = t \not\approx t_1.\ldots.t \not\approx t_k$, and $A = \varphi_{\alpha_1} \wedge (s \approx t) \wedge \varphi_{\alpha_2}$, $\overline{A} = \varphi_{\alpha_1}[s := t] \wedge (s \approx t) \wedge \varphi_{\alpha_2}[s := t]$.

(a) We check Definition 23(1).
Consider an arbitrary $l \in \mathsf{Lit}(\overline{A})$. Then the following holds.

- $l \in \overline{\alpha}^*$. Then $l \equiv (t \not\approx t_i)$ for some $i \in \{1, \ldots, k\}$. We conclude that $e(l)$ is not of the shape $r \approx r$.
- $l \in \alpha[s := t] \wedge (s \approx t) \wedge \alpha_2[s := t]$. We have shown that

$$\alpha[s := t] \wedge (s \approx t) \wedge \alpha_2[s := t] \equiv \alpha \wedge (s \approx t) \wedge \alpha_2.$$

Hence, $e(l)$ is not of the shape $r \approx r$.

(b) We check Definition 23(2).
By the observation above for each $(u \approx v) \in \mathsf{Lit}(A)$,

$$(u \approx v)[s := t] \equiv (u \approx v).$$

Consider an arbitrary $(u \approx v) \in \mathsf{Eq}(\overline{A})$.
Then one of the following holds.

- $(u \approx v) \in \mathsf{Eq}((s \approx t) \wedge \varphi_{\alpha_2}[s := t])$.
Taking into account

$$(s \approx t) \wedge \varphi_{\alpha_2}[s := t] \equiv (s \approx t) \wedge \varphi_{\alpha_2}$$

we obtain that Definition 23(2) holds.
- $(u \approx v) \in \mathsf{Eq}(\varphi_{\alpha_1}[s := t])$. Then taking into account Lemma 25 and Lemma 26 we obtain that Definition 23(2) also holds.

□

Proof. Consider an arbitrary path α in an EUF-ROBDD. We have to prove that φ_α is satisfiable. We claim that there exists a formula $\varphi_\alpha^{red} \in \mathsf{Red}$, which is satisfiable iff φ_α is. But φ_α^{red} is satisfiable by Theorem 17, which finishes the proof. We now prove the claim by induction on $|\mathsf{NPEq}(\varphi_\alpha)|$.

If $|\mathsf{NPEq}(\varphi_\alpha)| = 0$, note that $r \not\approx r$ cannot occur in simplified paths, so $\varphi_\alpha \in \mathsf{Red}$, and we are finished.

Next, if $|\mathsf{NPEq}(\varphi_\alpha)| > 0$, then φ_α is of the form $\varphi_1 \wedge (s \approx t) \wedge \varphi_2$, where $s \approx t$ is not propagated. Define $\varphi' := \varphi_1[s := t] \wedge (s \approx t) \wedge \varphi_2[s := t]$. By Lemma 27(1), $|\mathsf{NPEq}(\varphi')| < |\mathsf{NPEq}(\varphi_\alpha)|$, and by Lemma 27(2), no occurrences of $r \not\approx r$ are introduced. Taking into account Lemma 27(2) and by induction hypothesis, we find $\varphi'_{red} \in \mathsf{Red}$, equivalent to φ'.

We now show that φ_α and φ' are logically equivalent. Suppose $[\![\varphi_\alpha]\!]_\mathcal{D} = \mathsf{true}$ for a structure $\mathcal{D}$. Hence, $[\![s]\!]_\mathcal{D} = [\![t]\!]_\mathcal{D}$. For each $l[s := t] \in \mathsf{Lit}(\varphi')$, taking into account that $[\![l]\!]_\mathcal{D} = \mathsf{true}$, we obtain that $[\![l[s := t]]\!]_\mathcal{D} = \mathsf{true}$. Hence, $[\![\varphi']\!]_\mathcal{D} = \mathsf{true}$. Assume that $[\![\varphi']\!]_\mathcal{D} = \mathsf{true}$ for a structure $\mathcal{D}$. Taking into account the shape of φ', we obtain $[\![s]\!]_\mathcal{D} = [\![t]\!]_\mathcal{D}$. Hence, taking into account that for each $l \in \varphi_\alpha$, $[\![l[s := t]]\!]_\mathcal{D} = \mathsf{true}$, we obtain that $[\![l]\!]_\mathcal{D} = \mathsf{true}$. Hence, $[\![\varphi_\alpha]\!]_\mathcal{D} = \mathsf{true}$. We conclude that φ_α and φ' are logically equivalent.

□