A proof system and a decision procedure for equality logic

A Proof System and a Decision Procedure for Equality Logic

Olga Tveretina and Hans Zantema

Department of Computer Science, TU Eindhoven, P.O. Box 513,
5600 MB Eindhoven, The Netherlands
o.tveretina@tue.nl, h.zantema@tue.nl

Abstract. We give an approach for deciding satisfiability of equality logic formulas (E-SAT) in conjunctive normal form. Central in our approach is a single proof rule called *equality resolution* (ER). For this single rule we prove soundness and completeness. Based on this rule we propose a complete procedure for E-SAT and prove its correctness. Applying our procedure on a variation of the pigeon hole formula yields a polynomial complexity contrary to earlier approaches to E-SAT. Parts of the theory we developed for proving completeness of the proof rule and the algorithm are of interest in itself: we give techniques for removing clauses preserving unsatisfiability, and we give a general theorem globalizing a local commutation criterion for different proof systems.

Keywords: equality logic, satisfiability, resolution.

1 Introduction

The logic of equality with uninterpreted functions (UIFs) has been proposed for verifying hardware [5]. This type of logic is mainly used for proving equivalence between systems. When verifying equivalence between two formulas it is often possible to abstract away functions replacing them with UIFs. In [1] Ackermann showed that the problem of deciding the validity of the formula in equality logic with UIFs can be reduced to checking satisfiability of equality formulas. The abstraction process does not preserve validity and may transform a valid formula into the invalid formula. However, in some application domains the process of abstraction is justified. Bryant et al. [3] presented a different approach to transform a formula containing functions into a equality logic formula which can exploit the maximal diversity property, while Ackermann method cannot.

An equality formula is a formula consisting of equalities between a given set of variables and usual propositional connectives. The meaning of such a formula is given by interpreting the variables in any domain and the equality has its usual meaning.

In the past several years various procedures for checking satisfiability of such formulas have been suggested. Barrett at al. [2] proposed a decision procedure based on computing congruence closure in combination with case splitting.

Many approaches use transformation of equality logic to propositional logic. Having such a transformation Ψ then checking satisfiability of an equality formula ϕ proceeds as follows: compute $\Psi(\phi)$ and apply a standard satisfiability checker for propositional formulas. Goel et al. [6] and Bryant et al. [4] reduce an equality formula to a propositional one by adding transitivity constraints. In this approach it is analyzed which transitivity properties may be relevant.

A different approach is called range allocation [8, 11]. In this approach a formula structure is analyzed to define a small domain for each variable. Then a standard BDD based tool is used to check satisfiability of the formula under the domain.

Another approach is given in [7]. This approach is based on BDD computation, with some extra rules for dealing with transitivity. Unfortunately unicity of reduced BDDs is lost, but for E-SAT a full argorithm is given.

The main problem dealt with in this paper is: given an equality formula, decide whether it is satisfiable or not. This problem is called E-SAT, similar to the way propositional satisfiability is called SAT. Similar to propositional logic every equality formula can be transformed to an equality formula in conjunctive normal form (E-CNF) such that the original formula is satisfiable if and only if the E-CNF is satisfiable. Hence we may, and shall, concentrate on satisfiability of E-CNFs.

In this paper we propose a resolution-based approach to deal with equality formulas. One of the problem that our method must deal with is detecting inconsistent combinations of atoms, such as $x_1 \approx x_2, \ldots, x_{n-1} \approx x_n, x_1 \not\approx x_n$. We give a single proof rule called equality resolution (ER):

$$\frac{\{x_1 \approx x_2\} \cup C_1, \ldots, \ \{x_{n-1} \approx x_n\} \cup C_{n-1}, \{x_1 \not\approx x_n\} \cup C_n}{C_1 \cup \cdots \cup C_n}$$

Here we write '$\approx$' for equality rather than '$=$' to avoid confusion with other applications of the symbol '$=$' and we use the notation $x \not\approx y$ as an abbreviation of $\neg(x \approx y)$. We consider clauses as sets of literals, hence using set theory notation for joining clauses.

It is well-known that by the combination of paramodulation and resolution the empty clause can be deduced from unsatisfiable set of clauses [9]. However, earlier resolution-based proof systems for logics with equality can either generate numerous useless resolvents or generate clauses containing new literals. Our system has a number of advantages, for instance by our proof system in the newly generated clause only literals occur that already occur in the original formula. We will prove that the ER rule is sound and complete using completeness of paramodulation together with resolution.

A decision procedure is an essential component of formal verification systems. Since checking satisfiability of equality formula is NP-hard no general efficient algorithm exists. Our procedure for checking satisfiability given E-CNF is a variant of the classical Davis-Putnam procedure for propositional logic . In order to prove soundness and completeness of our procedure we develop some theory that is of interest itself.

The search space in saturation-based procedures can grow very rapidly. The procedure is more efficient when redundant clauses are removed during search. We give criteria for redundancy and we optimize our procedure by using these criteria.

As an example we apply this procedure to a formula parameterized by n that is a variation of the well-known pigeon hole formula. It turns out that our procedure will prove unsatisfiability of this formula very efficiently, even quadratic in n, while standard approaches fail to efficiently prove unsatisfiability of this formula.

Our paper is organized as follows. In Section 2 we give basic definitions. In Section 3 we introduce ER rule and and we present a general theorem globalizing a local commutation criterion for different proof systems. We give approaches to remove redundant clauses from given E-CNF in Section 4. In Section 5 the basic and optimized procedures are described and some examples are given. Some concluding remarks are in Section 6.

2 Basic Definitions and Preliminaries

Any formula in equality logic can be straightforwardly converted to an equivalent E-CNF just like this can be done in propositional logic. In worst case the size of the result is exponential in the size of the original formula. This can be avoided by adding extra variables. The well-known Tseitin transformation transforms an arbitrary propositional formula to a CNF in such a way that the original formula is satisfiable if and only if the CNF is satisfiable. Both the size of the resulting CNF and the complexity of the transformation procedure is linear in the size of the original formula. In this transformation new propositional are introduced, so applying it directly to equality formulas will yield a CNF in which the atoms are both equalities and propositional variables. However, if we have n propositional variables $p_1, \ldots, p_n$ we can introduce $n+1$ fresh domain variables $a, x_1, \ldots x_n$ and replace every propositional variable p_i by the equality $x_i \approx a$. In this way satisfiability is easily seen to be maintained. Hence we may and shall restrict to satisfiability of E-CNFs.

An E-CNF F is a conjunction of clauses. A *clause* C is a disjunction of literals. The *empty clause* is denoted by $\bot$. A *literal* l is a an *atom* $x \approx y$ or a *negated atom* $x \not\approx y$, where x and y belong to a set of *variables* V. We consider $x \approx y$ and $y \approx x$ as the same atom. Since conjunction and disjunction are associative and commutative an E-CNF can be viewed as a set of literals sets.

A *domain* D is defined to be a non-empty set. For any domain we define an *assignment* as a function $A : V \to D$. For an assignment A we define the corresponding *interpretation* I_A on literals by:

$I_A(x \approx y) = \mathsf{true}$ if $A(x) = A(y)$
$I_A(x \approx y) = \mathsf{false}$ if $A(x) \neq A(y)$
$I_A(x \not\approx y) = \neg I_A(x \approx y)$

An assignment A satisfies a literal l on some domain D if $I_A(l) = \mathsf{true}$. An assignment satisfies a clause if an interpretation of at least one of its literals

yields true. An assignment satisfies an E-CNF F if and only if it satisfies each of its clauses and we say that F is mapped to true. An E-CNF F on a set of variables S is called *satisfiable* if there is a domain D and an assignment A such that F is mapped to true. If there is no assignment that maps an E-CNF F to true then F is called *unsatisfiable*. For an E-CNF F we write V_F for the set of all variables contained in F and L_F for the set of all literals contained in F.

Definition 1. *A set of clauses is called minimally unsatisfiable if it is unsatisfiable and each of its subsets is satisfiable.*

Lemma 2. *The set of clauses* $\{\{x_1 \approx x_2\}, \ldots, \{x_{n-1} \approx x_n\}, \{x_1 \not\approx x_n\}\}$ *is minimally unsatisfiable.*

Proof. Obviously, this set of clauses is unsatisfiable.

We shall show that removing any clause does this set of clause satisfiable. Any subset of satisfiable set of clauses is also satisfiable.

There exists an assignment $A : V \to \boldsymbol{N}$ such that $A(x_1) = \cdots = A(x_n)$. The assignment satisfies $\{x_1 \approx x_2\}, \ldots, \{x_{n-1} \approx x_n\}$. For $i \in \{1, \ldots, n\}$ there exists an assignment $A : V \to \boldsymbol{N}$ such that $A(x_1) = \cdots = A(x_i) \neq A(x_{i+1}) = \cdots = A(x_n)$ satisfying $\{x_1 \approx x_2\}, \ldots, \{x_{i-1} \approx x_i\}, \{x_{i+1} \approx x_{i+1}\}, \ldots, \{x_{n-1} \approx x_n\}, \{x_1 \not\approx x_n\}$. □

An important notion in this paper is *contradictory cycle*.

Definition 3. *A contradictory cycle* θ *is defined to be a set of literals* $x_1 \approx x_2, \ldots, x_{n-1} \approx x_n, x_1 \not\approx x_n$*, where* $x_1, \ldots, x_n$ *are distinct variables.*

A contradictory cycle θ *is called a contradictory cycle of an E-CNF* F *if all literals in* θ *are contained in* L_F*.*

When drawing a graph consisting of the variables from an E-CNF F as nodes, and equalities from L_F as solid edges and inequalities from L_F as dashed edges, then a contradictory cycle of F corresponds exactly to a cycle in this graph in which one edge is dashed and all other edges are solid. For a given E-CNF such a graph is easily made and such cycles are easily established by looking for solid paths from the one end of a dashed edge to the other end. In Theorems 13 and 15 we will see that unsatisfiability of an E-CNF is preserved by removing clauses containing literals that are not on a contradictory cycle.

3 Proof Systems

In this section we present the proof systems that play a role in this paper, and we derive a fruitful commutation theorem.

3.1 ER Rule

It is well-known that in first order logic the combination of resolution and paramodulation is complete, see [9]. This means that a formula is unsatisfiable

if and only if the empty clause can be derived. In our particular case of equality logic paramodulation boils down to the following transitivity rule.

Transitivity rule:

$$\frac{\{x \approx y\} \cup C, \{y \approx z\} \cup D}{\{x \approx z\} \cup C \cup D}$$

For equality logic the resolution rule can be presented as following.

Resolution rule:

$$\frac{\{x \approx y\} \cup C, \{x \not\approx y\} \cup D}{C \cup D}$$

For $F' = F \cup \{C\}$ we shall use the notation $F \rightarrow_t F'$ if C was derived from F by the transitivity rule and $F \rightarrow_r F'$ if C was derived by the resolution rule.

Completeness of the combination of paramodulation and resolution now implies that an E-CNF is unsatisfiable if and only if the empty clause can be derived by only using the transitivity rule and the resolution rule. Instead of these two rules we now introduce one single rule, the equality resolution rule (ER), that is complete in its own, and has a number of advantages.

ER rule:

$$\frac{\{x_1 \approx x_2\} \cup C_1, \ldots, \{x_{n-1} \approx x_n\} \cup C_{n-1}, \{x_1 \not\approx x_n\} \cup C_n}{C_1 \cup \cdots \cup C_n}$$

Clearly the contradictory cycle $\theta = \{x_1 \approx x_2, \ldots, x_{n-1} \approx x_n, x_1 \not\approx x_n\}$ is closely related to this rule; in fact for every contradictory cycle we have corresponding instance of the ER rule.

If a new clause C is derived from an E-CNF F by the ER rule using the contradictory cycle θ then we write $F \rightarrow_{er}^{\theta} F'$ for $F' = F \cup \{C\}$. In case the involved contradictory cycle θ is not relevant we shortly write $F \rightarrow_{er} F_e$. We write $F \rightarrow_\theta F_e$ if ER rule is applied for a fixed contradictory cycle θ.

3.2 Commutation of Proof Systems

In order to prove completeness of the ER rule based on completeness of the combination of resolution and transitivity we will need a commutation property between the proof systems of resolution and transitivity. Later on in proving correctness of our procedure we will need a quite similar commutation property in another setting. Therefore we now develop the desired commutation results for arbitrary proof systems.

Here a proof system may be anything by which new statements, e.g. clauses, may be deduced from existing statements. For such a proof system s we use the notation $F \rightarrow_s G$ for $G = F \cup \{C\}$, where C is a statement deduced from F by the proof system s.

For any relation $\to$ we write $\to^*$ for its reflexive transitive closure, i.e., we write $F \to^* G$ if $F_0, \ldots, F_n$ exist for $n \geq 0$ satisfying

$$F = F_0 \to F_1 \to \cdots \to F_n = G.$$

Definition 4. *We write $F \sqsubseteq G$ if for any $C \in G$ there is $D \in F$ such that $D \subseteq C$.*

A proof system is $\sqsubseteq$-monotonic if from $F \to_s^* F'$ follows that for any $G \sqsubseteq F$ there is G' such that $G \to_s^* G'$ and $G' \sqsubseteq F'$.

Definition 5. *Let s_1 and s_2 are proof systems and $F \to_{s_1} F' \to_{s_2} F''$. We say that a proof system s_1 commutes over a proof system s_2 if for some finite n there exist $G_1, \ldots, G_n, G$ such that*

1. *$F \to_{s_2} G_i$ for any $i \in \{1, \ldots, n\}$.*
2. *$\bigcup_{i=1}^{n} G_i \to_{s_1}^* G$, where $G \sqsubseteq F''$.*

Lemma 6. *Let s_1 and s_2 are $\sqsubseteq$-monotonic proof systems such that s_1 commutes over s_2, and $F \to_{s_1}^* F' \to_{s_2} F''$. Then there are G', G'' such that $F \to_{s_2}^* G' \to_{s_1}^* G''$, where $G \sqsubseteq F''$.*

Proof. Let $F = F_0 \to_{s_1} F_1 \to_{s_1} \cdots \to_{s_1} F_n = F'$. The proof by induction on n.

Base case. $n = 0$. The lemma trivially holds.

Inductive step. Let the lemma hold for $n-1$. By definition there are $G_1, \ldots, G_m$ for some finite m such that

1. $F_{n-1} \to_{s_2} G_i$ for any $i \in \{1, \ldots, m\}$
2. $\bigcup_{i=1}^{m} G_i \to_{s_1} G$, where $G \sqsubseteq F''$.

By induction hypothesis for any $i \in \{1, \ldots, m\}$ there are $G'_1, \ldots, G'_m, G''_1, \ldots, G''_m$ such that $F \to_{s_2}^* G'_i \to_{s_1}^* G''_i$, where $G''_i \sqsubseteq G_i$.

As $G''_i \sqsubseteq G_i$ then by Definition 4 there is G'' such that $\bigcup_{i=1}^{m} G''_i \to_{s_1}^* G''$, where $G'' \sqsubseteq G$.

We choose $G' = \bigcup_{i=1}^{m} G'_i$. Then $F \to_{s_2}^* G' \to_{s_1}^* G''$, where $G'' \sqsubseteq F''$. □

Now we are ready to prove the theorem stating that any derivation consisting of any mix of s_1-steps and s_2-steps, can be rearranged in such a way that first only s_2-steps are applied and then only s_1-steps.

Theorem 7. *Let s_1 and s_2 are $\sqsubseteq$-monotonic proof systems such that s_1 commutes over s_2. Let s be the union of s_1 and s_2.*

If $F \to_s^* F'$ then there are G, G' such that $F \to_{s_2}^* G \to_{s_1}^* G'$, where $G' \sqsubseteq F'$.

Proof. Let $F = F_0 \rightarrow_s F_1 \rightarrow_s \cdots \rightarrow_s F_n = F'$. The proof by induction on n.

Base case. $n = 0$. The theorem trivially holds.

Inductive step. Let the theorem hold for $n-1$. Then by induction hypothesis there are F_1', F_2' such that $F \rightarrow^*_{s_2} F_1' \rightarrow^*_{s_1} F_2'$, where $F_2' \sqsubseteq F_{n-1}$.

If $F_{n-1} \rightarrow_{s_1} F_n$ then the theorem holds by Definition 4. If $F_{n-1} \rightarrow_{s_2} F_n$ then the theorem holds by Lemma 6. □

The next theorem is a further generalization of Theorem 7: now not only two proof systems are involved but an arbitrary number.

Theorem 8. *Let s be the union of $\sqsubseteq$-monotonic proof systems $s_1, \ldots, s_n$. Let for any $i > j$ s_i commutes over s_j.*

If $F \rightarrow^*_s F'$ then there are $F_1, \ldots, F_n$ such that $F \rightarrow^*_{s_1} F_1 \rightarrow^*_{s_2} \cdots \rightarrow^*_{s_n} F_n$, where $F_n \sqsubseteq F'$.

Proof. The proof by induction on n.

Base case. $k = 1$. Trivially holds.

Inductive step. Let the theorem hold for $n-1$. Let s' be the union of systems $s_2, \ldots, s_n$. By Theorem 7 there are F_1, F_1' such that $F \rightarrow^*_{s_1} F_1 \rightarrow^*_{s'} F_1'$, $F_1' \sqsubseteq F'$. By induction hypothesis there are $F_2, \ldots, F_n$ such that $F_1 \rightarrow^*_{s_2} \cdots \rightarrow^*_{s_n} F_n$, where $F_n \sqsubseteq F_1'$. Then $F_1, \ldots, F_n$ satisfy the theorem. □

3.3 Soundness and Completeness of ER

Theorem 9. *(Soundness of ER rule) Let $F \rightarrow_{er} F_e$. Then F is satisfiable iff F_e is satisfiable.*

Proof. Let $F_e = F \cup \{C_1 \cup \cdots \cup C_n\}$, where $C_1 \cup \{x_1 \approx x_2\}, \ldots, C_{n-1} \cup \{x_{n-1} \approx x_n\}, C_n \cup \{x_1 \not\approx x_n\} \in F$.

Let F_e be satisfiable. Then F is satisfiable as a subset of satisfiable set of clauses.

Let F be satisfiable. The set $\{\{x_1 \approx x_2\}, \ldots, \{x_{n-1} \approx x_n\}, \{x_1 \not\approx x_n\}\}$ is minimally unsatisfiable by Lemma 2. Then an assignment satisfying F satisfies C_i for some $i \in \{1, \ldots, n\}$. The same assignment satisfies $C_1 \cup \cdots \cup C_n$ and F_e. □

In order to prove completeness we will use a commutation property and Theorem 7.

Lemma 10. *The transitivity rule commutes over the ER rule.*

Proof. It is easily observed that transitivity and ER rules are $\sqsubseteq$-monotonic.

We have to prove that transitivity and ER proof systems satisfy Definition 5.

Let $F \rightarrow_t F' \rightarrow_{er} F''$, where $F' = F \cup \{C\}$, $F'' = F' \cup \{D\}$.

Let $C = C_1 \cup C_2 \cup \{x \approx z\}$, where $C_1 \cup \{x \approx y\}, C_2 \cup \{y \approx z\} \in F$; $D = \bigcup_{i=1}^{n} D_i$, where $D_i \cup \{l_i\} \in F'$ for any $i \in \{1, \ldots, n\}$, $\{l_1, \ldots, l_n\}$ be a contradictory cycle.

If $C \neq D_i$ for any $i \in \{1, \ldots, n\}$ then let us choose $G_1 = F \cup \{D\}$, $G = G_1 \cup \{C\}$. In this case $G = F''$.

Let $C = D_i$ for some $i \in \{1, \ldots, n\}$.

W.l.o.g. we can assume that $i = 1$. Then one of the following holds.

1. l_1 is equal to $x \approx z$.

   We have $D = C_1 \cup C_2 \cup \bigcup_{i=2}^{n} D_i$.

   Literals $x = y, y = z, l_2, \ldots, l_n$ form a contradictory cycle. Then D can be derived from F by ER rule. Let $G_1 = F \cup \{D\}$, $G = G_1 \cup \{C\}$. In this case $G = F''$.
2. $l_1 \in C_1 \cup C_2$.

   Let $C_1' = C_1 \cup \{x \approx y\}$, $C_2' = C_2 \cup \{y \approx z\}$, $D_j = C_j' \backslash \{l_1\} \cup \bigcup_{i=2}^{n} D_i$.

   If $l_1 \in C_1$ or $l_1 \in C_2$ then D_1 or D_2 can be derived from F by ER rule. Assume that $G_i = F \cup \{D_i\}$ for $i \in \{1, 2\}$.

   If a literal l_1 is not equal neither to $\neq x \approx y$ nor to $y \approx z$ then D can be derived from D_1 and D_2 by transitivity rule. If a literal l_1 is equal to $x \approx y$ then $D_1 \subseteq D$. If a literal l_1 is equal to $y \approx z$ then $D_2 \subseteq D$.

   If $l_1 \in C_1$ and $l_1 \in C_2$ then there are G_1, G_2, G satisfying Definition 5. If $l_1 \in C_1$ or $l_1 \in C_1$ then there are G_1, G or G_2, G satisfying Definition 5.

In all cases we are done. □

Theorem 11. *(Completeness of ER rule) An E-CNF is unsatisfiable iff the empty clause can be derived by ER rule.*

Proof. Assume that the empty clause can be derived from the E-CNF by ER. Then by Theorem 9 the original set of clauses is unsatisfiable.

Assume that the original set F of clauses is unsatisfiable. Then according to the well-known paramodulation result by transitivity and resolution the empty clause can be derived from it. Since resolution is a particular case of the ER rule we conclude that $F \rightarrow_s F'$ for some F' containing $\bot$, where s is the combination of transitivity and the ER rule.

Let $F \rightarrow_s F'$, where $\bot \in F'$. Then by Lemma 10 and Theorem 7 there are G, G' such that $F \rightarrow_{er}^* G \rightarrow_t^* G'$, $G' \sqsubseteq F'$. Since $\bot \in F'$ then $\bot \in G'$. By the transitivity rule $\bot$ cannot be derived so $\bot \in G$. □

4 Removing Redundant Clauses

Given an E-CNF F. For a literal $l \in L_F$ let $F|l$ denote the E-CNF obtained from F by deleting all clauses that contain l.

In the following we will use the fact that F is satisfiable if and only if it is satisfiable on the set of natural numbers $\mathbf{N}$. So w.l.o.g. we may and shall assume that $D = \mathbf{N}$.

4.1 Removing disequalities

Definition 12. *Let F be an E-CNF . The relation $=_F$ on V_F is defined by $x =_F y$ if and only if $x \approx y \in L_F$.*

The relation $\sim_F$ is defined to be the equivalence relation generated by $=_F$, i.e., the reflexive, symmetric, transitive closure of the relation $=_F$. By E_F^x we denote the equivalence class of x with respect to $\sim_F$.

Theorem 13. *Let $x \nsim_F y$. Then F is satisfiable iff $F|x \not\approx y$ is satisfiable.*

Proof. Let F is satisfiable then $F|x \not\approx y$ is satisfiable as a subset of satisfiable set of clauses.

Since $F|x \not\approx y$ is e-satisfiable there is a satisfying assignment $A : V_F \to \mathbf{N}$ for $F|x \not\approx y$. We define a new assignment A' in such a way that it preserves satisfiability of $F|x \not\approx y$ but also satisfies $x \not\approx y$. Choose number N satisfying $N > A(x') - A(y')$ for all $x', y' \in V_F$. We define $A' : V_F \to \mathbf{N}$ as follows.

$$A'(z) = \begin{cases} A(z) + N, z \in E_F^x \\ A(z), z \in V_F \backslash E_F^x \end{cases}$$

Since A is a satisfying assignment for $F|x \not\approx y$ then for any $C \in F|x \not\approx y$ there is $l \in C$ such that $I_A(l) = true$. We check that $I_{A'}(l) = true$.

1. A literal l is equal to $x' \approx y'$ for some $x'', y'' \in V_F$.
   If $x' \not\sim_F x$ then $A'(x'') = A(x'') = A(y'') = A'(y'')$. If $x'' \sim_F x$ than $A'(x'') = A(x'') + N = A(y'') + N = A'(y'')$.
   In both cases $I_{A'}(l) = true$.
2. A literal l is equal to $x'' \not\approx y''$ for some $x'', y'' \in V_F$.
   (a) $x'' \sim_F y''$.
   If $x'' \sim_F x$ then $A'(x'') = A(x'') + N \neq A(y'') + N = A'(y'')$. If $x'' \not\sim_F x$ then $A'(x'') = A(x'') \neq A(y'') = A'(y'')$.
   (b) $x'' \not\sim_F y''$.
   Using that $N > A(x') - A(y')$ for all $x', y' \in V_F$ we have:
   If $x'' \sim_F x$ and $y'' \not\sim_F x$ then $A'(x'') = A(x'') + N > A(y'') = A'(y'')$. If $x'' \not\sim_F x$ and $y'' \sim_F x$ then $A'(x'') = A(x'') < A(y'') + N = A'(y'')$. If $x'' \not\sim_F x$ and $y'' \not\sim_F x$ then $A'(x'') = A(x'') \neq A(y'') = A'(y'')$.
   In all cases $I_{A'}(l) = true$.

Since $N > A(x') - A(y')$ for all $x', y' \in V_F$ and $y \not\sim_F x$ we have $A'(x) = A(x) + N > A(y) = A'(y)$. So $I_{A'}(x \neq y) = true$. □

Example 14. As an example we have taken the formula from [8] raised during the process of translation validation. After abstracting concrete functions, performing the Ackermann reduction the following E-CNF is obtained:

$$F_1 = (x_1 \not\approx x_2 \vee x_3 \not\approx x_4 \vee y_1 \approx y_2) \wedge (y_1 \not\approx y_3 \vee y_2 \not\approx y_4 \vee z_1 \approx z_2) \wedge$$
$$y_1 \approx y_3 \wedge y_2 \approx y_4 \wedge z_1 \approx z_3 \wedge z_2 \not\approx z_3.$$

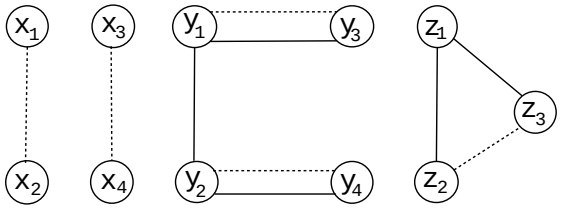

Fig. 1. The graph corresponding to F_1

As sketched before we may draw a graph consisting of the variables from F_1 as nodes, and equalities from L_{F_1} as solid edges and inequalities from L_{F_1} as dashed edges. The result is given in Figure 1.

In this graph notation $x \sim y$ means that there is a path from node x to node y purely consisting of solid edges. Hence in this example we clearly have $x_1 \nsim x_2$. Hence by Theorem 13 we may remove all clauses containing the literal $x_1 \not\approx x_2$ without changing satisfiability behavior. Hence we may remove the first clause, resulting in

$$F_2 = (y_1 \not\approx y_3 \vee y_2 \not\approx y_4 \vee z_1 \approx z_2) \wedge y_1 \approx y_3 \wedge y_2 \approx y_4 \wedge z_1 \approx z_3 \wedge z_2 \not\approx z_3$$

We see that variables x_1, x_2, x_3 and x_4 do not occur in F_2, and the resulting graph for F_2 is given in Figure 2.

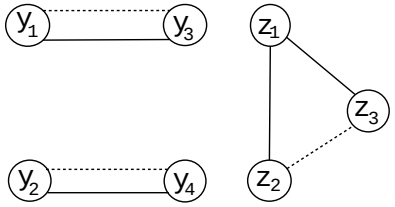

Fig. 2. The graph corresponding to F_2

4.2 Removing Equalities

Theorem 15. *Let F be an E-CNF and let $x \approx y \in L_F$ not be contained in any contradictory cycle of F. Then F is satisfiable iff $F|x \approx y$ is satisfiable.*

Proof. Let F is satisfiable then $F|x \approx y$ is satisfiable as a subset of a satisfiable set of clauses.

Conversely assume that $F|x \approx y$ is satisfiable. Then there is a satisfying assignment $A : V_F \to \mathbf{N}$ for $F|x \approx y$. We will show that there is a satisfying assignment for F.

We define the set P_x as follows:

- $x \in P_x$.
- $z \in P_x$ if $z \approx z' \in L_F$ and $A(z) = A(z')$ for some $z' \in P_x$.

Let $N > \max\limits_{x \in V_F} A(x)$. Then we may define a new assignment as follows:

$$A'(z) = \begin{cases} N, z \in P_x \cup P_y \\ A(z), z \in V_F \backslash (P_x \cup P_y) \end{cases}$$

As A is a satisfying assignment for $F|x \approx y$ then for any $C \in F|x \approx y$ there exists $l \in C$ such that $I_A(l) = true$. We will show that $I_{A'}(l) = true$.

Let us take an arbitrary clause C from $F|x \approx y$. Then one of the following holds:

1. A literal l is equal to $x' \approx y'$.
   Then either
   $x', y' \in P_x \cup P_y$ and $A'(x') = N = A'(y')$
   or
   $x', y' \in V_F \backslash (P_x \cup P_y)$ and $A'(x') = A(x') = A(y') = A'(y')$.
2. A literal l is equal to $x' \not\approx y'$.
   We consider the case when $x', y' \in P_x \cup P_y$.
   If $x' \in P_x$ and $y' \in P_y$ or vise versa then $x \approx y$ is contained in a contradictory cycle of F, contradicting the assumption of the theorem.
   If $x', y' \in P_x$ then $A(x') = A(y')$ and $I_A(l) \neq true$. In this case we also have a contradiction. By symmetry in case when $x', y' \in P_y$ we have a contradiction.
   Then one of the following holds:
   (a) $x' \notin P_x \cup P_y$, $y' \notin P_x \cup P_y$. Then $A'(x') = A(x') \neq A(y') = A'(y')$.
   (b) $x' \in P_x \cup P_y$, $y' \notin P_x \cup P_y$. Then $A'(x') = N > A(y') = A'(y')$ as $N > \max\limits_{x \in V_F} A(x)$.
   (c) $x' \notin P_x \cup P_y$, $y' \in P_x \cup P_y$. Then $A'(y') = N > A(x') = A'(x')$ as $N > \max\limits_{x \in V_F} A(x)$.
   In all cases $I_{A'}(l) = true$.

We take an arbitrary C such that $x \approx y \in C$. By definition of A', $A'(x) = A'(y)$. Then $I_{A'}(x = y) = true$.

The assignment A' preserves satisfiability of $F|x \approx y$ and satisfies clauses containing $x \approx y$. □

In the graph representation Theorem 15 states that every clause may be removed containing an equality corresponding to a solid edge for which no path between the end points of the edge exists containing exactly one dashed edge.

Example 16. Consider the formula F_1 from Example 14. By applying Theorem 15 we may remove all clauses containing the equality $y_1 \approx y_2$. As a result we again obtain F_2, of which the graph is given in Figure 2.

The fact that by applying Theorem 13 and Theorem 15 in this particular example the same clause is removed, is a coincidence; in more complicated examples one sees that the combination of both theorems is more powerful than applying only one of them.

5 The E-SAT Procedure

5.1 The Basic E-SAT Procedure

We shall describe the basic E-SAT procedure for the purpose proving the approach completeness. Based on this we shall present in the following subsection a modified version which is more efficient.

Given a nonempty E-CNF containing nonempty clauses the E-SAT procedure forms the set of all contradictory cycles Θ and then repeats the following steps.

- Choose a contradictory cycle $\theta \in \Theta$ and remove θ from Θ.
- Add all possible clauses derived from V by ER rule over θ.

We give a precise version of the procedure.

Procedure E-SAT(F);
```
begin
   Θ := ContrCycle(F);
   while (Θ ≠ ∅) do
   begin
      choose θ ∈ Θ;
      Θ := Θ\{θ};
      if ⊥ ∈ F return(unsatisfiable);
      F := F∪ ER(F, θ);
   end
   return(satisfiable);
end
```

Fig. 3. The basic E-SAT procedure

In this procedure the function $\texttt{ContrCycle}(F)$ forms the set of all possible contradictory cycles. The function $\texttt{ER}(F, \theta)$ forms the set of clauses derived from F by all possible θ-steps.

The procedure ends when either the empty clause derived or no contradictory cycles left. If the empty clause is derived the output the procedure "satisfiable". If the empty clause is not derived during the procedure the output is "satisfiable".

5.2 Soundness and Completeness of the Procedure

We shall prove the completeness of the basic procedure. Let $\theta_1, \ldots, \theta_n$ be contradictory cycles of an unsatisfiable E-CNF F_0. Based on the completeness of ER rule we shall show that there is a finite sequence $F_1, \ldots, F_n$ such that for any $i \in \{1, \ldots, n\}$ F_i consists of all clauses contained in F_{i-1} and clauses derived from F_{i-1} in one θ_i-step, and F_n contains the empty clause.

Lemma 17. *Let for $i \in \{1, 2\}$ θ_i-step be a proof system s_i. Then s_1 commutes over s_2.*

Proof. We shall prove that s_1 and s_2 satisfy Definition 5. Let $F \rightarrow_{\theta_1} F_1 \rightarrow_{\theta_2} F_2$.

Let $F_1 = F \cup \{C\}$, $F_2 = F_1 \cup \{D\}, D = \bigcup_{i=1}^{m} D_i \backslash \{l_i\}$ for some $D_1, \ldots, D_n \in F_1$.

Then one of the following holds.

1. $C \neq D_i$ for any $i \in \{1, \ldots, m\}$. We choose $G_1 = F \cup \{D\}, G = G_1 \cup \{C\}$. The lemma holds.
2. $C = D_i$ for some $i \in \{1, \ldots, m\}$. W.l.o.g. we can assume that $i = 1$.
   Let $C = \bigcup_{i=1}^{r} C_i \backslash \{l_i'\}$ for some $C_1, \ldots, C_r \in F$.
   W.l.o.g we can assume that $l_1 \in \{C_{i_1}$ for some $i \in \{1, \ldots, k\}$.
   We define
   (a) For any $i \in \{1, \ldots, k\}$ $G_i = F \cup \{C_i^*\}$, where $C_i^* = C_i \backslash \{l_1\} \cup \bigcup_{j=2}^{m} D_j \backslash \{l_j\}$.
   (b) $G = \bigcup_{i=1}^{k} G_i \cup \{C\} \cup \{D^*\}$, where $D^* = \bigcup_{i=1}^{k} C_i^* \backslash \{l_i'\} \cup \bigcup_{i=k+1}^{r} C_i \backslash \{l_i'\}$.

   We shall show that $D^* \subseteq D$.

$$
\begin{aligned}
D^* &= \bigcup_{i=1}^{k} C_i^* \backslash \{l_i'\} \cup \bigcup_{i=k+1}^{r} C_i \backslash \{l_i'\} \\
&= \bigcup_{i=1}^{k} (C_i \backslash \{l_1\} \cup \bigcup_{j=2}^{m} D_j \backslash \{l_j\}) \backslash \{l_i'\} \cup \bigcup_{i=k+1}^{r} C_i \backslash \{l_i'\} \\
&= \bigcup_{i=1}^{k} (C_i \backslash \{l_i'\}) \backslash \{l_1\} \cup \bigcup_{i=k+1}^{r} C_i \backslash \{l_i'\} \cup \bigcup_{j=2}^{m} D_j \backslash \{l_j\} \\
&= C \backslash \{l_1\} \cup \bigcup_{j=2}^{m} D_j \backslash \{l_j\} = \bigcup_{j=1}^{m} D_j \backslash \{l_j\} \subseteq D.
\end{aligned}
$$

Then for any $i \in \{1, \ldots, k\}$ G_i consists of clauses from F and a clause derived from F by θ_2-step, C and D^* can be derived from $\bigcup_{i=1}^{k} G_i$ by θ_1-step. It can be easily checked that $G \sqsubseteq F_2$. □

Theorem 18. *Let $\{\theta_1, \ldots, \theta_n\}$ be the set of all contradictory cycles in F. Let $F \rightarrow_{er}^* G$. Then $F \rightarrow_{\theta_1}^* F_1 \rightarrow_{\theta_2}^* \cdots \rightarrow_{\theta_m}^* F_m$, where $F_m \sqsubseteq G$ for some $F_1, \ldots, F_m$.*

Proof. The theorem follows from Theorem 8. □

Theorem 19. *Let $\{\theta_1, \ldots, \theta_n\}$ be the set of all contradictory cycles in F. Let $F \rightarrow_{er}^* G$. Then $F \rightarrow_{\theta_1}^* F_1 \rightarrow_{\theta_2}^* \cdots \rightarrow_{\theta_m}^* F_m$, where $F_m \sqsubseteq G$ for some $F_1, \ldots, F_m$.*

Proof. The theorem follows from Theorem 8. □

Theorem 20. *Let F and G be an E-CNFs, θ be a contradictory cycle, and $G = F \cup ER(F, \theta)$. If $G \rightarrow_\theta G'$ then $G \sqsubseteq G'$.*

Proof. If any $C \in G \backslash F$ does not contain a literal from θ then the theorem trivially holds. We shall show that for any $C \cup \{l\} \in G \backslash F$, where $l \in \theta$ there is $D \cup \{l\} \in F$ such that $D \subseteq C$.

Let $\theta = \{l_1, \ldots, l_n\}$; $C \cup \{l_i\} \in G \backslash F$ for some $i \in \{1, \ldots, n\}$; $C \cup \{l_i\} = D_1 \cup \cdots \cup D_n$, where $D_1 \cup \{l_1\}, \ldots, D_n \cup \{l_n\} \in F$. Then $D_i \subseteq C$.

Let $G' = G \cup \{C\}$. We shall show that there is $D \in G$ such that $D \subseteq C$.

Let $C = C_1 \cup \cdots \cup C_n$, where $C_1 \cup \{l_1\}, \ldots, C_n \cup \{l_n\} \in G$; $C_1 \cup \{l_1\}, \ldots, C_r \cup \{l_r\} \in G \backslash F$ for some $1 \leq r \leq n$. As it was shown above for any $i \in \{1, \ldots, r\}$ there is $D_i \cup \{l_i\} \in F$ such that $D_i \subseteq C_i$. Then $D = D_1 \cup \cdots \cup D_r \cup C_{r+1} \cup \cdots \cup C_n$ can be derived by θ-step, $D \subseteq C$ and $D_1, \ldots, D_r, C_{r+1}, \ldots, C_n \in F$. So $D \in G$. □

Theorem 21. *(Soundness and completeness of the basic E-SAT procedure) Let F be an E-CNF. Then F is unsatisfiable iff the output of the basic procedure is the empty clause.*

Proof. If there is a derivation of the empty clause by ER rule then F is unsatisfiable by Theorem 9.

If F is unsatisfiable then by Theorem 11 there is a derivation of the empty clause from F by ER rule.

Let $\{\theta_1, \ldots, \theta_n\}$ be the set of all contradictory cycles in F. Then by Theorem 19 there are $F_1, \ldots, F_m$ such that $F \rightarrow_{\theta_1}^* F_1 \rightarrow_{\theta_2}^* \cdots \rightarrow_{\theta_m}^* F_m$ and $\bot \in F_m$. By Theorem 20 $F_i = F_{i-1} \cup ER(F_{i-1}, \theta_i)$ for any $i \in \{1, \ldots, m\}$. It implies that the empty clause can be derived by the E-SAT procedure. □

5.3 The Optimized Procedure

The search space of the saturation-based procedures can grow very rapidly. The procedure becomes more efficient when we have criteria to remove redundant clauses from the search space. In the optimized procedure we use *subsumption* introduced by Robinson [10] for general resolution. Additional criteria to remove redundant clauses we obtain by means of the theorems proved in section 4.

The potential source of inefficiency is exponential in the size of a formula number of contradictory cycles. To avoid this problem an optimized procedure does not collect as first step the set of all contradictory cycles.

The optimized procedure repeats the following steps.

- Choose a contradictory cycle θ of the shortest length not contained in Θ and add it to Θ.
- Remove redundant clauses.
- Add all possible clauses derived from V by ER rule over θ.
- Remove clauses containing literals which are not in contradictory cycles not contained in Θ.

Procedure E-SAT(F);
begin
$\Theta := \emptyset$;
`while` $(F \neq \emptyset)$ `do`
`begin`
`RemoveRedundant` (F);
$\theta :=$ `ShortestContrCycle` (F, Θ);
$\Theta := \Theta \cup \{\theta\}$;
`if` $\perp \in F$ `return(unsatisfiable)`;
$F := F \cup$ `ER`(F);
`end`
`return(satisfiable)`;
`end`

Fig. 4. The optimized E-SAT procedure

The procedure `ShortestContrCycle` (F, Θ) chooses a contradictory cycle of the shortest length not contained in Θ. The procedure `RemoveRedundant`(F) repeatedly removes clauses from F by the following rules:

- if a clause C is a subclause of a clause C' then C' is removed (subsumption);
- remove a clause containing $x \not\approx y$ for which $x \nsim y$, see Theorem 13;
- remove a clause containing $x \approx y$ for which $x \approx y$ is not contained in any contradictory cycle, see Theorem 15;

until nothing can be removed any more. The function `ER`(F, θ) forms the set of clauses derived from F by all possible θ-steps.

The procedure ends if either the set of clauses is empty or the empty clauses is derived. If the empty clause is derived then the output is "unsatisfiable". If the set of clauses is empty then the output is "satisfiable".

Theorem 22. *(Soundness and completeness of the optimized E-SAT procedure) Let F be an E-CNF. Then F is unsatisfiable iff the output of the E-SAT procedure is the empty clause.*

Proof. If there is a derivation of the empty clause by ER rule then F is unsatisfiable by Theorem 9. If F is unsatisfiable then the empty clause can be derived by optimized procedure by Theorem 13, Theorem 15, and Theorem 21. □

Example 23. Consider the formula F_1 from Example 14. Removing redundant clauses yields the E-CNF F_2.

$$F_2 = (y_1 \not\approx y_3 \vee y_2 \not\approx y_4 \vee z_1 \approx z_2) \wedge y_1 \approx y_3 \wedge y_2 \approx y_4 \wedge z_1 \approx z_3 \wedge z_2 \not\approx z_3$$

The contradictory cycles contained in F_2 are following: $\theta_1 = \{y_1 \approx y_3, y_1 \not\approx y_3\}$, $\theta_2 = \{y_2 \approx y_4, y_2 \not\approx y_4\}$, $\theta_3 = \{z_1 \approx z_2, z_1 \approx z_3, z_2 \not\approx z_3\}$.

The empty clause is derived by the optimized procedure.

(1) $y_1 \not\approx y_3 \vee y_2 \not\approx y_4 \vee z_1 \approx z_2$	
(2) $y_1 \approx y_3$	
(3) $y_2 \approx y_4$	
(4) $z_1 \approx z_3$	
(5) $z_2 \not\approx z_3$	
(6) $y_2 \not\approx y_4 \vee z_1 \approx z_2$	(1,2)
(7) $z_1 \approx z_2$	(2,6)
(8) $\bot$	(4,5,7)

6 Example

As an example we consider a formula that is related to the pigeon hole formula in proposition calculus. Just like the pigeon hole formula our formula is parameterized by a number n, it is easily seen to be contradictory by a meta argument, and its shape is the conjunction of two subformulas. In our formula there are $n+1$ variables $x_1, \ldots, x_n, y$. The first subformula states that all values of $x_1, \ldots, x_n$ are different. The second subformula states that the value of y occurs in every subset of size $n-1$ of $\{x_1, \ldots, x_n\}$, hence it will occur at least twice in $\{x_1, \ldots, x_n\}$, contradicting the property of the first subformula. Hence the total formula

$$\Phi_n \equiv \bigwedge_{1 \le i < j \le n} x_i \not\approx x_j \wedge \bigwedge_{j=1}^{n} \Big(\bigvee_{i \in \{1,\ldots,n\}, i \neq j} x_i \approx y \Big)$$

is unsatisfiable as an E-CNF . It is easy to see that Φ_n is minimally unsatisfiable, hence in any proof of unsatisfiability all $\frac{n(n+1)}{2}$ clauses have to be used. The goal now is to prove unsatisfiability of Φ_n automatically.

We applied the bit vector encoding to this formula, i.e., in this formula every $z \approx w$ is replaced by $\bigwedge_i (z_i \leftrightarrow w_i)$ for i running from 1 to $\lceil \log(n+1) \rceil$ and then a standard SAT approach is applied for the resulting propositional formula. It turned out that both for a BDD-based approach and a resolution based approach this is a hard job. For $n = 50$ or even lower a combinatory explosion comes up.

However, by applying the approach introduced in this paper proving unsatisfiability of Φ_n can be done polynomial in n. It turns out that all contradictory cycles in Φ_n are of length 3 and are of the shape $\theta_{ij} = \{x_i \approx y, x_j \approx y, x_i \not\approx x_j\}$ for $1 \leq i < j \leq n$; the total number of these contradictory cycles is $\frac{n(n-1)}{2}$. Now we will study the behavior of our procedure consecutively proceeding all these contradictory cycles. Write C_j for the clause $\bigvee_{i\in\{1,\ldots,n\},i\neq j} x_i \approx y$ for $j = 1, \ldots, n$, and write C_{jn} for the clause obtained from C_j by removing $x_n \approx y$, for $j = 1, \ldots, n-1$. As a first contradictory cycle choose $\theta_{1,n}$. Then by applying a $\theta_{1,n}$-step on C_1, C_n and $x_1 \not\approx x_n$ we obtain the new clause C_{1n}. Another number of $\theta_{1,n}$-steps is possible, but each of them yields a clause in which C_{1n} is contained, hence will be removed. Also C_1 and C_n are supersets of C_{1n} and will be removed. So after treating this first contradictory cycle apart from the inequalities only the following $n-1$ clauses remain: $C_2, \ldots, C_{n-1}, C_{1n}$. As a second contradictory cycle choose $\theta_{2,n}$. Applying a corresponding step on C_2, C_{1n} and $x_2 \not\approx x_n$ yields the new clause C_{2n}. Since this is a subclause of all other clauses generated by $\theta_{2,n}$-steps, and also of C_2, after treating this second contradictory cycle apart from the inequalities only the following $n-1$ clauses remain: $C_3, \ldots, C_{n-1}, C_{1n}, C_{2n}$.

This pattern continues after choosing the $n-1$-th contradictory cycle $\theta_{n-1,n}$ apart from the inequalities only the following $n-1$ clauses remain: $C_{1n}, C_{2n}, \ldots, C_{n-1,n}$. Since now no equality occurs any more involving the variable x_n, there is no contradictory cycle any more containing the inequalities $x_i \not\approx x_n$ for $i = 1, \ldots, n-1$. It turns out that the remaining E-CNF is exactly Φ_{n-1}. Continuing with consecutively choosing $\theta_{1,n-1}$, $\theta_{2,n-1}$, ..., after $n-2$ steps the remaining E-CNF is exactly Φ_{n-2}. This goes on until the remaining E-CNF is exactly Φ_2 consisting of the three unit clauses $x_1 \approx y$, $x_2 \approx y$, $x_1 \not\approx x_2$ from which the empty clause is derived in one single θ_{12}-step.

We conclude that all $\frac{n(n-1)}{2}$ contradictory cycles were proceeded before the empty clause was derived. Surprisingly, after removing redundant clauses, in intermediate steps the total number of clauses was never greater than the original number of clauses.

7 Concluding Remarks and Further Research

We developed a new rule for reasoning with E-CNFs. We proved its soundness and completeness. We proposed an algorithm based on this rule for satisfiability of E-CNFs, and also proved soundness and completeness of this procedure. Until now we have this procedure only in a high-level pseudo-code. Many implementation details have not yet been considered. However, on a theoretical level we analyzed the complexity of our procedure when applied to a particular formula, yielding a polynomial complexity, while standard approaches applied to this formula show up an exponential behavior. This is quite hopeful for our new approach, and as a next step we will implement our procedure and will do experiments with real benchmarks.

References

1. Ackermann, W. *Solvable cases of the decision problem.* Studies in Logic and the Foundations of Mathematics. North-Holland, Amsterdam, 1954.
2. Barrett, C. W., Dill, D., and Levitt, J. Validity checking for combinations of theories with equality. In *Formal Methods in Computer-Aided Design (FMCAD'96)* (November 1996), M. Srivas and A. Camilleri, Eds., vol. 1166 of *LNCS*, Springer-Verlag, pp. 187–201.
3. Bryant, R., German, S., , and Velev, M. Processor verification using efficient reductions of the logic of uninterpreted functions to propositional log. *ACM Transactions on Computational Logic 2*, 1 (January 2001), 93–134.
4. Bryant, R., and Velev, M. Boolean satisfiability with transitivity constraints. *ACM Transactions on Computational Logic 3*, 4 (October 2002), 604–627.
5. Burch, J., and Dill, D. Automated verification of pipelined microprocesoor control. In *Computer-Aided Verification (CAV'94)* (June 1994), D. Dill, Ed., vol. 818 of *LNCS*, Springer-Verlag, pp. 68–80.
6. Goel, A., Sajid, K., Zhou, H., Aziz, A., and Singhal, V. BDD based procedures for a theory of equality with uninterpreted functions. In *Computer-Aided Verification (CAV'98)* (1998), A. J. Hu and M. Y. Vardi, Eds., vol. 1427 of *LNCS*, Springer-Verlag, pp. 244–255.
7. Groote, J., and van de Pol, J. Equational binary decision diagrams. In *Logic for Programming and Reasoning (LPAR'2000)* (2000), M. Parigot and A. Voronkov, Eds., vol. 1955 of *LNAI*, pp. 161–178.
8. Pnueli, A., Rodeh, Y., Shtrichman, O., and Siegel, M. Deciding equality formulas by small domains instantiations. In *Computer Aided Verification (CAV'99)* (1999), vol. 1633 of *LNCS*, Springer-Verlag, pp. 455–469.
9. Robinson, G., and Wos, L. Paramodulation and theorem-proving in first-order theories with equality. *Machine inteligence 4* (1969), 135–150.
10. Robinson, J. A machine-oriented logic based on the resolution principle. *Journal of the ACM 12(1)* (1965), 23–41.
11. Rodeh, Y., and Shtrichman, O. Finite instantiations in equivalence logic with uninterpreted functions. In *Computer Aided Verification (CAV'01)* (July 2001), vol. 2102 of *LNCS*, Springer-Verlag, pp. 144–154.