Topic

Event

Author: Aklima Zaman
Supervision: Erika Ábrahám
RWTH Aachen University, LuFG Informatik 2

WS 16/17

Abstract

We present a procedure as published in [1] which computes all the solutions of non-linear equalities with virtual substitution.The solution approach relies on some substitution rules of virtual substitution. The rules can be applied only on the real-arithmetic formulas which is linear or quadratic.

1 Introduction

Quantifier/ variable elimination for elementary real algebra is a fundamental problem. This problem can be solved easily by using virtual solution. In 1993, the concept of virtual substitution was first introduced. Initially it was a procedure to eliminate quantifier/variable elimination for linear real arithmetic formulas.Further, virtual substitution became a procedure of quantifier elimination for non-linear arithmetic formulas.Virtual substitution cannot eliminate quantified variables whose degree is higher than 2.

Section 2 consists some preliminaries with some definitions. How to construct the real zeros is explained in section 3. In the next section we will know some substitution rules by which we can eliminate variables from a formula with an example. In the section 4 an idea to eliminate quantifiers is explained by an example and conclude in the last section.

2 Preliminaries

2.1 CNF

A formula is said to be in CNF if and only if it is a conjunction of clauses adn a clause is a disjunction of literals. A literal could be a positive or negative instance of Boolean variable. A CNF formula φ is defined as follows,

$$page3$$

here, C_i is the i^{th} clause and n is the number of total clauses in φ, a_{ij} is the literal and m is the total number of literals in C_i.

Example2: Assume the following CNF formula,

$$clauseMarkKorteHobe\varphi = (x) \wedge (\neg x) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$$

where, $x, \neg x, y$ and $\neg y$ are the literals.

The aforementioned CNF formula will ne used as an example throughout this paper which is unsatisfiable.

2.2 Satisfiability Checking

Satisfiability checking determines whether the existentially quantified first-order-logic formulas are satisfiable by generating solutions automatically. The formulas are Boolean combinations of theory constraints and the form of theory constraints varies depending on with which theory we want to represent first-order-logic. Some example theory constraints from different theories are given in [2].By sat solving we can check the satisfiability of formulas, though SAT-modulo-theories (SMT) solvers are available and popular theories.

A SAT solver solves the Boolean satisfiability problem by implementing decision procedure. The DPLL algorithm is the basis for most modern SAT solvers. The input formula of it is expected to be in conjunctive normal form (CNF).

Example1: Let us consider a CNF $(x) \wedge (\neg x \vee \neg y) \wedge (z)$. First x is set to $true$. Boolean constraint propagation (BCP) indicates that y of the second clause must be $false$ to satisfy the CNF. There is still a unassigned variables. A new decision will be made and z is set to $true$. As all variables are assigned and there is no conflict, the satisfying soluiton is $x = 1, y = 0$ and $z = 1$.

SMT solvers can be applied on quantifier-free first-order-logic formulas with an underlying theory to check the satisfiability. There are two type of SMT solvers, Eager SMT solver and Lazy SMT solver. To know about SMT in details you can read [3].

2.3 Clause-Selector Variables

Authors of the paper [1] has used clause-selector variable by augmenting a CNF formula. Clause-selector variable is a variable w_i which is negated to augment each clause of C_i of a CNF such that $C'_i = (\neg w_i) \vee C_i$ in a new formula φ'. Notice that each C'_i is an implication, $C'_i = (\neg w_i \rightarrow C_i)$. It menas if w_i is set to $true$, the original clause is being enabled. Conversely, assigning w_i $false$ means C_i is being disabled or removed from the set of constraints, because C'_i is satisfied by the assignment to w_i. So, adding clause-selector variables provides the SAT solver the ability to enable and disable constraints as a part of its normal search.

For our example formula, after adding these clause-selector variables we get the following augmented formula φ',

$$\varphi' = (\neg w_1 \vee x) \wedge (\neg w_2 \vee \neg x) \wedge (\neg w_3 \vee \neg x \vee y) \wedge (\neg w_4 \vee \neg x \vee \neg y)$$

2.4 Minimal Unsatisfiable Subset and Minimal Correction Subset

A Minimal Unsatisfiable Subset (MUS) ia a subset of the clauses of an unsatisfiable CNF formula φ that is both unsatisfiable and minimal. A Minimal Correction Subset (MCS) is a subset of the clauses of φ whose removal from φ make it satisfiable and which is minimal. A formula can have multiple MUSes and MCSes. The formal definitionss of the set of MUSes and MCSes of φ is given following:

Definition1: A subset U be a MUS which is a subset of φ if U is unsatisfiable and for each clause C_i containing in U the removal of C_i from U is satisfiable.

$$khataThekeLikhteHobe$$

Defintion2: A subset M be a MCS which is a subset of φ if the removal of M from φ is satisfiable and for each clause C_i of M the removal of M from φ is unsatisfiable where M does not have C_i.

$$khataThekeLikhteHobe$$

For our example formula $\varphi = (x) \wedge (\neg x) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$ the MUSes and MCSes are shown in the following table:

$$tableAkteHobe$$

2.5 Hitting Sets

Assume a collection Ω of sets of a finite set D. A hitting set of Ω is a subset of D such that it contains at least one element from each subset in Ω. Formally, the definition is given below:

Definition3: A hitting set H of Ω is $H \subseteq D$ such that $\forall S \in \Omega$, $H \cap S \neq 0$ where S represents each subset of Ω.

In the paper [1], authors refer to minimal hitting sets which are the hitting sets from which no elements cannot be removed and if at least one element is removed, it will not be a hitting set anymore.

Example: Let us consider, $\Omega = \{\{a, b\}, \{b, c, d\}\}$. Then, $\{a, b\}, \{b, c, d\}, \{a, c, d\}, \{b\}, \cdots$ are the hitting sets where, $\{b\}$ is the only minimal hitting set.

2.6 MUS backSlashDiteHobe MCS Duality

There is a relationship between MUS and MCS which is the foundation of paper [1]. The relationship states that the set of MUSes of formula φ is equal to theset of minimal and irreducible hitting sets of the set of MCSes and vice-versa. This is the duality of MUS and MCS. Fomally it can said that:

1. A subset U of φ is an MUS if and only if U is an irreducible hitting set of MCSes(φ).
2. A subset M of φ is an MUS if and only if M is an irreducible hitting set of MUSes(φ).

$$dualityPicAkteHobe$$

To show duality we use our example formula $\varphi = (x) \wedge (\neg x) \wedge (\neg x \vee y) \wedge (\neg x \vee \neg y)$. First, take the set MCSes of φ and it is $\{\{C_1\}, \{C_2, C_3\}, \{C_2, C_4\}\}$. For this, the set of minimal and irreducible hitting sets is $\{\{C_1, C_2\}, \{C_1, C_{3,} C_4\}\}$ which is exactly like the set of MUSes (given in Table *blabla*). Similarly, we can show that the set of MCSes is the set of irreducible hitting sets of the set of MUSes. So, every MCS has at least one clause from every MUS and vice-versa. The duality is illustrated in Figure(blabla).

2.7 AtMost Constraints

AtMost Constraint is a type of counting constraint that can be generated from many types of constraints. If $\{l_1, l_2, l_3, \cdots, l_{\mathrm{n}}\}$ is a set of n literals and k is a positive integer, an AtMost constraint is defines as,

$$AtMosterFormulaLikhteHobe$$

where, $val(l_{\mathrm{i}})$ is 1 if l_i is assigned *true* and otherwise 0. k is a bound which tells that maximum how many literals can be assigned *true*. Authors used an implementation of the AtMost constraint where the variables are watched and propagates the negation of each literal when k of them are assigned *true*. This is implemented with exactly n watched literals and a counter that is incremented or decremented whenever on of them is assigned or unassigned.

3 Algorithms for Computing Minimal Unsatisfiable Subset

The approach of computing all MUSes of φ is to first find all MCSes(φ) and then to compute all irreducible hitting sets of MCSes(φ), which are all MUSes(φ). So, there are two phases to generate all MUSes of a unsatisfiable CNF formula. The first phase is to compute MCSes by using an algorithm and the second phase is to compute MUSes from the MCSes by using a recursive algorithm which authors have developed to compute irreducible hitting stes.

In the first phase, authors use a SAT solver (produces an satisfying assignment, if exists) to work with the input given and provide hitting sets of MUSes without revealing the underlying MUSes. For the second phase, they get all the information with MCSes and they use an recursive algorithm to generate minimal and irreducible hitting sets.

$$algorithmMCSLikhteHobe$$

3.1 Computing MCSes

Algorithm 1 finds MCSes by solving a CNF formula φ. The goal is to find minimal set of clauses which makes φ satisfiable. In Line 1, φ is augmented with clause-selector variables and create φ'. By augmenting φ increase the number of variables in φ. Also the search space grows for finding a satisfying assignment. Minimal set of variables w_i are assigned to $false$ so that minimum number of clauses are disabled to finding the satisfying assignment of φ'. The set of variables w_i are assigned to $false$ indicates the clauses as an MCS. Each iteration of the outer while loop (lines 4-12) finds an MCS of size k, which is incremented by 1 after each iteration.

3.2 Compuitng MUSes

4 Quantifier Elimination with the Virtual Substitution

Let $\varphi^{\mathbb{R}}$ is a quantifier-free real-arithmetic formula where $x \in \varphi^{\mathbb{R}}$ and x has at most degree 2 in $\varphi^{\mathbb{R}}$. After eliminating existential and universal qualifier with virtual substitution we will get the following equivalences,

$$\exists x.\varphi^{\mathbb{R}} \iff \bigvee_{t \in T(x,\varphi^{\mathbb{R}})} (\varphi^{\mathbb{R}}[t\backslash\backslash x] \land C_t) \tag{4.1}$$

$$\forall x.\varphi^{\mathbb{R}} \iff \bigwedge_{t \in T(x,\varphi^{\mathbb{R}})} (C_t \to \varphi^{\mathbb{R}}[t\backslash\backslash x]) \tag{4.2}$$

Now, let us consider $\exists x.\varphi$ where, $\varphi = (x^2y + x + y = 0) \land (y^2 - 2 < 0)$. We already constructed all the test candidates for x and y. Also we get to know about substitution rules with virtual substitution. By using the substitution rules, we eliminate all occurrences of x and y from φ. Then we get the following equivalence holds.

$$\exists x \exists y.\varphi \iff \bigvee_{t_1 \in T(x,\varphi)} (\bigvee_{t \in T(y,\varphi')} (\varphi'[t_2\backslash\backslash y] \land C_{t_2}))$$

where, $\varphi' = (\varphi[t_1\backslash\backslash x] \land C_{t_1})$.

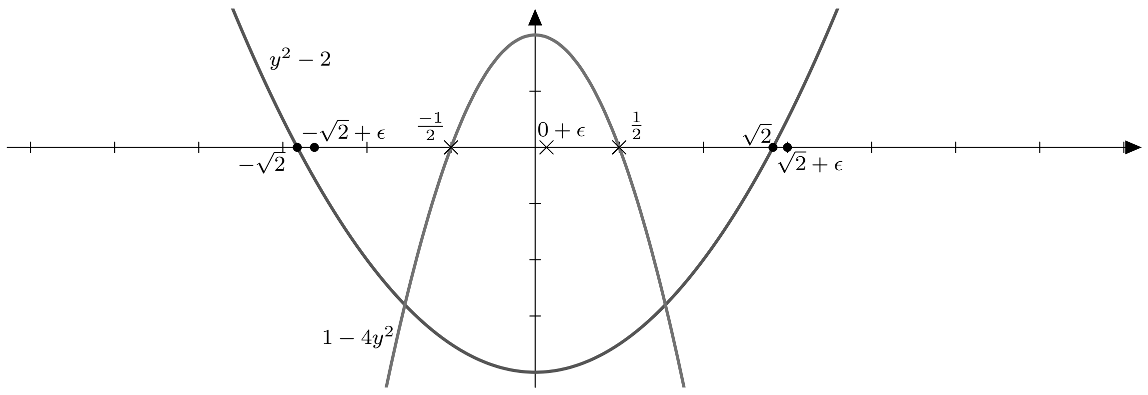

Figure 1: Solution of y in φ_4 where $x = x_0$ and x_2

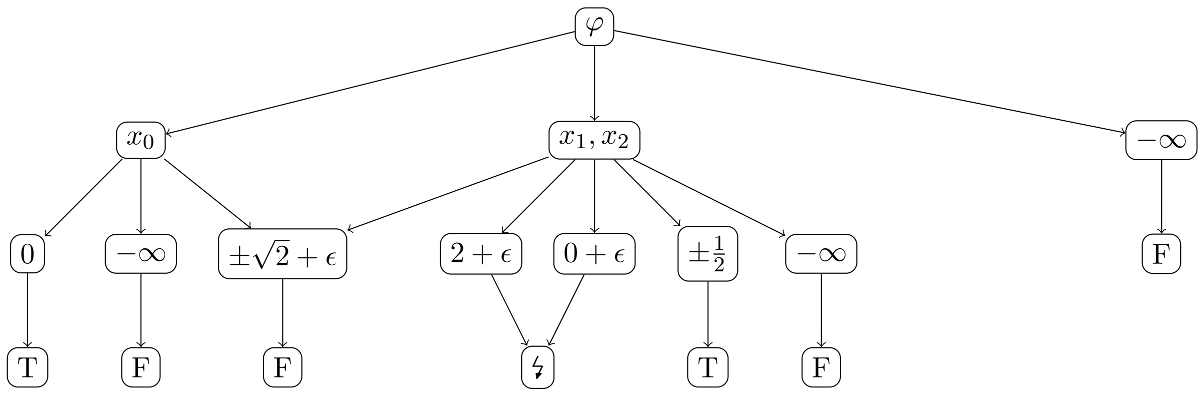

Figure 2: Example of Virtual Substitution.

The solutions of x and y for φ is shown in the figure 2 and 3. Also we can have an overview of virtual substitution though figure 4.

We get the proof of equation (4.1) which is explained on the above by an example. Equation (4.2) can be implied by equation (4.1) as follows.

$$\begin{aligned}
\forall x.\varphi^{\mathbb{R}} &\iff \neg\exists x.\neg\varphi^{\mathbb{R}} && (4.3)\\
&\overset{4.1}{\iff} \neg(\bigvee_{t\in T(x,\varphi^{\mathbb{R}})})\neg(\varphi^{\mathbb{R}}[t\backslash\backslash x] \wedge C_t) && (4.4)\\
&\iff \bigwedge_{t\in T(x,\neg\varphi^{\mathbb{R}})} (\varphi^{\mathbb{R}}[t\backslash\backslash x] \vee \neg C_t) && (4.5)\\
&\iff \bigwedge_{t\in T(x,\varphi^{\mathbb{R}})} (C_t \rightarrow \varphi^{\mathbb{R}}[t\backslash\backslash x]) && (4.6)
\end{aligned}$$

5 Conclusion

We have described the solving technique of non-linear equalities with virtual substitution step by step. It has two steps. One is to construct test candidates with side conditions and another one is to replace a variable by a test candidate in a formula. It has already said that the replacement is based on some substitution rules.Finally, we can have the solutions for which the formula is satisfied.

References

[1] V. Weispfenning, *Quantifier elimination for real algebra - the quadratic case and beyond.* Appl. Algebra Eng. Commun. Comput, 1997.

[2] R. Loss, V. Weispfenning, *Applying linear quantifier elimination.* The computer Journal 36 (1993), pp. 450-462.