ONE IMPLEMENTATION OF THE QUANTIFIER ELIMINATION FOR THE THEORY OF STRUCTURE $(\mathbb{Q}, +, -, <, 0)$

Dragan Doder, Mirna Udovičić, Miloš Milošević, Dejan Ilić,
Group for the intelligent systems (GIS), Mathematical Faculty Belgrade

Abstract *An implementation of quantifier elimination for the structure* $(\mathbb{Q}, +, -, <, 0)$ *is given with the discussion of further integration of developed software.*

1. INTRODUCTION

By quantifier elimination we assume the existence of an effective procedure of finding a quantifier free formula (i.e formal expression without occurrence of quantifier symbols $\forall$ and $\exists$) which is equivalent to the given formula in some fixed formalism. By formalism we usually assume a first order theory of some recursive language, or a first order theory of certain recursive structure.

The applicability of the method lies in the fact that each problem that can be reformulated by validity of some first order formula (possibly with quantifiers) can be reduced to the validity of a quantifier free first order formula. This yields a decision procedure if the ground model (structure) is recursive.

In this article we shall present a procedure for the quantifier elimination for $(\mathbb{Q}, +, -, <, 0)$.

2. QUANTIFIER ELIMINATION

By a language we will assume a recursive set L which elements (if any) are constant, functional and relational symbols. Of course, sets of constant symbols, functional symbols and relational symbols are pairwise disjoint. By a theory of language L we will assume a recursive set of sentences of L.

We say that T admits quantifier elimination if for each formula φ of L there is a quantifier free formula ψ such that

$$T \vdash \varphi \leftrightarrow \psi, \tag{1}$$

i.e. there is a finite sequence of formulas of L which finishes with $\varphi \leftrightarrow \psi$ and any other member of the sequence is either axiom of a first order predicate calculus, or belongs to T, or it is obtained by some inference rule (modus ponens and generalization) from some predecessors.

To simplify notation, instead of $T \vdash \varphi \leftrightarrow \psi$ we shall write $\varphi \sim \psi$. Suppose that we can effectively eliminate the existential quantifier from each formula of the form $\exists x\varphi$, where φ is a quantifier free formula. Then we also have a recursive quantifier elimination procedure described as follows:

Input: formula φ
Output: quantifier free formula ψ such that $\varphi \sim \psi$.

- If φ has a form $\varphi_1 * \varphi_2$, where $*$ is one of the connectives $\wedge, \vee$, $\rightarrow$ and $\leftrightarrow$, then first find quantifier free formulas ψ_1 and ψ_2 such that $\varphi_i \sim \psi_i$, then $\psi = \psi_1 * \psi_2$.
- If φ has a form $\neg\varphi_1$, then first find a quantifier free formula ψ_1 such that $\varphi_1 \sim \psi_1$, then $\psi = \neg\psi_1$.
- If φ has a form $\exists x\varphi_1$, then first find a quantifier formula ψ_1 such that $\varphi_1 \sim \psi_1$, then find a quantifier free formula ψ such that $\exists x\psi_1 \sim \psi$.
- If φ has a form $\forall x\varphi_1$, then proceed as described in the previous two clauses for the formula $\neg\exists x\neg\varphi$.

Thus, theory T admits the quantifier elimination if and only if we can eliminate the existential quantifier from each formula of the form $\exists x\varphi$, where φ is a quantifier free formula.

Back to our problem, let T be a theory of the structure $(\mathbb{Q}, +, -, <, 0)$, i.e. T is the set of all sentences which are true in the mentioned model. We want to show that T admits the quantifier elimination. To do that, first we need to analyze quantifier free formulas by its complexity. Up to equivalence, the atomic formulas have one of the two following forms:

$$n_1x_1 + \cdots + n_kx_k = 0 \tag{2}$$

and

$$n_1x_1 + \cdots + n_kx_k < 0, \tag{3}$$

where n_i are integers $\neq 0$ and, for example, $2x$ and $-2x$ are respectively abbreviations for the expressions $x + x$ and $(-x) + (-x)$ (here "+" and "−" are treated as formal symbols). Further, the negation of the formula (2) is the formula

$$(\sum_{i=1}^{k} n_ix_i) < 0 \vee (\sum_{i=1}^{k} -n_ix_i) < 0, \tag{4}$$

and the negation of the formula (3) is the formula

$$(\sum_{i=1}^{k} n_ix_i) = 0 \vee (\sum_{i=1}^{k} -n_ix_i) < 0. \tag{5}$$

If we combine this with the fact that negations of formulas $\varphi \wedge \psi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$ and $\varphi \leftrightarrow \psi$ are respectively formulas $\neg\varphi \vee \neg\psi$, $\neg\varphi \wedge \neg\psi$, $\varphi \wedge \neg\psi$ and $(\neg\varphi \wedge \psi) \vee (\varphi \wedge \neg\psi)$, we can conclude that each quantifier free formula is, up to equivalence, a "nice" (i.e. without occurrence of $\neg, \rightarrow$ and $\leftrightarrow$) boolean combination of atomic formulas (2) and (3). To further simplify notation, let

$$\frac{m_1}{n_1}x_1 + \cdots + \frac{m_k}{n_k}x_k = 0 \tag{6}$$

be an abbreviation of

$$m_1 n_2 \cdots n_k x_1 + \cdots + m_k n_1 \cdots n_{k-1} x_k = 0, \tag{7}$$

and let

$$\frac{m_1}{n_1} x_1 + \cdots + \frac{m_k}{n_k} x_k < 0 \tag{8}$$

be an abbreviation of

$$m_1 n_2 \cdots n_k x_1 + \cdots + m_k n_1 \cdots n_{k-1} x_k < 0 \tag{9}$$

(here n_i are positive integers). Finally, we are ready to sketch the algorithm:
Input: a formula $\exists x \varphi$, where φ is a quantifier free formula.
Output: a quantifier free formula $A(\varphi)$ with the property $\exists x \varphi \sim A(\varphi)$.

- Check whether x is a dummy variable for φ or not. If x is a dummy variable, then $A(\varphi) = \varphi$. Otherwise, proceed with an algorithm.
- If φ is a formula of the form $\bigvee_{i=1}^{n} \varphi_i$ then $A(\varphi)$ is the formula $\bigvee_{i=1}^{n} A(\varphi_i)$.
- If φ is a formula of the form $\bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} \varphi_{ij}$ with at least one i such that $n_i > 1$ holds, then $A(\varphi)$ is the formula
$$\bigvee_{f \in \mathcal{F}} A\Big(\bigwedge_{i=1}^{m} \varphi_{if(i)}\Big),$$
where $\mathcal{F}$ is the set of all functions with domain $\{1, \ldots, m\}$ such that $f(i) \in \{1, \ldots, n_i\}$, for all i from the domain.
- If φ is a formula of the form $n_0 x + A = 0 \wedge \psi(x)$, where A is a term without occurrence of x, then $A(\varphi)$ is the formula $A(\psi(-A/n_0))$.
- If φ is a formula of the form $\bigwedge_{i=1}^{k} (p_i x + A_i < 0)$, where A_i are terms without occurrence of x and p_i are positive integers, then $A(\varphi)$ is the formula representing logical truth.
- If φ is a formula of the form $\bigwedge_{j=1}^{l} (n_j x + B_j < 0)$, where B_j are terms without occurrence of x and n_j are negative integers, then $A(\varphi)$ is the formula representing logical truth.
- If φ is a formula of the form
$$\Big(\bigwedge_{i=1}^{k} (p_i x + A_i < 0)\Big) \wedge \Big(\bigwedge_{j=1}^{l} (n_j x + B_j < 0)\Big)$$
where A_i and B_j are terms without occurrence of x, p_i are positive integers and n_i are negative integers, then $A(\varphi)$ is the formula
$$\bigwedge_{i=1}^{k} \bigwedge_{j=1}^{l} (p_i B_j + n_j A_i < 0).$$

Note on complexity: It is well known that this problem is NP-complete. This procedure can be easy transformed to the procedure for nondeterministic machines of polynomial complexity. On deterministic machines it has exponential complexity.

3. IMPLEMENTATION

Usage: The input file is a simple ASCII file containing commands `\input` , `\def` and `\write`. Symbols "`%`" and "`#`" denotes commentary to the end of the current line. For example, lines

```
\input "inputf#first commentary
#second commentary
%third commentary
ile.qe"
\de#fourth commentary
f f(first se%fifth commentary
cond)
\forall first \exi%
%
%
sts second first < second
```

have the same effect as

```
\input "inputfile.qe"
\def f(first second)
\forall first \exists second first < second
```

Identifier is any sequence of letters and/or digits starting with a letter. There are no keywords. The syntax of the `\input` command is

`\input "`*filename*`"`

where *filename* is the name of the file. This name must be written in only one line. The effect of this command is the same as replacing it with complete contents of the named file (similar to the preprocessor directive `#include "`*filename*`"` in the programming language C). The syntax of the `def` command is

`\def`*label*`(`*variableslist*`)` *formula*

where *label* is any identifier and *variableslist* is the list of identifiers separated by white space characters. The effect of this command is in assigning of the identifier *label* to the given formula which all free variables are some of listed variables in the *variableslist*. The syntax of the `write` command is

`\write` *formula*`(`*formulaelist*`)`

where *formula* is any identifier and *formulaelist* any list of any number of formulae separated with white space characters. The effect of this command is in the printing of formula assigned by the identifier *formula*, where free variables are replaced by formulae listed in *formulaelist* with respect to the order of variables from the definition of the *formula*.

Now we describe the format of the formula in the input file.

Unsigned integer: Any sequence of letters and digits starting with letter. It is nonzero if contains at least one nonzero digit.

Integer: Any sequence of any of forms n, $+n$, $-n$, where n is unsigned integer. It is nonzero if such is n.

Constant: Any sequence of any of forms m, m/n, where m is integer and n is nonzero integer. It is nonzero if such is m.

Variable: Any sequence of letters and digits starting with letter.

Term: The set of all terms is the least set of sequences with following properties:

1. Constants and variables are terms.
2. If c is a constant and t is a term, then sequences `c*t`, `t*c` are also terms. If c is nonzero and t is a term, then sequence `t/c` is also the term.
3. If t and t' are terms, then `t+t'` and `t-t'` are also terms.
4. If t is a term, then (t) is also a term.

Formula: The set of all formulae is the least set of sequences with following properties:

1. If t and t' are terms, then $t \cdot t'$ is also a formula, where $\cdot$ is any of sequences `<`, `>`, `<=`, `>=`, `=`, `<>`.
2. If f is a formula, then (f) is also a formula.
3. If f and f' are formulae, then $f \cdot f'$ is also a formula, where $\cdot$ is any of sequences `/\`, `&`, `\land`, `\wedge`, `\/`, `|`, `\lor`, `\vee`, `=>`, `->`, `\rightarrow`, `\Rightarrow`, `<=>`, `<->`, `\leftrightarrow` and `\Leftrightarrow`.
4. If f is a formula then $\cdot f$ is also a formula, where $\cdot$ is any of sequences `~`, `\lnot` and `\neg`.
5. If x is a variable and f is formula, then any of sequences `\exists` x f and `\forall` x f is a formula.
6. If f is a formula, then any of sequences `[\exists` *varlist*`]`f and `[\forall` *varlist*`]`f is the formula, where *varlist* is arbitrary list of variables separated by white space characters.
7. If f is a formula, then the sequence `\qe(f)` is also a formula. The effect of the function `\qe()` is in replacing of given formula by equivalent quantifier free formula.

Tokens can be separated by arbitrary number of white space characters. The only exception is argument of the `\input` command. In the above definitions white space characters occur only in those cases where at least one white space character is needed. Instead of `\exists` one can use `\e`, and instead of `\forall` one can use any of `\f` and `\a`.

4. FURTHER RESEARCH

The main reason for this implementation is our attempt to solve the following open problem:

Which is the minimal number of squares (each square has a different length of the edge) required for covering of unit square in euclidian plane?

It is known that such covering exists and that in each covering squares have rational lengths of edges. It is also known that this problem can be reduced to the validity of certain formula in the structure $(\mathbb{Q}, +, -, <, 0)$.

REFERENCES

[1] C. C. Chang, H. J. Keisler, "Model Theory", Springer-Verlag 1990

[2] M. Milošević, Dragan Doder, Mirna udovičić, Filip Marić, "Quantifier Elimination - Algorithms and Applications", in *Proc. 2nd Serbian-Hungarian Joint Symposium on Intelligent Systems*, Subotica 2004, pp 215-222

[3] M. Milošević, M. Udovičić, D. Doder, D. Ilić, "Quantifier Elimination in Mathematical Theories", *Proc. XLVIII ETRAN Conference*, Čačak 2004