Combining Logical and Algebraic Techniques for Natural Style Proving in Elementary Analysis

Robert Vajda, Tudor Jebelean and Bruno Buchberger

RISC
Johannes Kepler University
Linz, Austria

Abstract

PCS (Proving-Computing-Solving) [Buchberger 2001] and S-Decomposition [Jebelean 2001] are strategies for handling proof problems by combining logic inference steps (e.g., modus ponens, Skolemization, instantiation) with rewriting steps (application of definitions) and solving procedures based on algebraic techniques (e.g., Groebner Bases, Cylindrical Algebraic Decomposition).

If one formalizes the main notions of elementary analysis like continuity, convergence, etc., usually a sequence of alternating quantifier blocks pops up in the quantifier prefix of the corresponding formula. This makes the proof problems involving these notions not easy. S-Decomposition strategy is especially suitable for property-preserving problems like continuity of sum, because it is designed for handling problems where the goal and the main assumptions have a similar structure. During proof deduction, existentially quantified goals and universal assumptions are handled by introducing metavariables, if no suitable ground instance is known in advance. For finalizing proof attempts, the metavariables should be instantiated in such a way that they satisfy the cumulated algebraic constraints collected during the proof attempt. The instantiation problem is considered to be difficult in the logical calculus. Appropriate instances can be often found using quantifier elimination (QE) over real closed fields. In order to obtain witness terms we utilize the QE method based on cylindrical algebraic decomposition (CAD) [Collins 1975]. However, the QE method alone is not sufficient. One needs to pre-process the (closed, quantified) conjectured formula and post-process the resulting CAD-structure after the call of the QE algorithm.

Key words: automated theorem proving, extended quantifier elimination, CAD, Groebner Bases
MSC 2000: 68T15, 03B35, 03C10

1 Introduction

In automated reasoning, a calculus given by its inference rules and axioms is used in order to deduce automatically the goal formula (ψ) from the assumptions (from the ϕ_i's).

$$\phi_1, \phi_2, \cdots, \phi_n \vdash \psi$$

The resolution calculus has only one inference rule and it is a *refutational* calculus, i.e. the resolution method aims to show the unsatisfiability of the formula set formed by the assumptions and the negation of the goal ($\{\phi_1, \phi_2, \cdots, \phi_n, \neg\psi\}$). In fact, a pre-processing step is needed for transforming all forumale involved in the problem to normal form before a contradiction may be deduced by resolution. Resolution based automatic theorem proving was successful in many areas (classification of quasigroups, axiomatization of Boolean algebras etc.), but it turns out to be not sufficiently efficient for practical application fields like mathematical education or program verification. One main reason for this appears to be the difficulty of using other mathematical techniques (e.g., algebraic computation) in conjunction with the resolution method. Also, with resolution, human readability and human-like inferencing is lost (cf. [12] for treating elementary calculus problems using resolution). The alternative would be to use a system of natural deduction. However, there are currently no efficient strategies for combining natural inference rules into a general prover for predicate logic.

One promising way for improving the capabilities of natural deduction systems is to apply a particular strategy with combination of domain-specific decision methods which are based on algebraic techniques. PCS (Proving-Computing-Solving) [3] and S-Decomposition [6] are strategies for handling proof problems by combining logic inference steps (e.g., modus ponens, Skolemization, instantiation) with rewriting steps (application of definitions) and solving procedures based on algebraic techniques (e.g., Groebner Bases, Cylindrical Algebraic Decomposition).

Email address:
{Robert.Vajda,Tudor.Jebelean,Bruno.Buchberger}@risc.uni-linz.ac.at
(Robert Vajda, Tudor Jebelean and Bruno Buchberger).

2 Logical Reduction and Decomposition

While in a refutation based proving procedure one needs a (complete) unsatisfiable set of formulae for deducing a contradiction (the empty clause), we think that human reasoning usually starts from the definitions of the basic notions involved and then proceeds by adding the necessary knowledge as the proof develops.

Therefore, we base our approach on the following scenario:

- at the beginning, the current knowledge contains the basic definitions;
- by logical inferences, some necessary additional facts are identified (conjectures) which appear to be useful for completing the proof;
- these conjectures are proved using additional knowledge from certain theories (e.g., knowledge about constraints-handling over the reals, integers, etc.) in an algorithmic form.

The latter step can be typically solved by using a computer algebra system (but some additional transformations of the expressions involved may be needed - see below).

More challenging appears to be the reduction of the initial proving problem to subproblems which can be handled by the algebraic algorithms [3].

In this paper we focus on problems from elementary analysis. The definitions of the main notions in this theory (like continuity, convergence), usually involve a sequence of alternating quantifiers, which makes the respective proofs relatively difficult. A typical example is the convergence of a real sequence:

$$\text{Convergent}[f] \Leftrightarrow \exists_a \forall_{\epsilon>0} \exists_N \forall_n \, (n \geq N \Rightarrow |f[n] - a| < \epsilon)$$

Definition of the unary predicate "Convergent"

The logical reduction of the problem consists in applying the following types of inference steps:

- Standard rewrite for expanding definitions of predicates and functions: If a formula contains a subformula like Convergent$[f_0]$, the subformula will be rewritten using the right hand side of the definition with suitable substitutions.
- Introduction of Skolem constants for handling existential assumptions and universal goals: They stand for arbitrary but fixed elements, and they will not change during the current proof.
- Introduction of metavariables for handling universal assumptions and existential goals: They stand for terms which are currently unknown, and which

have to be determined during the proof.

Additionally, in order to generate polynomial expressions, one may have to introduce new variables for terms in which (universally quantified) function symbols occur.

We demonstrate these steps on a simple example from elementary analysis: Prove that *if a real valued sequence f is bounded from above then $\ominus f$ is bounded from below* ($\ominus$ stands for the standard unary 'minus' operator for sequences).

We trace only the transformations of the goal formula, although in our reasoning system we transform proof situations into other proof situations. For brevity, we omit the usual definitions and the type information. In each step, the current goal will be implied by the new goal.

$$\begin{array}{c} \forall_f \, (\mathrm{BoundedAbove}[f] \Rightarrow \mathrm{BoundedBelow}[\ominus f]) \\ \Uparrow \\ \exists_{K_1} \forall_{n_1} \, (f_0[n_1] < K_1) \Rightarrow \exists_{K_0} \forall_{n_0} \, (-f_0[n_0] > K_0) \\ \Uparrow \\ \forall_{K_1} \exists_{K_0} \forall_x \, (x < K_1 \Rightarrow -x > K_0) \end{array}$$

Example 1

Note that in order to obtain the last formula, we transformed the previous formula into prenex normal form and we introduced a new variable x (taking into account the quantification and order of the variables in the prenex normal form). Now since in the last goal all the variables ranges over the reals, this formula can be handled by real quantifier elimination methods. It turns out that this is a valid formula (in the theory of real closed fields).

In the simple example above the number of variables involved in the final reduced problem, which was handled by the QE method, was relatively small (3). However, if we start to investigate other important notions like limit, convergence, continuity, etc. and their interactions with binary operations, the number of variables can grow rapidly. Since most of the relevant algebraic algorithms have exponential complexity, a problem with 9-12 quantified variables is usually unfeasible with the current software and hardware. Therefore we use logical decomposition strategies to reduce the intitial problem to smaller problems. A typical decomposition step of the S-Decomposition method [6] is presented in the sequel. Suppose that the main formulae of the proving problem have similar structure:

$$\forall_x \, P_1[x] \Rightarrow Q_1[x], \ldots, \forall_x \, P_n[x] \Rightarrow Q_n[x] \; \vdash \; \forall_x \, P_0[x] \Rightarrow Q_0[x]$$

In this case two subproblems are generated:

$$P_0[x_0] \vdash P_1[x_1^*] \wedge \ldots \wedge P_n[x_n^*]$$

$$Q_1[x_1^*] \wedge \ldots \wedge Q_n[x_n^*] \vdash Q_0[x_0],$$

where x_0 is new Skolem constant and $x_1^*, \ldots x_n^*$ are metavariables.

Note that the subproblems are smaller than the original problem. Moreover, if the formulae $P_0, P_1, \ldots, P_n$ and Q_0, $Q_1 \ldots, Q_n$, respectively contain less variables than the original problem (in other words: the set of variables is separable), then there is a good chance that the induced subproblems are already tractable directly with QE or using a different logical decomposition rule, they could be simplified further.

3 Finding Witness Terms

After the decomposition of the initial problem, in order to finalize the proof attempt, we have to close each branch of the proof tree. From those branches which could be validated by the QE method, we need in fact more information besides the fact that the generated sentence is valid, namely we need to extract witness terms for the existentially quantified variables. These witness terms will be used as values for the metavariables, that is, in order to instantiate variables in universally quantified assumptions or in existentially quantified goals. We also say that the found witness term solves the quantified constraint problem on the corresponding branch. Moreover, since there can be dependencies among the open branches, it may happen that a particular instance which solves a QE problem on one branch must be propagated to another branch.

Turning back to Example 1, we will show that using QE and CAD a suitable witness, like $\{K_0 \leftarrow -K_1\}$, can be extracted for the initally existentially quantified variable K_0 in the goal. The extraction of witness terms is closely related to the problem of extended quantifier elimination.

4 The problem of extended quantifier elimination

The quantifier elimination (QE) problem is a well known problem in logic and in computer algebra [7, 9, 15] and can be summarized roughly as follows:

Given a first order language L and a first order theory T, decide if for all (arbitrarily quantified) formula ϕ there exist a quantifer-free formula ψ (of the language L), such that ϕ is T-equivalent to ψ. If this holds, then we say that the theory T admits quantifier elimination.

The 'effective', algorithmic solution of the above problem consists of giving an algorithm which constructs for an arbitrary input formula ϕ, an equivalent, quantifier-free formula ψ.

Several important first order theories, like the theory of dense linear orders with endpoints (DLO1), or algebraically closed fields (ACF), real closed fields (RCF), etc. admit quantifier elimination, moreover algorithms for carrying out the effective elimination of quatifiers with their complexity analyis are known as well.

Since we are aiming to exploit the quantifier elimination for elementary analysis, from this point on we focus on the theory of RCF and the corresponding QE problem.

The first QE-algorithm for the reals (with elementary complexity) is based on Collins' Cylindrical Algebraic Decomposition (CAD) [1, 4]. Other methods, like Weispfenning's virtual substitution method [8, 13] etc. are also used, but they are less suitable for our purpose (as we explain later). Several papers reported recently combining CAD with Groebner Bases computations for speeding up the elimination algorithm [10, 2].

If the input formula of the QE problem is closed (has no free variables), then the resulting formula contains no variables at all, hence it is a ground formula (like $1+1<0$). These formulae are decidable, so the algorithm finally provides **True**/**False**. Now, if the outermost quantifier (block) of a valid closed formula is an existential ($\exists_x \phi$), and the algorithm, besides the truth value **True**, provides an instance a for which $\phi_{x \leftarrow a}$ is True, then we say that the algorithm solves the *extended quantifier elimination* (EQE) [5, 11] problem.

$$\exists_x\, x > 0 \wedge x^3 - 2x^2 - x + 2 = 0 \quad \rightarrow_{QE} \quad \textbf{True}$$

$$\exists_x\, x > 0 \wedge x^3 - 2x^2 - x + 2 = 0 \quad \rightarrow_{EQE} \quad \{\textbf{True}, x = 1\}$$

Example 2: An extended quantifier elimination problem.

Note that the provided instance is in general not unique and in several problems it could be chosen from an infinite set. If the formula contains free variables as well, then the instance might be not generic, i.e., for different assignments of the free variables it may have different instances.

To the author's knowledge, currently three systems provide implementations

of real QE algorithms: *Mathematica*[1], *QEPCAD*[2], and *REDLOG*[3]. Only the REDLOG system provides an implementation for solving the extended QE problem. Since our reasoning system is implemented in Mathematica and this system provides efficient implementations of real QE and CAD, we found it reasonable to use those algorithms in order to implement EQE. However, we will continue to investigate the possible use of other systems, in particular REDLOG. Although this system has a different approach to QE (the answers typically involve new infinitesimal quantities), it appears that a suitable use of the relevant commands produce the kind of witness terms that are needed [4].

The reader should note that our purpose is not to discover or reimplement various algorithms for CAD, QE, etc., but rather to use them for the improvement of automated reasoning engines.

5 Extraction of Witness Terms

The pre-processing of input formula consist of the following steps:

1) eliminate terms containing function symbols by introducing new variables,
2) partition the quantifier prefix into alternating quantifier blocks,
3) mark the existentially and universally quantified variables and delete the quantifiers in the outermost block.

These are the steps of the post-processing of the CAD-expression produced by the computer algebra algorithm:

1) transform the CAD into disjunctive normal form,
2) check projection conditions,
3) isolate the constraints for the originally existentially quantified variables,
4) choose a sample value for each existential variable satisfying the constraints.

We demonstrate the steps through another simple example from elementary analysis:
Prove that *if the real valued sequences f and g are convergent, then so is their sum, $f \oplus g$* . We did not repeat the formal definition of convergence, but refer immediately to the first two conjectures which were generated by our implementation. The initial proof situtation before calling the conjecture

[1] http://www.wolfram.com/
[2] http://www.cs.usna.edu/~qepcad/B/QEPCAD.html
[3] http://www.fmi.uni-passau.de/~redlog/
[4] We kindly acknowledge the thorough help offered by Thomas Sturm for understanding and using the REDLOG system.

generator contained only the main goal and the formulae for the defintion of convergence and for the operator $\oplus$:

Conjecture 1:

$$\forall_{a_1,a_2}\exists_{a_0}\forall_{\epsilon_0}\exists_{\epsilon_1,\epsilon_2}\forall_{x_1,x_2}(|x_1-a_1|<\epsilon_1 \wedge |x_2-a_2|<\epsilon_2) \Rightarrow |(x_1+x_2)-a_0|<\epsilon_0$$

Conjecture 2:

$$\forall_{N_1,N_2}\exists_{N_0}\forall_n(n \geq N_0 \Rightarrow n \geq N_1 \wedge n \geq N_2)$$

Example 3: Generated conjectures (Convergence of Sum)

As a consequence of the decomposition strategy, in the first conjecture every variable ranges over the reals, in the second conjecture all variables range over the naturals. Attacking the first conjecture with the real extended quantifier elimination algorithm yields (besides the validity of the formula) the instances: $a_0 \leftarrow a_1+a_2, \epsilon_1 \leftarrow \frac{\epsilon_0}{2}, \epsilon_2 \leftarrow \frac{\epsilon_0}{2}$ (Example 4).

$$\forall_{\epsilon_0}\exists_{\epsilon_1,\epsilon_2}\forall_{x_1,x_2}(|x_1-a_1|<\epsilon_1 \wedge |x_2-a_2|<\epsilon_2) \Rightarrow |(x_1+x_2)-a_0|<\epsilon_0$$

$$\rightarrow_{QE} \quad a_0 = a_1+a_2$$

$$\forall_{x_1,x_2}(|x_1-a_1|<\epsilon_1 \wedge |x_2-a_2|<\epsilon_2) \Rightarrow |(x_1+x_2)-(a_1+a_2)|<\epsilon_0$$

$$\rightarrow_{QE} \quad 0<\epsilon_1<\epsilon_0 \wedge 0<\epsilon_2 \leq \epsilon_0-\epsilon_1$$

Example 4: Pre-processed formulae and conditions for the witnesses

The validity of the second formula can be similarly checked by integer QE and the provided instance for N_0 should be greater than the maximum of N_1 and N_2. We think that our method is suitable for handling the following problem types:

- boundedness of real valued sequences
- convergence of real valued sequences
- continuity of real valued functions
- uniform continuity of real valued functions

In particular one can investigate with the method the properties of concrete sequences or functions (e.g. *continuity of* $f[x]=x^2$ or *the convergence of the sequence* $a[n]=1/n$), the interaction of the notions below with the standard operators (e.g., *convergence of the product*) or interactions of several notions (

e.g., *the product of a null-convergent and a bounded sequence is null-covergent*). It is particularly interesting to investigate the problem of the *uniform continuity of the product* with the method. In the first attempt the method fails to prove that the product of two uniformly continous functions is uniformly continuous. Since the method is not complete, we cannot consider this as disproving the original conjecture, but only as a hint that the conjecture might be incorrect. Additionally, considering only products of linear polynomials now, we can actually find concrete linear polynomials whose product is not uniformly continuous.

6 Conclusion and Future Work

We combine logical and algebraic techniques in order to support efficient reasoning in natural style for elementary analysis. Using logical decomposition strategies, we reduce the initial proof situations to problems which can be handled by real quantifier elimination.

In particular, on the basis of Cylindrical Algebraic Decomposition, we are able to find real witness terms for difficult instantiations. Problems similar to real EQE can occur if all the variables range over the integers or if we have a problem which involves reals and integers as well. The solution of the (extended) quantifier elimination problem in the latter cases is only partially supported by the existing systems (see e.g., [14]). The integration of those algorithms to a reasoning sytem would significantly extend its capabilities.

References

[1] Arnon-Collins-McCallum: Cylindrical Algebraic Decomposition I: The Basic Algorithm. In: Caviness-Johnson (eds): Quantifier elimination and cylindrical algebraic decomposition (pp. 136-151), Springer 1998.

[2] C. W. Brown- S. McCallum: On using bi-equational constraints in CAD construction. In: Proceedings of the 2005 international symposium on Symbolic and algebraic computation, Beijing, China pp. 76 - 83.

[3] B. Buchberger: The PCS Prover in Theorema. In: Lecture Notes in Computer Science (LNCS) 2178, pp. 19-23, Springer Verlag, Berlin, 2001.

[4] G. E. Collins: Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In: Caviness-Johnson (eds): Quantifier elimination and cylindrical algebraic decomposition (pp. 85-121), Springer 1998.

[5] A. Dolzmann-T. Sturm-V. Weispfenning: Real Quantifier Elimination in Practice. MIP-9720, Universität Passau, December 1997, Algorithmic Algebra

and Number Theory, Springer 1998, Matzat, B. H. and Greuel, G.-M. and Hiss, G. (ed.), pp. 221-247.

[6] T. Jebelean. Natural Proofs in Elementary Analysis by S-Decomposition. Technical report no. 01-33 in RISC Report Series, University of Linz, Austria. November 2001.

[7] G. Kreisel-J.L. Krivine: Modelltheorie. Springer, Berlin, 1972

[8] R. Loos and V. Weispfenning: Applying linear quantifier elimination. The Computer Journal, 36(5):450-462, 1993.

[9] D. Marker: Model Theory: An Introduction, Springer, Berlin, 2002.

[10] J. Schicho - A. Tesacek: Improved Projection Operator for CAD using Groebner Bases Technical report no. 99-04 in RISC Report Series, University of Linz, Austria. 1999.

[11] A. Seidl: Extending Real Quantifier Elimination by Cylindrical Algebraic Decomposition to Get Answers. In V. G. Ganzha, E. W. Mayr and E. V. Vorozhtsov, editors, Proceedings of the Seventh International Workshop on Computer Algebra in Scientific Computing (CASC 2004), St. Petersburg, Russia.

[12] Tie-Cheng Wang-W.W. Bledsoe: Hierachical Deduction. in: JAR 3(1), 1987, pp. 35-78

[13] V. Weispfenning: Quantifier elimination for real algebra the quadratic case and beyond Applicable Algebra in Engineering, Communication and Computing, 1993

[14] V. Weispfenning: Mixed Real-Integer Linear Quantifier Elimination. ISSAC 1999: pp. 129-136

[15] F. Winkler: Polynomial Algorithms in Computer Algebra. Texts and Monographs in Symbolic Computation. Springer, 1996.