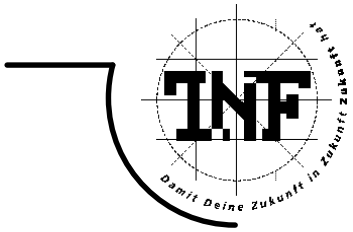
JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis



Supporting Exploration in Elementary Analysis by Computational, Graphical and Reasoning Tools

Dissertation

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

Angefertigt am *Institut für Symbolisches Rechnen*

Betreuung:

Univ.-Prof. Dr.phil.DDr.h.c. Bruno Buchberger
Univ.-Prof. Mag. Dr. Wolfgang Schlöglmann

Beurteilung:

Univ.-Prof. Dr.phil.DDr.h.c. Bruno Buchberger
Univ.-Prof. Mag. Dr. Wolfgang Schlöglmann

Eingereicht von:

Robert Vajda

Linz, Juni 2009

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Linz, Juni 2009

Robert Vajda

Abstract

The present thesis

- investigates the requirements for a *computer supported learning environment for elementary analysis* and
- it contributes to the *development* of such an environment.

An ideal environment would be able to provide computer support for any phases of the mathematical exploration. It turns out that existing known mathematical assistant systems, like general-purpose computer algebra systems provide efficient and user-friendly *graphical* and *computational* tools for the phase of experimentation and application. So, they should be just adjusted to the specials needs of a particular mathematical field like analysis.

However, the essential part of mathematics, i.e. *reasoning*, is rarely covered yet by assistant systems. Therefore, in the focus of this thesis is the *computer support for proving problems occurring in elementary analysis*. The main contribution of the author is the extension of the combined techniques for proving standard theorems in elementary analysis: the combined methods link logical and algebraic techniques.

The algebraic techniques provide instantiations on existentially quantified variables in the goal formula, when finding appropriate witnesses for the existentially quantified variables within a purely logical calculus is difficult and inefficient. Based on *the PCS-method* and *the S-Decomposition strategy* and exploiting the method of *quantifier elimination*, *reasoners for elementary analysis* are designed and implemented in the Theorema proving system. Several deduction examples are provided by the author in the case studies for elementary analysis, putting an emphasis also on didactical issues during the exposition.

Keywords: computer supported learning environment, computer algebra, automated reasoning, elementary analysis, quantifier elimination, Mathematica, Theorema

Zusammenfassung

Diese Dissertation

- untersucht die Bedingungen und Forderungen für eine *Computerunterstützte Lernumgebung für elementare Analysis* und
- leistet eine Beitrag zu der *Entwicklung* solcher Lernumgebung.

Eine ideale Umgebung wäre es fähig, jede Phase einer mathematisches Exploration zu unterstützen. Es stellt sich heraus, dass gut bekannte mathematische Assistantsysteme wie generelle Computeralgebra Systeme bieten effiziente und benutzerfreundliche graphische und rechnerische Tools für die Phasen von Experimentieren und Anwendung. Diese Tools sollen nur die spezifische Ansprüchen eines mathematischen Gebiet wie Analysis angepasst werden.

Jedoch der wichtigste Teil der Mathematik, d.h. *Beweisen* wird noch nur selten unterstüzt durch Assistantsysteme. Deshalb setzt die Dissertation den Akzent auf die Computer-Unterstützung von Beweisproblemen, die in elementarer Analysis auftauchen. Der Author leiset einen Beitrag zu diesem Gebiet mit der Erweiterung der kombinierten Methoden für automatisches Beweisen standard Theorems von elementarer Analysis. Die kombinierte Methode knüpfen algebraische und logische Techniken zusammen.

Basierend auf das *PCS-Method* und die *S-Decomposition* Strategie und Quantorelimination, Provers für elementare Analysis wurde ausgelegt und entwickelt in dem Rahmen des Theorema Beweissystems. Es wird mehrere Deduktionsbeispiele gegeben in einem Case Study. Auch die didaktische Aspekte werden während der Darstellung betont.

Schlüsselwörter: Computerunterstützte Lernumgebung, Computeralgebra, Automatisches Beweisen, Elementare Analysis, Quantorelimination, Mathematica, Theorema

Contents

Chapter 1

Introduction

Motto:*[I]ntuition comes to us much earlier and with much less outside influence than formal arguments which we cannot really understand unless we reached a relatively high level of logical experience and sophistication.[...] In the first place, the beginner must be convinced that proofs deserve to be studied, that they have a purpose, that they are interesting.*[Poly65, Vol 2, p.128]

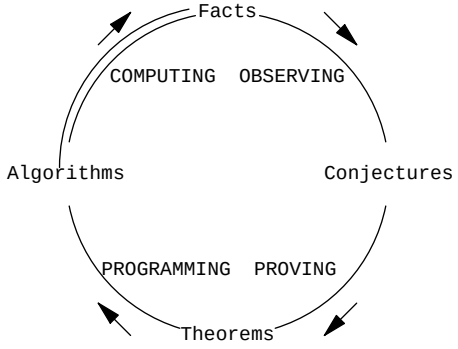
For a starting point of this thesis, I claim that computer algebra systems (CASs) found their way to mathematics education. Their role for supporting mathematical modeling are accepted. Their computational power extended with visual tools and a user-friendly human-machine interface made them attractive even for students and teachers. *However, most CAS lacks of support for proving and this thesis focuses on this issue.* To understand the place and role of proving in mathematics, we would like to emphasize two aspects from Polya's thought from above:

(1) Convenient environments for experimentation are needed in order to to gain insight regarding mathematical objects, their properties and relations, before we can start with formalization and rigorous proofs. These learning environments are now realizable on computers and the availability of such environments makes a difference, since the development of mathematical ability is notoriously dependent upon 'doing' rather than upon 'being told' or 'remembering'.

(2) The teaching of mathematics can reach its own final goal only if students detect, when a statement or a conjecture – discovered in the experimentation phase –, merits a proof and they understand how a proof is to be done. This means that we cannot derogate the role of proofs in the math curriculum. Exploration and proof have been and remain complementary [Kort04].

To embed the two opposite aforementioned aspects of mathematics into a broader model and think about them further, we invoke Buchberger's *creativity spiral* [Buch98]. Buchberger underlines that the essence of mathematics is *proving*, i.e., mathematics is not really mathematics without rigorous proofs [Buch97a]. However, this does not mean that he reduces mathematics to proving. For getting new insights and ideas for proving, we do in fact a lot more.

A general scheme for sorting different mathematical activities along the timeline is offered by the creativity spiral. In this model, mathematical exploration has a structure of a spiral. The spiral structure expresses both circularity and evolution. In each cycle of the spiral one starts with examples. Observing several examples one can detect patterns, dependencies and relationships among the important quantities. This is more likely to happen if one designs experiments carefully: the sequence of examples are generated under the control of certain quantities. The observed relations can be expressed in form of general conjectures. By proving, the conjectures can become validated propositions, i.e. theorems. Theorems build a basis for systematic problem solving. After transforming the content of the theorems into algorithms, the latter can be applied to real or even to fictious data. At the end of a cycle, the application of mathematical knowledge can generate new insights, problems and data. Thus, observation and experimentation may continue in a higher level [Buch98, HKL96, Kutz00].



One fruitful way of teaching mathematics is to follow the path of the discovery and exploration. Therefore we can learn much from the above model; In fact, the model of the creativity spiral is especially applicable for computer supported mathematics education [HKL96]. The three phases of the exploration cycle (experimentation, proving, application) warn us that the teaching of mathematics cannot neglect any phase of the cycle, and the same holds if we want to develop a computer supported learning environment.

Besides the above spiral model, Buchberger's *White Box/Black Box Principle* [Buch90] is also used as an important guideline in the development of the learning environment and educational units presented in the thesis. The thesis mainly focuses on the *proving phase*, because computer support for that is rarely present in math education yet. In addition, the thesis will also touch briefly the application of existing graphical and computational tools available in CA systems for supporting the experimentation and application phase in the field of elementary analysis.

One crucial point regarding the application of graphical and computational tools in a computer supported learning environment is an appropriate user in-

terface. To make the description simpler, only tools for the experimental phase are considered. This is justified here, because the target user of the units is a freshman calculus student and the emphasis is put on concept development in most of the units. The interaction of the students with computer happens through standard graphical user interface elements (adjusting sliders, clicking on buttons or points by mouse) or typing and evaluating simple commands in an interactive notebook. Hence the tools are adaquate also for those students, who do not possess deep programming skills or system-specific syntactical preknowledge. Although the tools are based on the basic services of CA sytems, they are not identical with them: they represent a higher level. The tools in the experimenting phase focus either on typical mathematical models or representations and the switching back and forth between them, if there are equivalent ones. Or, they focus on a property and provide several instances for testing the property. With those tools, the learner gets sufficient basis for anchoring the abstract and general notions and structures of typical mathematical definitions to concrete instances. Some of the tools applied in a computer lab is particulary suitable for organizing small group activities and provoking discussions about the interpretation of the outputs of the tools. After the experiments, the learner should be able to make observations and remarks about the investigated phenomena, even then, if they can do that only in a rough lingustic form. To each tool a clear operational goal and the mode and limitations of its usage has to be assigned by the instructor. Even the limits and the imperfection of the tools in use might lead the learner to the need for a new phase where the observations can be made more precise and convincing arguments for the formulated observation can be given.

Classical CA systems do not contain tools for reasoning. However, CAS is not the only mathematical assistant system (MAS). Reacting to the needs of theoretical computer science and mathematics, automated deduction systems, specific constraint solving systems etc. were developed as well. They are certainly not so well known and integrated to the math education as CAS, which has several reasons. We will address the difficulties in a later section.
The proving phase is not only one of the exploration phases, but rather it has a distinguished role [Buch00b], thus, it is obvious, that a general purpose MAS, or a new learning environment for mathematics should integrate also reasoning-supporting capabilities. The present thesis claims that such *new learning environments with reasoning capabilities are reasonable, realizable and it makes a contribution how this is to be achieved*. Instead of building a learning environment from scratch, the author exploited the professional services of a well-known CAS, Mathematica [Wolf99], and extended it with reasoning tools.

From the learners point of view, the provers provided in the learning environment enable the study of proof presentation in a natural style. The learner specifies formally the proof goal and the knowledgebase using the language of logic before the prover call. In the learning units, templates and prototype formulae are offered for entering typical proving problems. The prover attempts to deduce the goal formula using the formulae in the knowledgebase by specific inference rules automatically. The result (proof object) is shown as a self-contained

interactive document, too. The learners can investigate the overall structure of a deduction or the particular inference rules. By failing proof attempts, they are urged to investigate the reasons for failure, modify the goal formulae or extend the knowledgebase with further formuale. Moreover, they could adjust the granularity of inference steps, if desired.

In many notions of elementary analysis such as boundedness, limits, continuity etc., an alternating sequence of universal and existensial quantifiers occurs in the corresponding formal definition. That makes even the automation of the elementary standard theorems related to these notions difficult. The main contribution of the author is the *extension of the combined techniques for proving standard theorems in elementary analysis*: the combined methods link logical and algebraic techniques. The algebraic techniques will provide instantiations on existentially quantified variables in the goal, when the finding of appropriate witnesses for the existentially quantified variables within a purely logical calculus is difficult and inefficient. Based on the PCS-method [Buch01, Dupr00] and the S-Decomposition strategy [Jebe01] and exploiting the method of quantifier elimination [ACC98], *reasoners for elementary analysis are designed and implemented* in the Theorema proving system [Buch00a]. Several deduction examples are provided by the author in the case studies for elementary analysis, putting an emphasis also on didactical issues during the exposition.

The author collected many insights during his work in the frame of the CreaComp project [MSW07, Vajd09b, Wind07] at the Johannes Kepler University of Linz, directed by Bruno Buchberger, Erich Peter Klement and Günter Pilz. The project aimed at developing interactive mathematical courseware for university students in several topics: Elementary Analysis, Groebner Bases, Fuzzy Logic, Set Theory etc. The author of this thesis was responsible for developing educational units for elementary analysis.

The thesis is organized as follows. Chapter 2 introduces a possible frame for proving in natural style. Chapter 3 reviews the literature on quantifier elimination. Chapter 4 contains the main results of the thesis. It explains how the algebraic techniques for quantifier elimination can be used for obtaining witnesses for deduction and describes the combination of the algebraic and logical techniques and the implementation of the combined method. Chapter 5 is a case study on using these tools for the exploration of notions in elementary analysis. Chapter 6 reports about related work in the field.

Chapter 2

Reasoning

2.1 Deduction Systems

The hope to be able to support reasoning also in the field of mathematics education (at least at certain subfields) originates from the following facts: For the general, first order predicate logic one can construct adequate (both correct and complete) formal deduction systems. Practically relevant first order theories, like Algebraically Closed Fields (ACF), Real Closed Fields (RCF), Presburger Arithmetic(PA), etc. admit quantifier elimination and/or are decidable. The main theoretical results mentioned above were mostly known already in the 60's, but those alone would still not suffice. Only with the development of efficient symbolic algorithms, the speed of current hardwares and the growing interest at observing how proofs are done by human (experts) in daily routine, even if this is a domain specific heuristic, make it possible to envision a mathematical e-learning environment where reasoning is also supported by machine.

We already mentioned that automated reasoning systems were implemented and successful in mathematical research, especially in abstract algebra (model checking, model construction, axiomatization), but they never really entered math education. The first reason is that they worked in a restricted set of formulae, namely on clauses. The inference rules of the systems were simple (resolution extended with paramodulation) and hence really suitable for automation, but as a consequence of this, the generated proofs (proof traces) were typically long and not human readale (digestable). So we have to take into consideration these results, but should choose better suited formal calculus for imitating human inferencing, i.e., if we want to prove in a natural style. One possible choice for such a system is a variant of Gentzen sequent calculus. Gentzen expressed the goal of his investigation as follows:

Ich wollte zunächst einmal einen Formalismus aufstellen, der dem wirklichen Schließen möglichst nahe kommt. So ergab sich ein "Kalkül des natürlichen Schließens [Gent34]

In the sequel we summarize the inference rule of a core calculus, explain how

the deduction works, give an example of a deduction and afterwards discuss what can make a proof system 'natural'. It will turn out that there is a lot of points where one must make a choice and we will also slightly deviate from this calculus.

2.2 G-calculus: Framework for Reasoning

Let X, Y denote arbitrary formulae, Γ, Δ sets of formulae, t an arbitrary term, a is a (fresh) variable. The next table shows the inference rules of the calculus:

$$\frac{\langle \Gamma, \Delta \cup \{X\}\rangle, \langle \{Y\} \cup \Gamma, \Delta\rangle}{\langle \{X \Rightarrow Y\} \cup \Gamma, \Delta\rangle} \qquad \frac{\langle \Gamma \cup \{X\}, \{Y\} \cup \Delta\rangle}{\langle \Gamma, \{X \Rightarrow Y\} \cup \Delta\rangle}$$

$$\frac{\langle \Gamma \cup \{X, Y\}, \Delta\rangle}{\langle \Gamma \cup \{X \wedge Y\}, \Delta\rangle} \qquad \frac{\langle \Gamma, \Delta \cup \{X\}\rangle, \langle \Gamma, \Delta \cup \{Y\}\rangle}{\langle \Gamma, \{X \wedge Y\} \cup \Delta\rangle}$$

$$\frac{\langle \Gamma \cup \{X\}, \Delta\rangle, \langle \Gamma \cup \{Y\}, \Delta\rangle}{\langle \Gamma \cup \{X \vee Y\}, \Delta\rangle} \qquad \frac{\langle \Gamma, \{X, Y\} \cup \Delta\rangle}{\langle \Gamma, \{X \vee Y\} \cup \Delta\rangle}$$

$$\frac{\langle \Gamma, \{X\} \cup \Delta\rangle}{\langle \{\neg X\} \cup \Gamma, \Delta\rangle} \qquad \frac{\langle \Gamma \cup \{X\}, \Delta\rangle}{\langle \Gamma, \{\neg X\} \cup \Delta\rangle}$$

$$\frac{\langle \{\phi[t/x]\} \cup \{\underset{x}{\forall}\ \phi\} \cup \Gamma, \Delta\rangle}{\langle \{\underset{x}{\forall}\ \phi\} \cup \Gamma, \Delta\rangle} \qquad \frac{\langle \Gamma, \{\phi[a/x]\} \cup \Delta\rangle}{\langle \Gamma, \{\underset{x}{\forall}\ \phi\} \cup \Delta\rangle}$$

$$\frac{\langle \{\phi[a/x]\} \cup \Gamma, \Delta\rangle}{\langle \{\underset{x}{\exists}\ \phi\} \cup \Gamma, \Delta\rangle} \qquad \frac{\langle \Gamma, \{\phi[t/x]\} \cup \{\underset{x}{\exists}\ \phi\} \cup \Delta\rangle}{\langle \Gamma, \{\underset{x}{\exists}\ \phi\} \cup \Delta\rangle}$$

Inference rules of the G sequent system:
6 rules for manipulating the left component ($L1 - L6$)
and 6 for the right component ($R1 - R6$)

We would like to emphasize the following characteristic properties of this proof system:

- It models a proof-problem or the initial proof-situation with a pair $\langle \Gamma, \Delta \rangle$. Γ and Δ is a finite set of (well formed) formulae of the language. The first(left) component will contain premises (known or assumed formulae) and the second (right) component will correspond to the goal formula. The second set contains typically only one formula ($\Delta = \{Y\}$), but for assuring symmetry in the calculus, more formulae can occur also in second component of the pair. We will usually refer to the components briefly as (current) knowledge base and goal: $\langle \mathrm{KB}, \mathrm{G} \rangle$.

- If we are able to 'solve' a proof problem, i.e. deduce a proposition Y ($\mathrm{G} = \{Y\}$) relative to set of premises $\mathrm{KB} = \{X_1, \cdots, X_n\}$ in this calculus, this will correspond to the fact that the formula $\wedge_i X_i \Rightarrow Y$ is a tautology (semantics). With other notations, $X_1, \cdots, X_n \vdash Y$. In the general case, when $\mathrm{KB} = \{X_1, \cdots, X_n\}$ and $\mathrm{G} = \{Y_1, \cdots, Y_m\}$, the deduction sould go through iff the formula $(\wedge_j X_j) \Rightarrow (\vee_j Y_j)$ is valid (note the disjunction on the right hand side of the implication).

- If we attempt to deduce a proposition relative to set of premises, we will build up a deduction tree rather than constructing a sequence of formulae (in contrast to deduction in a Hilbert-type or in a resolution calculus). A proof situation will be transformed to one or several new proof situations by inference rules.

- For each logical quantifier and operator there will be a corresponding inference rule both for the knowledge base and for the goal. Those inference rules try to capture naturally such proving techniques like case disctinction ($L3$), conditional proving ($R1$), etc. One can consider every inference rule as an attempt for reduction of the current proof situation to simpler ones.

- We finished a deduction if all branches of the proof-tree are closed, i.e., if all the leaves of the proof tree are axioms (positive terminating proof situations). To be more specific here, *axioms* are the following sequents (and only these): $\langle \Gamma \cup \{X\}, \{X\} \cup \Delta \rangle$

In the next example we demonstrate how the calculus works.

2.2.1 Example

Let us investigate the following inference[1] in G. There are two premises (P1, P2) and the conclusion is denoted by C.

P1 : No French sportsman qualified for the finals.

P2 : Denise is a french sportman.

C : There is a sportman who did not qualify for the finals.

[1]Example taken from the famous Hungarian elementary logic book 'Element of Logic' by I. Ruzsa and L. Pólos

Introducing suitable unary predicate symbols and a constant, we have (S-sportsman, F-French, Q-qualified for the finals, d-Denise):

$$\text{P1} : \neg(\exists_x(S[x] \wedge F[x] \wedge Q[x]))$$

$$\text{P2} : S[d] \wedge F[d]$$

$$\text{C} : \exists_x(S[x] \wedge \neg Q[x])$$

The initial proof situation is as follows: $\langle\{\text{P1}, \text{P2}\}, \{\text{C}\}\rangle$. The deduction should be read from bottom up:

$$\langle\{S[d],\ \mathbf{F}[d],\ Q[d]\},\ \{\mathbf{F}[d], (2),\ (1)\}\rangle, \langle\{S[d],\ F[d],\ \mathbf{Q}[d]\},\ \{\mathbf{Q}[d], (2),\ (1)\}\rangle$$

$$\frac{\langle\{\mathbf{S}[d],\ F[d],\ Q[d]\},\ \{\mathbf{S}[d], (2),\ (1)\}\rangle,\quad \overline{\langle\{S[d],\ F[d],\ Q[d]\},\ \{F[d] \wedge Q[d], (2),\ (1)\}\rangle}}{\overline{\langle\{S[d],\ F[d], Q[d]\},\ \{S[d] \wedge F[d] \wedge Q[d], (2),\ (1)\}\rangle}}$$

$$\frac{\overline{\langle\{S[d],\ F[d], Q[d]\},\ \{\exists_x\ (S[x] \wedge F[x] \wedge Q[x])^{(2)},\ (1)\}\rangle}}{\overline{\langle\{S[d],\ F[d]\},\ \{\neg Q[d],\ \exists_x\ (S[x] \wedge F[x] \wedge Q[x]),\ (1)\}\rangle}^{*}}$$

$$\frac{\langle\{\mathbf{S}[d],\ F[d]\},\ \{\mathbf{S}[d],\ \exists_x\ (S[x] \wedge F[x] \wedge Q[x]),\ (1)\}\rangle\ ,\quad \langle\ (\text{II})\ \rangle^{*}}{\overline{\langle\{S[d],\ F[d]\},\ \{S[d] \wedge \neg Q[d],\ \exists_x\ (S[x] \wedge F[x] \wedge Q[x]),\ \exists_x\ (S[x] \wedge \neg Q[x])^{(1)}\}\rangle}}$$

$$\frac{\overline{\langle\{S[d],\ F[d]\},\ \{\exists_x\ (S[x] \wedge F[x] \wedge Q[x]),\ \exists_x\ (S[x] \wedge \neg Q[x])\}\rangle}}{\overline{\langle\{S[d] \wedge F[d]\},\ \{\exists_x\ (S[x] \wedge F[x] \wedge Q[x]),\ \exists_x\ (S[x] \wedge \neg Q[x])\}\rangle}}$$

$$\frac{}{\overline{\langle\{\neg\exists_x\ (S[x] \wedge F[x] \wedge Q[x]),\ S[d] \wedge F[d]\}, \{\exists_x\ (S[x] \wedge \neg Q[x])\}\rangle}}$$

Deduction in the G calculus

Due to space constraints, we show the deduction tree in two parts. The parts are linked through the proof situation labeled by (II) and shown in blue. At the bottom of the deduction-tree, we find the initial proof situtation. First the rule $L4$ and $L2$ will be applied. At the top of the tree, four leaves of the tree (printed in red) contain terminating proof-situations: Those ground atomic formulae are boldfaced in the leaves which occur in both sides of the proof situation. Some formula is labeled and is not being carried through the whole deduction. All branches of the deduction-tree are closed.

Pelletier [Pell99] claims that not one specific, but several properties of a proof system determines whether the system can be considered to be a *natural deduction system*. We think that it is very important that system gives a direct method, i.e., not a refutational calculus. It does not restrict the formulae appearing in the deduction to any normal forms and it the original language is rich enough (does not exclude from the syntax such symbols as implication, existential quantifier etc.) The basic inference rules are close to the elementary steps of routine reasoning. It is also important to mention that there is another

aspect that we need to address, if we want to come up with computer support for reasoning. This aspect is the *suitability for automation* and obviously, it is sometimes contradictory to "naturalness". It is worth to consider the rules of G once more. Every inference rule is invertible, so we have a freedom in which order we apply the rules for deduction. In particular, it means that we cannot be stuck in a proof situation if the original sequent (the formula corresponding to the initial sequent) is valid.

2.2.2 Modifications

We will deviate from this system slightly and we give reasons for that:

- We handle equivalences and equalities $(\Leftrightarrow, =)$ in a special manner. The logical operator $\Leftrightarrow$ is not part of the language described in this section and there is no rule in the G calculus for handling equivalences. However, there would be a straightforward way to use the logical equivalence $\phi \Leftrightarrow \psi \equiv (\phi \Rightarrow \psi \wedge \psi \Rightarrow \phi)$ for eliminating $\Leftrightarrow$. Instead of this, we will often treat formulae with universal equivalence as definitions of predicate symbols and we try to exploit rewriting for unfolding definitions. In a similar way, universal equalities can stand for definition of terms, or identities between terms [Dupr00, Section 3].

- Since we "model" a typical mathematical proof problem where we have finitely many premises and only one goal formula, we break up the symmetry regarding left and right rules and work with only one formula on the RHS. Thus, some rules, which generate more than one formula on the RHS will be changed. For instance, let us consider rule R3 of the G calculus. It is clear that even if the goal formula is the simplest disjunction $(\Delta = \{A_1 \vee A_2\})$, the rule R3 produces two formulae in the second component. One way to avoid this is to replace R3 with a new inference rule R3':

$$\frac{\langle \Gamma \cup \{\neg A_1\}, \{A_2\}\rangle}{\langle \Gamma, \{A_1 \vee A_2\}\rangle}$$

- We may use implication $(\Rightarrow)$ for imitating forward and backward chaining.

- Instantiation in practice is difficult (see L5 and R6): We want to exploit not only purely logical, but rather algebraic techniques for specific domains. An algebraic technique may be able to check/decide the validity of certain quantified formulae and even provide existential witness term for the instantitation. Thus, as a modification of rule R6, usually a new metavariable will be introduced where the metavariable stands for a yet unknown term which will be provided by the algebraic techinque. For better reading, we denote the metavariables with a star (*) in the superindex. Compare R6 with the new rule R6'.

$$R6: \quad \frac{\langle \Gamma, \{P[t]\}\rangle}{\langle \Gamma, \{\underset{x}{\exists}\ P[x]\}\rangle}$$

$$R6': \quad \frac{\langle \Gamma, \{P[x^*]\}\rangle}{\langle \Gamma, \{\underset{x}{\exists}\ P[x]\}\rangle}$$

Rules R6 and R6' .

In the modified rule instantiation is postponed. Suitable value for the metavariable x^* is provided by algorithmic reasoning based on algebraic techniques. Algebraic techniques for algorithmic reasoning will be discussed in the next section.

Chapter 3

Algebraic Techniques

3.1 Introduction, Motivation

To support practical reasoning in the field of elementary analysis, a pure logical calculus is not efficient, because, in many proof problems, a crucial step in the deduction is to find elements from a domain which satisfy certain algebraic constraints. One may think as simple instances of the problem to find a limit of a rational function at infinity, or to find an upper bound of a sequence. A pure logical calculus would be able to solve such a problem, if knowledge about the underlying domains would be stored in a lemmata system (in a declarative form). The basic technique for a reasoner is then to match (unify) the current goal formula against the formulae in the knowledge base.
Problem 1 (problem involving reals)

Find ϵ_1, ϵ_2 such that $\epsilon_1 > 0 \wedge \epsilon_2 > 0 \wedge \epsilon_1 + \epsilon_2 = 1$.
Knowledge base:
(l1) $1 > 0$
(l2) $2 > 0$
(l3) $\underset{x}{\forall}\ x/2 + x/2 = x$

(l4) $\underset{x,y}{\forall}\ (x > 0 \wedge y > 0) \Rightarrow x/y > 0$

Using (l3), (since the third conjunct of the constraint in the goal matches (l3)), derive (l3a) $1/2 + 1/2 = 1$ (extension of the knowledge base). For proving the positivness of the term $1/2$, use (l4) (backchaining) and finally close the remaining branches by (l1) and (l2). Conclude that the substitutions $\epsilon_1 \leftarrow \frac{1}{2}, \epsilon_2 \leftarrow \frac{1}{2}$ solve the constraint problem. Since we want to solve more sophisticated constraint problems (see the next example) which involve

- many variables, not only two,
- parameters,
- not only linear conditions

- even quantified variables,

this approach is not efficient. The disadvantage of this approach is that we have to search for adequate formulae in the knowledge base which maybe only partially match the constraint goal and additionally there is no guarantee that knowledge about the underlying domain (e.g, about the arithmetic of the reals) is stored in the desired form.

As an alternative, we will combine the deduction step of the logic reasoner with constraint solvers. With the access to constraint solvers knowledge about the underlying domains (typically, reals and integers) available in an algorithmic form. In the first problem, since all constraints are linear, we could use an algebraic method similar to the Gaussian elimination. The Fourier-Motzkin method can solve the system of inequalities in the form $Ax \leq b$ ($A \in \Re^{m \times n},\ b \in \Re^m$ are given). After possible normalization we could achieve that in all inequalities the first variable ϵ_1 occurs with coefficient 0, +1 or -1.

$$\begin{array}{rcrrcrr} (-1) & \epsilon_1 & + & (0) & \epsilon_2 & < & 0 \\ (0) & \epsilon_1 & + & (-1) & \epsilon_2 & < & 0 \\ (1) & \epsilon_1 & + & (1) & \epsilon_2 & \leq & 1 \\ (-1) & \epsilon_1 & + & (-1) & \epsilon_2 & \leq & -1 \end{array}$$

We can work with the coefficient matrix:

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 1 & 1 & 1^* \\ -1 & -1 & -1^* \end{pmatrix}$$

Now we partition the inequalities according to the signs of ϵ_1.

$$\left(\begin{array}{rrr} -1 & 0 & 0 \\ -1 & -1 & -1^* \\ \hline 0 & -1 & 0 \\ \hline 1 & 1 & 1^* \end{array}\right)$$

We combine the rows which contain -1 and +1 in each possible way. Doing so, the first variable will be eliminated and we remain with an equivalent system, in the sense that the original inequality system has a solution iff the new has one:

$$\begin{pmatrix} 0 & -1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

Doing similar steps, after only one variable remained, we collect the best bounds for the last variable. If the contraints are consistent on the last variable, we select a sample solution. Here, $0 < \epsilon_2 < 1,\ \epsilon_2 \leftarrow \frac{1}{2}$. The method in fact eliminates a variable by introducing new constraints on the set of the remaining variables. The initial constraints are solvable iff the constraints which do not contain the main variable are solvable and the maximum of the lower bounds on the main

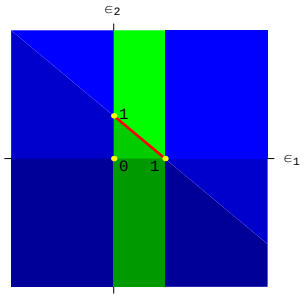
variable less or equal then the minimum of the upper bounds on the main variable.

As an alternative to the Fourier-Motzkin method, we could use for Problem 1 the general Collins' Cylindrical Algebraic Decomposition algorithm, which can handle not only linear, but general polynomial constraints over the reals as well (e.g., we could have instead of $\epsilon_1 + \epsilon_2 = 1$ the condition $\epsilon_1^2 + \epsilon_2^2 = 1$). Since this will play a central role in the present thesis, we give a sketch how this approach would solve the constraint Problem 1, although applying it here is like "shooting with canons on sparrows" .

Collins' decomposition method, after reducing all equalities and inequalities to zero, filters out from the constraint system multivariate polynomials and based on the sign evaluation of those polynomials, it decomposes the ϵ_1, ϵ_2-plane into finitely many, connected, sign-invariant cells. We introduce the details in Appendix A, here it suffices to say that the polynomials in Problem 1 are

$$p_1 = \epsilon_1,\ p_2 = \epsilon_2,\ p_3 = \epsilon_1 + \epsilon_2 - 1,\ p_4 = \epsilon_1 - 1,$$

and the decomposition is illustrated in the next picture. The cells of the decompositions can be points, line segments and two-dimensional subsets as well. The picture shows that the points of only exactly one cell (red) satisfies all constraints and therefore there is a solution of Problem 1 and as a sample solution one could pick up any point from the cell, e.g. $(\frac{1}{2}, \frac{1}{2})$ is just fine.



Problem 2

Find ϵ_1, ϵ_2 such that $\epsilon_1 > 0 \wedge \epsilon_2 > 0 \wedge \forall_{x,y}(|x| < \epsilon_1 \wedge |y| < \epsilon_2) \Rightarrow |x+y| < \epsilon_0$, assuming that ϵ_0 is a positive parameter.

Solution: $\epsilon_1 \leftarrow \epsilon_0/2, \epsilon_2 \leftarrow \epsilon_0/2$.

The main technique for handling quantified algebraic constraints over the reals or the integers is (effective) *quantifier elimination* (QE). However the historical development of QE is independent from the practical, (quantifier-free) constraint problems and it originates from the model theory, where the existence of quantifier-free equivalents over some model class has important theoretical consequences [Stur02, Weis06]. To be self-contained, in the next subsection we briefly review the main notions and results of model theory regarding quantifier elimination and afterwards discuss how we can extend those techniques for solving quantified constraints which we are interested in.

3.2 Quantifier Elimination (QE)

3.2.1 Model Theoretic Approach

Assume that L is a first order language with the corresponding function-, constant-, and predicate symbols. Every symbol has a fixed arity. For the sake of simplicity, we assume that the binary predicate symbol equality $=$ is always part of the language and interpreted in the standard way. Furthermore, we use the notion of term, formula, sentence, L-structure, and interpretation in the usual way. We give here the signature of two, practically relevant theories: the language of ordered rings has binary, unary and 0-ary function symbols $(+, \cdot, -, 0, 1)$ and a binary predicate symbol $<$. The language of PA has only one binary function symbol (addition), but for each natural n there is a binary predicate symbol $\equiv_n$ (congruence modulo n).

$$L_{\text{or}}: \quad (+, \dot{-}; -, 0, 1; <)$$
Language of orderded rings

$$L_{\text{PA}}: \quad (+, -; 0, 1; <, \equiv_n)$$
Language of Presburger Arithmetic

Definition (Model) Let $\mathbf{A}$ be an L-structure and ϕ is a sentence in L. We say that $\mathbf{A}$ is a model of ϕ ($\mathbf{A} \models \phi$), if ϕ is true in $\mathbf{A}$. The definition extends over set of sentences and over class of L-structures.

Definition (Theory) Let L be a language. We say that a set of sentences in L is a theory.

Definition (Closed Theory) If a theory T it is closed under the usual rules of inference, then it is said to be a closed theory.

For practical problems, if possible, we typically describe a theory T by a finite or a countable, recursively enumerable set of axioms S. The set $S \subset T$ is an axiom system for T, if $\phi \in T$ implies $S \vdash \phi$. Now there is a straightforward way to assign sentences to a structure $\mathbf{A}$ (to a structure-class $\mathbf{\Sigma}$): We collect all the sentences which are true in $\mathbf{A}$ (in all $\mathbf{A} \in \mathbf{\Sigma}$). If $\mathbf{A}$ is a structure, we call the corresponding set of sentences the $\mathbf{A}$-theory. Notation: $\text{Th}(\mathbf{A})$. In a similar way for a (consistent) set of L-sentences Φ we can collect all models of Φ. Notation: $\text{Mod}(\Phi)$. Such a structure class is called *elementary*.
For further reference we close the subsection with two more definitions.
Definition (Complete Theory) A theory T is said to be complete if for all sentence ϕ either $\phi \in T$ or $\neg\phi \in T$ holds.
Definition (Decidable Theory) A theory T is said to be decidable if there exists an algorithm such that for every sentence the algorithm is capable of deciding in finitely many steps whether the formula is provable or not.

3.2.2 Definition

We follow here [Weis06] and define quantifier elimination w.r.t. a structure class $\mathbf{\Sigma}$; other introductory model theory books define quantifier elimination for a first order theory T [Mark02]. The definition is the same if $T = \text{Th}(\mathbf{\Sigma})$.

Definition (Quantifier Elimination (QE)) Assume that L is a fixed language, $\mathbf{\Sigma}$ is an L-structure class, Φ is a set of L-formulae. We say that $\mathbf{\Sigma}$ admits quantifier elimination for Φ, if for all $\phi \in \Phi$ there exists a quantifier-free formula ψ such that

$$(*)\mathbf{\Sigma} \models \phi \Leftrightarrow \psi.$$

We explicitly mention two special cases of the definition because of their practical relevance: If Φ is the set of all L-formulae, then we say that $\mathbf{\Sigma}$ admits QE. If $\mathbf{\Sigma}$ contains only one structure $\mathbf{A}$, then we say that $\mathbf{A}$ admits QE.

Remarks

1. Since ϕ can be an arbitrary formula, ϕ, ψ can both contain free variables. Let us assume that all the free variables of ϕ, ψ are in $\bar{v} = \{v_1, \ldots, v_n\}$. Then (*) can be expressed as follows: $\models \forall_{\bar{v}}\ (\phi[\bar{v}] \Leftrightarrow \psi[\bar{v}])$
2. If the language L contains at least one constant symbol or ϕ contains a free variable and ϕ has a quantifier-free equivalent, then it also has one which has the same (or less) free variables as the input formula ϕ.
3. Quantifier elimination was used in (early) model theory to show that certain theories possess important theoretical properties like completeness, model-completeness, decadibility etc. For example if a theory admits QE, decidability boils down showing that quantifier-free formulae are decidable.

Definition (Quantifier Elimination Algorithm) A QE-algorithm is an algorithm which explicitly constructs for all $\phi \in \mathbf{\Phi}$ a quantifier-free equivalent ψ.

3.2.3 Main Theorems

Theorem 1. Let $L = \emptyset, A = \{1\}, B = \{1, 2\}, \Sigma = \{A, B\}$. Σ does not admit QE.

Proof. Consider the formula $\phi :\ \exists x\ \neg(x = y)$. Since y is the only free variable, if ϕ would have a quantifier-free equivalent ψ, then it has to be True, False, or $y = y$ or their boolean combination. But any of these is equivalent to True or False. On the other hand, $B \models \phi, A \models \neg\phi$, so ϕ is not equivalent to True or False in Σ.

Theorem 2. (DLO1) Let $L = \{<\}$, $\Sigma = \mathrm{Mod}(\mathrm{Th}(\mathbf{R}))$, Σ admits quantifier elimination and is decidable.

Theorem 3. (DOAG) Let $L = \{+, -, 0, 1; <\}$, $\Sigma = \mathrm{Mod}(\mathrm{Th}(\mathbf{R}))$, Σ admits quantifier elimination and is decidable.

Theorem 4. (PA, [Pres29]) Let $L = L_{\mathrm{PA}}$, $\Sigma = \mathrm{Mod}(\mathrm{Th}(\mathbf{Z}))$. Σ admits quantifier elimination and is decidable.

Theorem 5. (RCF-A, [Tars54]) Let $L = L_{\mathrm{or}}$, $\Sigma = \mathrm{Mod}(\mathrm{Th}(\mathbf{R}))$. Σ admits quantifier elimination and is decidable.

Theorem 6. (RCF-B, [Coll75]) Let $L = L_{\mathrm{or}}$, $\Sigma = \mathrm{Mod}(\mathrm{Th}(\mathbf{R}))$. Σ admits quantifier elimination. Moreover, there exists a quantifier elimination algorithm that is double exponential in the number of variables in the input formula. Fixing the number of variables, the QE algorithm is polynomial in the number of the

polynomials involved and in the maximum degree of the polynomials involved in the input formula. This effective quantifier elimination method is based on cylindrical algebraic decomposition (CAD).

Remarks.

- We gave each theorem an index and a label except for the first one. The labels are the standard abbreviations for the corresponding model classes (DLO1 - dense linear orders without endpoints, DOAG - torsion-free, divisible ordered Abelian groups, PA - Presburger Arithmetic, RCF - (ordered) real closed fields). The corresponding first order theories are either finitely axiomatizable or axiomatized by recursively enumerable axiom sets.
- Only those structure classes are enumerated here which are relevant for the purpose of the current thesis.
- Theorem 3 is stronger than Theorem 2 and Theorem 6 is stonger than Theorem 3: Theorem 6 contains the richest signature: notice that even multiplication between terms are allowed.
- The last theorem emphasizes the algorithmic, computational point. Collins' worst case complexity analysis of the CAD based effective quantifier elimination method opened the road to the practical implementation of real quantifier elimination.
- For the rather long and technical proofs of the positive results of the above theorems we refer to [BrMa07, Weis06, ACC98]. We only illustrate the results with several examples in the sequel. From the discussion of A a proof sketch for Theorem 5 can be exploited.

3.2.4 Examples

Example 1. Let $L = \{<\}$, $\Sigma = \{\mathbf{R}\}$ and ϕ the formula:

$$\phi : \underset{\substack{x,y \\ x \neq y}}{\forall} \underset{z}{\exists} (x < z \wedge z < y) \vee (y < z \wedge z < x)$$

ϕ is equivalent to the quantifier-free formula True.
Example 2. Let $L = L_{\mathrm{PA}}$, $\Sigma = \{\mathbf{Z}\}$ and ϕ the formula:

$$\phi : \underset{x,y}{\exists} \; 2x \geq 1 \wedge y \geq 0 \wedge 7x + y \leq 10 \wedge x + y \geq z$$

Then $\psi : z \leq 4$ is a quantifier-free formula, equivalent to ϕ.
Example 3. Let $L = L_{\mathrm{or}}$, $\Sigma = \{\mathbf{R}\}$ and ϕ the formula:

$$\phi : \underset{x}{\exists} \; x^2 + bx + c = 0$$

ϕ is equivalent to the familiar quantifier-free formula $b^2 - 4c \geq 0$.

Example 4. Let $L = L_{\mathrm{or}}$, $\Sigma = \{\mathbf{R}\}$ and ϕ the formula:

$$\phi : \underset{x,y}{\exists}\ ax+by = e \wedge cx+dy = f \wedge \underset{u,v}{\forall}\ au+bv = e \wedge cu+dv = f \Rightarrow (u = x \wedge v = y)$$

ϕ is equivalent to the familiar quantifier-free formula $ad - bc \neq 0$.

From now on we will be interested in theory of real closed fields (RCF) and the corresponding quantifier elimination problem (RQE) and consider the general QE problem only briefly in later chapters. To the interested readers, we give in Appendix A the sketch of Collins' CAD algorithm and the related notions. We also illustrate how an RQE problem is solved by the algorithm. In Appendix B we enumerate the available real quantifier elimination systems (QEPCAD [Brow03], REDLOG [DoSt97], Mathematica [Wolf99, Strz99]). A report about the main application fields and about the experiments when we used current software and hardware to test the practical scope of the method can be found in appendix C. It turns out, that suitable formalization, variable order, subproblem-decomposition, etc. can influence the successful application of the available methods and implementations.

The next chapter contains the main results of the thesis. It explains how the author exploited the quantifier elimination methods for generating witness terms. The implementation of the witness generating method in form of a *quantified constraint solver* is also included.

Chapter 4

The Extended Quantifier Elimination and Didactics

4.1 Extended Quantifier Elimination (EQE)

If the input formula of the real 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) [DSW97, Seid04] problem.

$$\exists_x \; x > 0 \wedge x^3 - 2x^2 - x + 2 = 0 \quad \rightarrow_{QE} \qquad \text{True}$$

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

Example: An extended quantifier elimination problem.

Note that the instance provided is not unique in general. The set of possible instances may even be infinite. If the formula contains free variables as well, then the instance may be not generic, i.e., for different assignments of the free variables it may have different instances:

$$\epsilon > 0 \Rightarrow \exists_{\substack{\delta \\ \delta > 0}} \forall_{x,y} \; |x - y| < \delta \Rightarrow |cx - cy| < \epsilon \quad \rightarrow_{QE} \qquad \text{True}$$

$$\epsilon > 0 \Rightarrow \exists_{\substack{\delta \\ \delta > 0}} \forall_{x,y} \; |x - y| < \delta \Rightarrow |cx - cy| < \epsilon \rightarrow_{EQE} \left\{ \text{True}, \delta \leftarrow \begin{cases} \frac{\epsilon}{c}, & c > 0 \\ 1, & c = 0 \\ \frac{-\epsilon}{c}, & c < 0 \end{cases} \right\}$$

Example: An extended quantifier elimination problem.

4.2 Main Contribution: Didactical Use of QE

4.2.1 Introduction

The author of the current thesis started to investigate the problem of automated reasoning in the field of elementary analysis during his PhD study at RISC, mainly through getting acquainted with the Theorema reasoning system [Buch00a]. The PCS prover in Theorema[Buch01, Dupr00] exploited the CAD algorithm for proving elementary analysis theorems like **CONT+**[1]. The research topic of generating witness terms by algebraic methods for deductions was initiated by Prof. B. Buchberger and Prof. T. Jebelean.
At the starting point around 2003 we knew that the implementation of the CAD algorithm is available in Mathematica 4.2 (although not yet as a standard built-in function) and a CAD based QE algorithm was implemented in the QEPCAD system. Since the Theorema reasoning system is built on top of Mathematica, it was reasonable to link the algebraic and logical techniques in the frame of the Mathematica computer algebra system. In 2004, with the release of the Mathematica version 5.1, the QE algorithm was available for the users and this provided the environment for the first experiments by the author.
A report from the first experiments and results of the author was presented in the Omega-Theorema-Ultra Workshop in Saarbruecken 2005 [Vajd05] and later also presented in the ACA06 Conference in Varna [Vajd09a].

4.2.2 Generating Existential Witness Terms

In the sequel I show how I exploited the available QE algorithm for generating real witness terms. Assume that we have a closed formula of RCF. I sketch here the main steps of the algorithms for solving the extended quantifier elimination problem over the reals. For generating the witness terms the following steps are needed:

Algorithm EQE (Input: Closed formula ϕ in PNF, Output: True/False depending on the validity of ϕ and in case of validity, witness terms for the existential variables in form of a substitution list)

1) Pre-process the input closed formula (in prenex form) by separating the variables in the outermost universal quantifier block (fixedvars), in the outermost existential quantifier block (metavars) and the remaining bound variables (boundvars),

2) Delete the first (outermost) universal and existential quantifier block,

3) Call RQE with the pre-processed formula with suitable variable order,

4) Post-process the output formula: Call CAD with the the output of RQE, if necessary. This formula (in the extended Tarski language) contains an

[1] The sum of two continuous functions is continuous

explicit description of a semialgebraic set in the (smaller dimensional) space of the fixedvars and metavars,

5) If the projection conditions hold (i.e. the initial sentence was valid), construct a DNF form,

6) In each disjunct where the condition for the fixedvars hold, exploit the cylindrical decomposition to produce algebraic descriptions of the metavars in terms of the fixed vars by consecutive backsubstitutions.

Here is an example. The input formula is:

$$\underset{a_1,a_2}{\forall}\ \underset{a_0}{\exists}\ \underset{\substack{\epsilon_0\\ \epsilon_0>0}}{\forall}\ \underset{\substack{\epsilon_1,\epsilon_2\\ \epsilon_1>0\wedge\epsilon_2>0}}{\exists}\ \underset{x,y}{\forall}\ |x-a_1|<\epsilon_1\wedge|y-a_2|<\epsilon_2\Rightarrow|(x+y)-a_0|<\epsilon_0$$

1) There are 3 $\forall\exists$-blocks, fixedvars: a_1, a_2, metavars: a_0, (other) boundvars: $\epsilon_0, \epsilon_1, \epsilon_2, x, y$:

$$\overbrace{\underset{a_1,a_2}{\forall}\ \underset{a_0}{\exists}}\ \overbrace{\underset{\substack{\epsilon_0\\ \epsilon_0>0}}{\forall}\ \underset{\substack{\epsilon_1,\epsilon_2\\ \epsilon_1>0\wedge\epsilon_2>0}}{\exists}}\ \overbrace{\underset{x,y}{\forall}}\ |x-a_1|<\epsilon_1\wedge|y-a_2|<\epsilon_2\Rightarrow|(x+y)-a_0|<\epsilon_0$$

2) Deleting the outermost universal and existential quantifiers:

$$\underset{\substack{\epsilon_0\\ \epsilon_0>0}}{\forall}\ \underset{\substack{\epsilon_1,\epsilon_2\\ \epsilon_1>0\wedge\epsilon_2>0}}{\exists}\ \underset{x,y}{\forall}\ |x-a_1|<\epsilon_1\wedge|y-a_2|<\epsilon_2\Rightarrow|(x+y)-a_0|<\epsilon_0$$

3) RQE result:

$$a_0-a_1-a_2=0$$

4-6) Post-processing the result:

$$\{\text{True},\{a_0\leftarrow a_1+a_2\}\}$$

- **[backsubstitution, processing next alternating quantifier block]**

1) There are 2 $\forall\exists$-block, fixedvars: ϵ_0, metavars: ϵ_1, ϵ_2, (other) boundvars: x, y:

$$\overbrace{\underset{\substack{\epsilon_0\\ \epsilon_0>0}}{\forall}\ \underset{\substack{\epsilon_1,\epsilon_2\\ \epsilon_1>0\wedge\epsilon_2>0}}{\exists}}\ \overbrace{\underset{x,y}{\forall}}\ |x-a_1|<\epsilon_1\wedge|y-a_2|<\epsilon_2\Rightarrow|(x+y)-(a_1+a_2)|<\epsilon_0$$

2) Deleting the outermost universal and existential quantifiers:

$$\epsilon_0>0\Rightarrow\epsilon_1>0\wedge\epsilon_2>0\wedge\underset{x,y}{\forall}\ |x-a_1|<\epsilon_1\wedge|y-a_2|<\epsilon_2\Rightarrow|(x+y)-(a_1+a_2)|<\epsilon_0$$

3) RQE result:

$\epsilon_0 > 0 \wedge 0 < \epsilon_1 < \epsilon_0 \wedge 0 < \epsilon_2 < \epsilon_0 - \epsilon_1$

4-6) [...] Post-processing the result: $\{\text{True}, \{\epsilon_1 \leftarrow \epsilon_0/2, \epsilon_2 \leftarrow \epsilon_0/2\}\}$.

- **EQE summary:**

  $\{\text{True}, \{a_0 \leftarrow a_1 + a_2, \epsilon_1 \leftarrow \epsilon_0/2, \epsilon_2 \leftarrow \epsilon_0/2\}\}$

Remarks.

- Prof. V. Weispfenning in the ACA06 Conference pointed out to the author that besides Mathematica and QEPCAD, there is another system available which addresses the problem of quantifier elimination. The REDLOG computer logic system is able to carry out quantifier elimination in several domains [DoSt97, DSW97]. Searching in the literature the author of the thesis found a presentation by A. Dolzmann[Seid04] where he described basically the extended QE problem and the method for the solution. This presentation was not known by the author till 2006. However, we must emphasize that they do not give witnesses for all the existential variables in one stroke and the witness terms can contain infinitesimal symbols which is not suitable for our purposes. Moreover, a couple of QE problems, which were not solvable with REDLOG, were sovable with Mathematica in practice. The author contacted T. Sturm and A. Dolzmann and A. Seidl. T. Sturm helped a lot by installing the REDLOG system and explaining its capabilities. There is a hope that new versions of REDLOG will support EQE in RCF, PA and other first order theories for a lot of practical problems.

- The REDLOG and the Mathematica system performs not only CAD based QE, but they also use other methods like Weispfenning's virtual substitution method [LoWe93, Weis97]. In fact, in step 3 above, the RQE algorithm can use other techniques if that allows significant speedup. However, the explicit CAD representation of the parameter space is still needed for generating the witnesses.

- We know that successful application of the above methods depends also on the chosen varible ordering. In addition, for a given input formula one may call a CAD based elimination method for all the quantified variables in one stroke or as an alternative, one may try to eliminate the first quantified variable by the virtual substitution method and then by CAD the next one, see [MoVa08]. Thus, an integrated approach needs a heuristic for choosing an optimal variable ordering, the optimal method or it tries out all possibilities and terminates at the first (time-constrtained) successful application. We refer to [Stur99]. The problem is that a priori one cannot decide in all cases which method is applicable for a given input formula and which variable ordering is optimal. However, there is still another option: In the multicore-processor era, one starts two or more alternative methods at the same time parallel and returns if one of the strategies solves

the elimination problem. The author of the thesis made such experiments on a two core machine by using the latest Mathematica's `ParallelTry` function. `ParallelTry` returns the first result received. Roughly, one core tried to eliminate the current variable by the virtual substitution, the other one by the the CAD based method. The same idea can be used by two possible alternative variable ordering.

4.2.3 Towards Simpler Witness Terms

In practice, there are two reasons to post-process the extracted witness terms:

It may happen that a witness returned by the method is a higher order algebraic number which is not expressible with radicals or a so called indexed root expression of certain parameters. To make this claim clear, consider the following EQE problem:

$$\underset{\substack{K \\ K>0}}{\exists} \; \underset{x}{\forall} \; ((x^4 - x + 1)^2 \geq K).$$

For the existential variable K, we have the constraint

$$0 < K \leq \text{Root}[-52441 + 238080\#1 - 196608\#1^2 + 655361^3, 1].$$

Instead of just returning the root expression or half of it, we may try to choose a rational instance[2], like $K \to \frac{1}{5}$. To clarify the notation above, let us consider the bigger root of $p[x] = x^2 - 2$, i.e. $\sqrt{2}$. This is represented as $[-2 + \#1^2, 2]$, because the polynomial p has two roots, and this is the bigger one. In Mathematica, roots are indexed in such a way that complex roots come always at the end of the sequence of roots, so one could also say that here the index i is always the index for the i-th real root of the given polynomial in the natural order.
Indexed root expressions with parameters usually can be avoided, if one allows piecewise polynomials or rational expressions. As we will see in the example for the local continuity of the function $f[x] = x^2$, the witness $\delta \to \sqrt{1+\epsilon} - 1$ can be taken for the constraint

$$0 < \delta \leq \sqrt{1+\epsilon} - 1.$$

Now one can argue that this is not the standard and elegant witness term with which students would come up. So we may want to search for another simpler witness term which still does the job. For that we may try quantifier elimination or series expansion. First case: If we have a linear Ansatz, we would fail. Combining two linear polynomials, i.e., piecewise defined polynomials on the positive half line, we succeed. The method yields for the coefficients $a = \frac{1}{4}$,

[2]Needless to say, if the corresponding cell has only one point with algebraic coordinates, then we have no choice.

$b = 1$ and $c = 7$, thus, we may replace the initial witness by

$$\delta \to \begin{cases} \frac{\epsilon}{4}, & \epsilon < 7 \\ 1, & \epsilon \geq 7. \end{cases}$$

Second case: the Taylor expansion yields $\sqrt{\epsilon + 1} - 1 = \frac{\epsilon}{2} - \frac{\epsilon^2}{8} + O[\epsilon^3]$, so near to zero we must choose a constant $a > 2$, such that $\frac{\epsilon}{a}$ is fine.
We give now the second argument for the necessity of post-processing. For this, we have to recall the remark made at the beginning of this chapter about the uniformity of the instances. If universally quantified variables are also present in the first quantifier block of the input formula of the EQE problem, then the witness generated for the first block may be not generic, i.e., for different assignments of the universal variables it may have different instances, as in Example 2 (see 4.1):

$$\delta \to \begin{cases} \frac{\epsilon}{c}, & c > 0 \\ 1, & c = 0 \\ \frac{-\epsilon}{c}, & c < 0 \end{cases}$$

If the goal is to reduce the number of branches (note that simpler here means something else as above), then one can try to merge two branches noting, that they have the same or almost the same witness terms. E.g.,

$$\delta \to \begin{cases} \frac{\epsilon}{|c|}, & c \neq 0 \\ 1, & c = 0 \end{cases}$$

4.2.4 Generating Universal Certificates

When generating witness terms for all the existential variables and not only for the variables in the outermost existential quantifier block, after backsubstituting the terms to the original formula, we are left with a formula where all remaining variables are universally quantified, i.e. the formula belongs to the universal theory of RCF[3]. Thus, one could see the result of the EQE not only as providing information for the existentially quantified variables, but as a reduction of the initial "solving" problem to a "checking" problem. In the example above, the corresponding checking problem would be the following:

$$\underset{a_1,a_2}{\forall} \underset{\substack{\epsilon \\ \epsilon>0}}{\forall} \underset{x,y}{\forall} |x - a_1| < \epsilon/2 \wedge |y - a_2| < \epsilon/2 \Rightarrow |(x + y) - (a_1 + a_2)| < \epsilon.$$

This has very important didactical consequences, because the checking problem is easier and there are theoretical results for gaining universal proving certificates for special classes of universal formulae [Pari03, MLHa05]. E.g., consider the following very simple checking problem:

$$\underset{x,y}{\forall} \left(\frac{x + y}{2}\right)^2 \geq xy.$$

[3]The phrase "quantifier-free fragment of the theory" is also used.

After reducing the right hand side to zero, this formula corresponds to a problem type where semidefiniteness of a multivariate polynomial should be decided. If a sum of square (SOS) decomposition of the polynomial exists, those terms in the SOS construction can play the role of a universal certificate in a deduction (here $\left(\frac{x+y}{2}\right)^2 - xy = \left(\frac{x-y}{2}\right)^2$). For a more complex example, consider the universal formula

$$\underset{a,b}{\forall}\ a+b=1 \Rightarrow a^3+b^3 \geq \frac{1}{4}.$$

Once more, after reducing the right-hand sides of the atomic subformulae to zero, checking of nonnegativity on a variety is asked. The certificates on the right-hand side of the decomposition

$$4a^3+4b^3-1 = \left(\sqrt{3}(a-b)\right)^2 + (a+b-1)(4b^2+4a^2-4ab+a+b+1)$$

makes nonnegativity obvious.
Finally, turning back to our starting example in 4.2.4, after eliminating the absolute value, the proving problem is equivalent with the following one:

$$\underset{a_1,a_2,\epsilon,x,y}{\forall}\ \epsilon^2-4(x-a_1)^2>0 \wedge \epsilon^2-4(y-a_2)^2>0 \Rightarrow$$
$$\epsilon^2-((x+y)-(a_1+a_2))^2>0.$$

We suggest in this case to consider the decomposition:

$$\epsilon^2-((x+y)-(a_1+a_2))^2 =$$
$$((x-a_1)-(y-a_2))^2+\tfrac{1}{2}(\epsilon^2-4(x-a_1)^2)+\tfrac{1}{2}(\epsilon^2-4(y-a_2)^2).$$

Positivity of term on LHS is clear, since there is a sum with a nonnegative and with two positive terms on the right hand side. This insight will be considered and unfolded in 4.3, where we describe our implementation.

4.2.5 Summary

Let us recall that reasoning can provide certainty and explanative power. In the author's opinion the generated (existential) witness terms and (universal) proving certificates contribute reasonably to the explanative power of a deduction. However, this is not achieved by validating each step of the existing decision algorithms, but rather by information-extraction. This is important for math education, because the student has the chance to understand the generated proof without knowing the steps of the underlying decision procedure.

In the next section we describe our implementation for generating witness terms and syntactical certificates. It should be emphasized that in the spirit of software reuse, the main goal was not to reimplement any existing techniques such as CAD based QE, but rather to use the available implementation of different algorithms for speeding up reasoning in specific domains. Thus we typically classify the problem, pre-process the input formula of the problem and pass it to a suitable Mathematica built-in function, but we do not exclude the possibility

of calling a self-implemented function(solver) for a particular problem type, if necessary.

Let us collect the advantages and disadvantages of our approach for generating real witness terms:

Advantages:

- We handle each quantifier block separately. This allows us that any available methods (e.g. also virtual substitution method) can be used for the elimination of quantifiers, and we do not necessarily have to use the same methods for eliminating different quantifier blocks.
- If we have several quantifier blocks, the presence of quantifiers of the preprocessed formula can make the problem easier, because not full, only partial CAD can be used.

Disadvantages:

- There is an extra time we have to pay for calling the QE method several times, if we have several quantifier blocks. However, if the initial problem is tractable with the method, the probability is big that reduced problems which contains fewer variables, i.e. the problem which gives rise to the witnesses in the second, third, etc., quantifier block, will be still tractable, moreover less time is needed for their solutions compared to the initial problem.

4.3 Implementation: Quantified Constraint System Solver (QCSS)

4.3.1 Main function

In the sequel we demonstrate the witness generator program.
We give here a piece of *Mathematica* code for the constraint solver. As pointed out in earlier subsections (3.1), the main algorithm offers flexible solution to the category of constraint problems under discussion. The name of the function is **QuantifiedConstraintSystemSolver (QCSS)**. A flexible interface is provided by exploiting Mathematica's standard construction for function options: The function **QCSS** has several options like *Mode* and *TimeConstraint*.

For the design of the interface and the choice of the options of the solver it was very helpful to study carefully a well known quantifier elimination benchmark problem family, see the details in C.2. The options have predefined (default) values, but the user can overwrite any of them and so influence the behavior of the function. The input is a closed formula with suitable signature and variable types, the output contains the information about the solvability of the constraint system and additional information. The existing implementation covers only real constraint systems thoroughly, but the frame makes it possible to extend easily the problem types and integrating other solving methods for other theories (e.g., integers or mixed real and integer linear problems).

The next table summarizes the possible values of the different options:

Option	Values
Mode	NoWitness, OneWitness
TimeConstraint	$[0, \infty)$
Methods	First, All
TranslationList	{Abs, Min, Max}
OnlyOuterWitness	True, False

Description: *Mode*=OneWitness is the didactical mode. For valid constraint sytems, the output contains sample solution as well. If *OnlyOuterWitness* is True, witnesses are only for the outermost quantifer block generated. If *TimeConstraint* $= t < \infty$, a solver will be aborted within t seconds. If *Methods* set to All, all available methods are tried for solving the constraint system. If *Translationlist* is not empty, functionsymbols contained in the list like Min, Max, Abs, are eliminated from the input formula before giving it to the solver.

```
QuantifiedConstraintSystemSolver::usage := "QCSS tries to exploit the quantifier
elimination method in RCF in order to construct witness terms. INPUT: special Tma
sentence. OUTPUT: list of valid substitutions (in case of success)."

Options[QuantifiedConstraintSystemSolver] :=
{Mode → OneWitness, TranslationList → {}, TimeConstraint → Infinity, Methods →
First, OnlyOuterWitness→False }

QuantifiedContrsintSystemSolver[labeledformula, opts___?OptionQ] :=
Module[{mode, translationlist, methods, onlyouterwitness,
   loclformula = UnWrapFormula[labeledformula], semafor, output},

   {mode, translationlist, timeconstraint, methods, onlyouterwitness} =
   {Mode, TranslationList, TimeConstraint, Methods. OnlyOuterWitness} /. {opts} /.
   Options[QuantifiedConstraintSystemSolver];

   loclformula = PreProcessFormula[loclformula, {}, translationlist];
   semafor = ProblemCategory[loclformula];

   output = Switch[semafor,

   1, CallRealPureCheck[loclformula, mode, timeconstraint, methods],
   2, CallRealPureSolve[loclformula, mode, timeconstraint, methods],
   2, CallRealParametricSolve[loclformula, mode, timeconstraint, methods],
   4, CallRealQE[loclformula, mode, timeconstraint, methods, onlyouterwitness],
   (* Other problem categories... *)
   _, Failed
   ];
   output]
```

4.3.2 Examples

Formula	Option	Output
$\forall_{x,y\in\mathbf{R}} \left(\frac{x+y}{2}\right)^2 \geq xy$	Mode→NoW	{True, {}}
$\forall_{x,y\in\mathbf{R}} \left(\frac{x+y}{2}\right)^2 \geq xy$	Mode→OneW	{True, $\{\left(\frac{x-y}{2}\right)^2\}$}
$\exists_{\epsilon_1,\epsilon_2\in\mathbf{R}\ \epsilon_1\geq 0\wedge\epsilon_2\geq 0} \epsilon_1+\epsilon_2=1$	Mode→NoW	{True, {}}
$\exists_{\epsilon_1,\epsilon_2\in\mathbf{R}\ \epsilon_1\geq 0\wedge\epsilon_2\geq 0} \epsilon_1+\epsilon_2=1$	Mode→OneW	{True, $\{\epsilon_1\to\frac{1}{2}, \epsilon_2\to\frac{1}{2}\}$}
$\forall_{\epsilon_0\in\mathbf{R}\ \epsilon_0\geq 0} \exists_{\epsilon_1,\epsilon_2\in\mathbf{R}\ \epsilon_1\geq 0\wedge\epsilon_2\geq 0} \epsilon_1+\epsilon_2=\epsilon_0$	Mode→NoW	{True, {}}
$\forall_{\epsilon_0\in\mathbf{R}\ \epsilon_0\geq 0} \exists_{\epsilon_1,\epsilon_2\in\mathbf{R}\ \epsilon_1\geq 0\wedge\epsilon_2\geq 0} \epsilon_1+\epsilon_2=\epsilon_0$	Mode→OneW	{True, $\{\epsilon_1\to\frac{\epsilon_0}{2}, \epsilon_2\to\frac{\epsilon_0}{2}\}$}

4.3.3 Description of the Interface to the Built-in Mathematica Functions

It should be clear from the code of the main program that, after the problem classification, the formula is given further to the special solver method. E.g., in the case of the pure real check the built-in Resolve function will be called with the parameter `loclformula`, which contains the pre-processed formula and the sequence of bound variables. Somewhat complicated is the function for the EQE where only the outermost witnesses are required:

```
RealEQE2[lf_,types_]:=
   Module[{ qeinputformula, bvarlist, mvarlist, fvarlist, cadcond, lsubs= {}},

   (*Init*)
   {qeinputformula,bvarlist,mvarlist,fvarlist,cadcond} =
   PreProcess1[{lf,{types}, {}, {},{True},{•[ForAll], •[ Exists]}}];

   (*Call*)
   cadres = Resolve[ qeinputformula /. {•[ ForAll] → ForAll,
          •[Exists] → Exists}, Join[fvarlist, mvarlist], Reals];

   If[CheckQEProjCond[cadres, fvarlist, mvarlist] === False, Return[False, {}]];

   (*Cylindrical Structure*)
   cadres = lexp[ CylindricalDecomposition[ cadres∧ And @@ cadcond,
   types[[All, 1]]]];

   lsubs = GenerateRealWitnesses[cadres, fvarlist, mvarlist, And @@ cadcond];

   {true,lsubs}
   ]
```

4.3.4 Witness Generation

Finally we give a piece of code for the generation of witnesses. For the sake of simplicity, assume that we have to find witnesses for ϵ_1, ϵ_2 in terms of $\epsilon_0 (> 0)$ (cf. 4.2.2) and as input we get the following CAD description:

$$\epsilon_0 > 0 \wedge 0 < \epsilon_1 < \epsilon_0 \wedge 0 < \epsilon_2 < \epsilon_0 - \epsilon_1.$$

The variable order is $\epsilon_0, \epsilon_1, \epsilon_2$. All variables stem from the same quantifier block and universally quantified variables come first. Schematically, we have to deal with the conjunction as follows: $P_1[\epsilon_0] \wedge P_2[\epsilon_0, \epsilon_1] \wedge P_3[\epsilon_0, \epsilon_1, \epsilon_2]$. We generate the witnesses for each existential variable separately. Since ϵ_1 has to satisfy the constraint P2, we could choose the witness from $(0, \epsilon_0)$ as we wish. So we may take the midpoint of the interval $(\frac{\epsilon_0}{2})$. In another situtation, for the constraints $\epsilon_0 \geq \epsilon_1$ or $\epsilon_0 < \epsilon_1$ we may take ϵ_0 or $\epsilon_0 + 1$, respectively. After backsubstitution of the chosen terms to the original formula to ϵ_1, the next existential variable (ϵ_2) can be handled in the same way. Here is the code for the different cases:

```
TakeWitness[expr1_ < expr2_, mvar_]:=mvar → (expr1+expr2)/2;
TakeWitness[expr1_ < mvar_, mvar_]:=mvar → expr1+1;
TakeWitness[mvar_ <expr1_, mvar_]:=mvar → expr1-1;
   ...
```

The substitutions and final substitution-list with the witness terms are generated by the routine `TakeWitnesses` below. As input we have the constraints in conjunctive form with suitable variable ordering and the list of metavariables to be instantiated. As output, the list of rules for the instantiations are generated. At the beginning we introduce the local variable `lslist` (local substitution list) and initialize the RHS of each rule with a symbol which cannot occur as a valid instance (for technical reasons we choose the symbol 'I')

```
TakeWitnesses[cnj_, mvars_List]:=Module[
{loccnjl, lslist = Map[Rule[#, I] &, mvars], lvar, loccond},
loccnjl = cnj;
(*Special Case: Trivial Substitutions*)
If[loccnjl === True, Return[lslist /. I → 0]];
While[loccnjl == {},
   loccond = First[loccnjl]; loccnjl = Rest[loccnjl];
   lvar = Cases[loccond, Alternatives @@ mvars][[1]];
(*Substitution List Updated*)
   lslist = lslist /. Rule[lvar, I] → TakeWitness[loccond, lvar];
(*Constraints Updated, One Metavariable Eliminated*)
   loccnjl = loccnjl /. TakeWitness[loccond, lvar]
   ]
(*Unassigned Metavars Gets Trivial Instantiation (0)*)
lslist /. Rule[var_, I] → Rule[var, 0] ]
```

4.4 Integration of the Solver into Deduction Systems

4.4.1 PCS Type Prover

The PCS prover introduced by Buchberger[Buch01] is a prover that combines **P**roving, **C**omputing and **S**olving. A meta-reasoner (strategy) orchestrates among the three different phases. Implementations of the PCS type provers can be found in the Theorema system [Dupr00, Wind01]:

- **P** stands for proving, i.e. standard routine inferencing. E.g. it handles universal quantification and implications in the goal and conjunctions in the knowledge base. See the inference rules $L2, R1, R5$, etc., in the table of subsection 2.2.
- **C** stands for symbolic computing, i.e. for rewriting formulae based on definitions of function- and predicate symbols.
- **S** stands for solving, i.e. solving quantifier-free constraints over some specific domain.

The work of the present thesis can be considered as *continuation* and *extension* of the PCS paradigm in the domain of the reals. The implementation of the PCS prover in [Dupr00] switches to the solving phase, only if the considered constraint problem is quantifier-free. In our approach, the reduction of the proving problem to algorithmic reasoning may occur in an earlier phase, where the goal formula contains quantifiers.
For the implementation of the inference rules collected in the **P**-phase we use the pattern matching machinery of the Mathematica programming language. Each inference rule transforms the current proof situation (sequent) to a new one. E.g, schematically, the implementation of the Rule $R1$ is as follows (PPND stands for Propositional Proving by Natural Deduction, see [Buch00a]):

```
PPND[f_⇒ g_, asm__]:=Module[{ }, ··· ; Psit[g,Prepend[asm,f]] ]
```

The implementation is in fact more complicated, because one also has to generate new labels for the new formulae f and g such that later we can refer to them.
Now we turn to the implementation of the **C**-phase. Based on the definitions in the knowledge base, rewrite rules will be generated and, in case of matching, the formulae in the LHS of equivalences will be rewritten to the formulae in the RHS:
Schematically, if the unary predicate P defined in terms of Q and R, for instance, the formula $\underset{x}{\forall} P[x] \Leftrightarrow (Q[x] \wedge R[x])$ is in the initial knowledge base, then the rewrite rule $P[x\_] \rightarrow (Q[x] \wedge R[x])$ is generated and is used for transforming formulae having the form $P[a]$, (a is constant) to $Q[a] \wedge R[a]$. The **S**-phase is covered by the QCSS solver.
We illustrate the phases and the reduction through a simple example. Assume that we want to prove that the function defined by $f[x] = x^2$ is locally continuous

at a particular point, say at 1. Then, we start the reasoning by considering the initial goal formula

$$\mathrm{LCont}[f, 1] \ (1)$$

Formula (1) by unfolding the definitions for local continuity and for square-function are rewritten (**C**-phase) to

$$\underset{\substack{\epsilon \\ \epsilon>0}}{\forall} \ \underset{\substack{\delta \\ \delta>0}}{\exists} \ \underset{x}{\forall} \ (|x-1| < \delta \Rightarrow |x^2 - 1^2| < \epsilon), \ (2)$$

which can be handled (meaning: its validity can be deduced) by the RQE algorithm.
In the implementation, we take into consideration Buchberger's White Box-Black Box principle for using symbolic computation in math education [Buch90] and apply it here as follows: If students are in the white box phase in learning notions of elementary analysis, i.e. they are interested in a detailed proof, the prover options are set in such a way that all the information extracted by using the witness generator are presented. However, if somebody is not interested or already in the black box phase, then for him is enough to close the deduction above by saying that

"The validity of formula (2) can be checked by RQE method." $(3a)$

Thus, let us consider the first case. The quantified constraint solver, as explained in Section 3, can provide additional information regarding Formula 2: If required, it returns particular instances for the existentially quantified variable δ, thus enabling to produce a more detailed deduction. For formula (2) it constructs the witness term $\delta \rightarrow \sqrt{1+\epsilon} - 1$.
In the educational context, explanative power is added to the deduction. Thus, instead of finishing proving Formula (2) in one step, consider the following sequence of steps:

P-Phase (universal quantifier in goal): Take ϵ arb. but fixed, assume

$$\epsilon_0 > 0$$

and prove

$$\underset{\substack{\delta \\ \delta>0}}{\exists} \ \underset{x}{\forall} \ (|x-1| < \delta \Rightarrow |x^2 - 1^2| < \epsilon_0) \ (3b)$$

We have to find a δ^* such that

$$\delta^* > 0 \wedge \underset{x}{\forall} \ (|x-1| < \delta^* \Rightarrow |x^2 - 1^2| < \epsilon_0) \ (4)$$

Using the witness provided by the solver we can instantiate the variable δ. Instantiate δ with $\sqrt{1+\epsilon_0+1} - 1$: Thus, the current goal formula is reduced to:

$$(|x_0 - 1| < \sqrt{1 + \epsilon_0} - 1) \Rightarrow |x_0^2 - 1^2| < \epsilon_0. \quad (5)$$

Now, as discussed in 4.2.3, after eliminating the absolute value by squaring or by case distinction, a universal proving certificate can stand at the end of the deduction.

4.4.2 S-Decomposition Prover

The S-Decomposition method was proposed by T. Jebelean [Jebe01]. It is a proving strategy for problems having formulae with the same structure in the knowledge base and the goal. The first version of the prover can resolve constraint problems after the reduction of the proving problem, only if a suitable lemma is present in the knowledge base. In that case it finds the terms needed for the successful completion of the proof by pattern matching. The integration of the constraint solver **QCSS** into the S-Decomposition prover makes it possible, that the proof succeeds even without the auxiliary lemmata, because the prover learns from failures, generates suitable conjectures and passes the generated conjectures to the solver. If the solver manages to handle the passed constraint problem, the completion of the proof will be based on the witness terms provided by the solver.
The so called preservation theorems lead to proving problems, in which the main formulae in the knowledge base and in the goal have the same structure. For instance, if we know that the function f and g is bounded from above then the same holds for their sum or product, i.e. for $f \oplus g$ and for $f \otimes g$. Thus, addition and multiplication preserves the property of the initial functions. Therefore we need to use different instantiations of the definition of boundedness in the assumption and in the goal. Let us consider a solution of this problem by the S-decomposition method.
In contrast to a standard refutional calculus, when a complete set of unsatisfiable formulae is needed to derive the empty clause, here we start reasoning by using the definition for the unary predicate BoundedAbove and for the standard pointwise function operations. The main difficulty of handling the preservation theorems of elementary analysis in automated reasoning is the alternating quantifier structure of the formula which formalizes the main notions like boundedness, convergence and continuity.

Assume we put the following two formulae to the knowledge base (definitions)

$$\text{BoundedAbove}[f] \Leftrightarrow \underset{K}{\exists} \underset{x}{\forall} f[x] \leq K,$$

$$\underset{x}{\forall} (f \oplus g)[x] = f[x] + g[x],$$

and the initial goal formula is

$$(\text{BoundeAbove}[f] \wedge \text{BoundedAbove}[g]) \Rightarrow (\text{BoundedAbove}[f \oplus g]).$$

Standard rules from the **C**-phase lead to the problem:

$$\langle \{ \underset{K}{\exists} \underset{x}{\forall}\ f[x] \leq K,\ \underset{K}{\exists} \underset{x}{\forall}\ g[x] \leq K \},\ \{ \underset{K}{\exists} \underset{x}{\forall}\ f[x] + g[x] \leq K \} \rangle.$$

The S-Decomposition now considers the quantifier structure and handles the outermost quantifier of the formulae present in the problem in one stroke by either introducing Skolem constants or metavariables. We use subindices for the constants and $(^*)$ for the metavariables as before. The order of the introduction of the variables is important $(K_1, K_2, K^*, x_0, x^*, x^{**})$.

Without additional knowledge the prover fails, but from the failure it generates a conjecture reconsidering the final proof situations. Here, for completing the proof, the following formula should be proven:

$$(f[x^*] \leq K_1 \wedge g[x^{**}] \leq K_2) \Rightarrow (f[x_0] + g[x_0] \leq K^*)$$

The only thing we 'know' about f and g is that they are real functions. The crucial observation is that we may eliminate some terms containing those function symbols from the formula by introducing new variables. The terms $f[x^*]$ and $f[x_0]$ may be eliminated by substituting x_0 for x^* and introducing the new variable a. Similarly, for $g[x_0]$ and $g[x^{**}]$. Thus, for proving the current goal formula, it suffices to prove

$$\underset{K_1,K_2}{\forall}\ \underset{K_0}{\exists}\ \underset{a,b}{\forall}\ a \leq K_1 \wedge b \leq K_2 \Rightarrow a + b \leq K_0.$$

This is the point where the generated conjecture is passed to the quantified constrained system solver QCSS and this tries deciding validity and extracting witness terms of the input closed formula. The solver returns in this case $\{\text{True}, \{K \rightarrow K_1 + K_2\}\}$. If the conjecture by the solver (by algorithmic reasoning) can be proven, the prover takes the witnesses and completes the proof. Moreover, by possibly reorganizing the deduction steps, it can present a proof which is easy for checking.

This is a rather simple example. Problems involving more quantifier alternation and when the solver provides terms with case distinction is technically more sophisticated but the techniques remain the same.

The elimination of the terms containing function symbols by introducing new variables in general is an interesting and non-obvious task. We note that usually the conjecture that can be generated is not unique. It can depend on the admissible order of the variables and on the matching/unifying of the terms containing functional symbols, contansts and metavariables. In the background, RQE can orient the prover in the metalevel: It can tell which possible conjecture should be taken from the candidates.

Moreover, the S-Decomoposition method in fact does more than we presented until now. Namely, it tries to decompose the initial proving problem into smaller problems. It introduces a main branch, a condition branch and a type branch. The proof strategy is to close the main branch first. However, for closing the remaining branches, the witnesses found by the constraint solver on the main branch has to be propagated to the different, yet unclosed branches of the deduction tree. In this sense the generated witnesses play a crucial role for the success of the method.

4.4.3 Evaluation of the New Approach

We differentiate between two strategies of proof presentation: The first strategy could be characterized as "follow the invention". This means: Try to avoid surprising steps and add as much explanative power as possible in the reasoning. The second kind which could be called "easy proof-checking" is just the opposite: The roots of the ideas need not to be presented, but the particular steps of the proofs must be simple and easily checkable. Thus, the deduction should have a convincing power for anybody.
The method presented here with witness generation is somewhere in-between. It is close to the second approach, since it does not tell the audience why a certain existential variable should be instantiated with the generated term, but if one accepts the proposal by the solver, then the generated proof can be easily checked. On the other hand, even for those students, which are not familiar with quantifier elimination, the presented dedution which is based on the witness generation, gives hints to them in terms of breaking down the quantifier structure of the main formula explicitly, providing witness terms and universal certificates, so it definitely gives more explanatory prover than the use of RQE as pure black box.
Now in our opinion, the underlying algebraic technique is so strong that one may also use it for such a proof presentation when the emphasis is on the invention of the solving terms. In that case, one does not come up with the solving terms regarding the first quantifier block from nowhere, but by the introduction of metavariables and constants one postpones the instantiations as long as this can be done. It is possible to reach quantifier-free problems in this way. If one wants to avoid the presentation and use of CAD for quantifier-free problems (which is reasonable in elementary analysis courses), then those problems should be handled differently. Even then, CAD can give support on the background for the reduction of the problems, but we do not investigate this direction in the frame of this thesis further.
We evaluate and compare now the solver and the extended provers with the initial real PCS approach and the former implementation by [Dupr00] in the Theorema system. Let us recall that the initial PCS handles generally only one quantifier block and without additional lemmata in the knowledge base it cannot prove the theorems in elementary analysis regarding notions with alternating quantifier structure.

In this sense the new approach can be seen as an *extension* of that prover.

As we will see in Section 5, a lot of easy analysis proving problems (which are not easy for automated reasoning at all) can now be handled with the prover. Our formalization follows mainly the $\epsilon-\delta$ definitions given by Cauchy for limits, continuity, etc. The difficulty and the time needed for solving grows if we have more variables and more alternating quantifier blocks. In our experiments, the problems contain typically 4-5 variables and maximum 3 alternating quantifier blocks, and only some problem needed 8-9 variables. Most of these problems are handled in a few seconds by the solver and the most difficult, but didactically still reasonable examples are still in the scope of the method. However one might need 30 minutes for solving the corresponding constraints in those cases, which is not optimal in an interactive didactical environment. To overcome this difficulty, either we can try to replace the input formula by pre-processing it and solving the associated contstraints which are easier and we may hope that still better hardware and software will be available in the near future.
It is not hard to see that problems involving polynomial-, piecewisely defined polynomial-, algebraic- and rational functions are in the scope of the method. Real transcendental functions like sin, cos, exp cannot be handled with the method directly. We mention here that recent research considers a sequence of lower and upper polynomial bounds on an interval for these elementary functions and with the aid of the bounds they exploit CAD for proving inequalities about transcendental functions. This extension assumes that built-in polynomial estimates for the particular transcendental function is a priori available for the prover. Consider the following example [AkPa07]:

$$\underset{\substack{x\\0<x\leq 1}}{\forall} \quad e^x \leq 1 + x + x^2$$

Consider the sequence of upper bounds on $(0, 1]$:

$$\overline{\exp(x, n)} = \frac{1}{\underline{\exp(-x, n)}} = \left(\sum_{i=0}^{2(n+1)+1} \frac{(-x)^i}{i!} \right)^{-1}$$

Taking $n = 0$, we have: $e^x \leq (1 + (-x) + (-x)^2/2 + (-x)^3/6)^{-1}$. Therefore, it is sufficient to prove the inequality:

$$\underset{\substack{x\\0<x\leq 1}}{\forall} \quad \frac{1}{(1 - x + x^2/2 - x^3/6)} \leq 1 + x + x^2.$$

This is already tractable with RQE and valid. If we would fail with the chosen upper bound $n = 0$, we may try to choose a sharper bound, i.e. a bound with a bigger index.
Recently, an extension of the real QE method was proposed by McCallum and Weispfenning [AMWe08] and also by Strzebonski [Strz08], which completetly handles the problems when the outermost variable is occuring as the argument of an exponential expression.

We close this subsection by mentioning that the direct method can be extended in another direction. Assume we have a family of polynomials, i.e., a polynomial for each natural number n. Typical examples are the Chebyshev polynomials and Legendre polynomials. If we want to prove an inequality for a particular member of the polynomial family, we have no problem, the direct method is applicable. The situation is however different, if we want a prove an inequality for all n. We refer to [GeKa06] for treating this problem.

4.4.4 Summary

We presented the method for solving the extended real quantifier elimination problem and explained how the extracted information in form of witness terms can play a crucial role in proof presentation. We discussed the implementation of a constraint solver and the integration of it into reasoning systems. We evaluated the suitability of the method for supporting reasoning in elementary analysis in educational context. We emphasize that

- the work above is a *continuation* and *extension* of the PCS method introduced by Buchberger,
- we followed the concept of software reuse for solving our problem,
- we gave explanatory power to deduction not by validating each step of the underlying RQE algorithm, but by extracting witness terms and universal certificates,
- we integrated algorithmic reasoning into deduction systems and provided a flexible interface for the quantified constraint solver,
- we were able to handle several interesting proving problems from elementary analysis.

In Section 5 we illustrate the usage of the solver and provers in math education.

Chapter 5

The CreaComp Project and Examples of Increasing Difficulties

5.1 Introduction

In this chapter we report about the CreaComp project. The goal of the project was to produce a computer supported environment for learning mathematics at freshman level. Several topics were covered by the developers and offered for the students in form of Mathematica notebooks. We will focus on the elementary analysis related notebooks, which is the contributon of the author to the project. After the description of the project and the learning system, we provide a case study on elementary analysis proving problems. We consider systematically the elementary analysis problem types which we have solved with the extended PCS method. We give each problem in terms of the defining and goal formulae, afterwards discuss the problem type. The **Proof** part will contain the Deduction/ProofObject exactly in the form how the Theorema proving system produced it automatically. We expect that the user gives definitions and lemmata and he/she states the proving problem by giving a proposition. The Prover call is always as follows:

Prove[Proposition["1"], using → {Definitions, Lemmata}]

We note here that from the inner logical point of view there is no difference between proposition and definition. The formulae in the list after the "using" keyword form the current knowledge. The keywords "proposition", "definition" and "lemma" are just wrappers for referencing logical formulae. They say something about the role of the formuale in the proofs, but do not add anything to their meaning.
We structure the investigated problems regarding the quantifier structure. We start with the simplest quantifier structure and move to more and more com-

pound structure. We differentiate between direct reducible and not directly reducible problems. We call a problem directly reducible, if after unfolding the definitions of functions and predicates, the resulting proving problem is tractable with RQE. As we discussed in Section 4.4, e.g., the preserving theorems lead to not directly reducible problems. After giving several fully elaborated examples with the complete proof trace, we might skip to a presentation mode where we only give the reduced quantified system solving problem and give the solution presented by the solver **QCSS**.

5.2 CreaComp

Several researchers in math education pointed out that although we know more and more examples where computer algebra systems could be exploited in a reasonable way for mathematics teaching and learning, typically, their integration to the curriculum and the mathematical coursewares are not completed, i.e., exercise books still contain the same old (CAS-free) exercises and methods, even without mentioning CAS, or some functionalities of CAS mentioned separately only in the appendix. So why do we not integrate also CAS in an organic way to course materials, or even more, why do we not create a new learning environment which exploits the full range of computer-support? An ideal mathematical assistant system, or math learning environment would support all types of mathematical activity. Classical CA systems are now strong in supporting exploration by computational and graphical tools. During the experimentation with tools, students can observe interesting and unexpected relations between mathematical objects and they could systematically collect data and form conjectures. In mathematics conjectures become theorems only when we established a proof for them. Thus the conjectures coming from the experiments should be verified or disproved. If this is accepted by the educators — which is not always the case —, then this is done by paper and pencil, since CAS does not really provide tools for expressing the conjectures rigorously (formalization), reasoning about them and finally structuring and maintaining bigger pieces of justified knowledge. As an experiment we tried to fill this gap and we extended a well known CA system, Mathematica [Wolf99] with reasoning capabilities. The Theorema proving system [Buch00a] provides full predicate logic with a user-friendly two-dimensional syntax and a couple of automated reasoners that produce proofs in an easy-to-read and natural presentation.

The CreaComp project [MSW07, Wind07] at the Johannes Kepler University of Linz, directed by Bruno Buchberger, Erich Peter Klement and Günter Pilz, aimed at developing interactive mathematical courseware for university students based on the graphical tools of Mathematica and the computational and reasoning tools of Theorema. The courseware covers several topics at the tertiary level: Elementary Analysis, Gröbner Bases, Fuzzy Logic, Set Theory, etc. The educational units are available as Mathematica notebooks. For the usage of the system one needs to have a locally installed Mathematica and the Theorema and CreaComp files: Those have to be put into an appropriate sub-

folder of Mathematica. Mathematica plays for the environment a threefold role: its front end provides interactive documents in a form called notebooks. As we mentioned already, it also offers a variety of numeric and symbolic algorithms. Finally its programming language is used by the developers to create all kinds of (graphical and reasoning) tools.

The author of the present thesis was not only developing the units, but took part also in the implementation of the graphical and reasoning tools. In the sequel we describe the available tools in the new environment. and show excerpts from the unit on Continuity of Real Valued Functions.

5.3 Tools in a New Learning Environment

The most important novelty of the new learning environment is the presence of *reasoning* tools. Theorem proving systems are not quite new in computer science, but they can become attractive also for math education. However, this will the case if

- they have a reasonable interface for inputting proof problems,
- they choose a readable language for formalization,
- their inference rules and proof presentation are 'natural',
- their control strategy is good enough for generating easy proofs in a reasonable time.

The last point is without doubt dependent on the current hardware as well. Before we explain in details, how the Theorema proving system aims to address these requirements, we note that in the CreaComp project some significant improvements were achieved also regarding graphical and computational tools: Graphical tools like CreaComp widgets were not available in former generation of CAS. The CreaComp computational tools allow fine user control of transformation rules for manipulating expressions. An additional requirement for education is the uniformity of syntax of the different tools: the same formulae given in a textbook-similar format should work for all plotting, computation and reasoning.

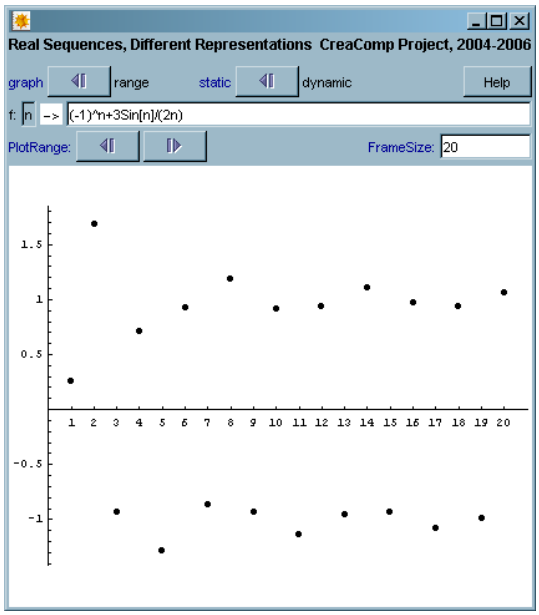
5.3.1 Graphical Tools

During our experience in using CAS in classroom we were always confronted with the fact that additional effort and time needs to be spent for adapting the particular syntax of the system, even if we would like to access the most basic functions like plotting graphs, or solving an equation or simplifying, factoring expressions. At the introductory level, the need for introducing detailed syntax can be avoided. One way to do that is to provide an additional CAS-external layer which interacts with CAS on one side and with the user/student on the other side. The external layer offers a convenient graphical interface for the

students and translates the input and output of the user and the CAS. Such solution is provided by the author in his former work [VaKo03] and also by other researchers [RoLo07]. In contrast to previous versions, many popular computer algebra systems now also provide tools for designing and implementing standard graphical interfaces within the system (e.g., widgets in Mathematica, maplets in Maple). The targeted user in a task interacts with the system by using standard graphical user interface (GUI) elements like buttons, sliders, textfields, etc. and not by typing strings in the command line. For the developers of computer supported educational units the new interfaces give challenges: They should carefully select reasonable self-contained experiments, design the corresponding graphical surface and bind all possible events triggered by the user to mathematical algorithms.

Starting with Mathematica 5, based on the GUIKit package, the development of such interactive interfaces is possible also using the standard Mathematica programming language.[1]

Example 1: Widget for Visualizing Sequences



Besides the above described GUIs, the phrase "graphical tool" preserves its original meaning: Namely they are tools for visualizing mathematical objects. In the first example we see a sequence widget: The students can study the different graphical representations of a real sequence provided by using the new GUI. If they change the number in the "FrameSize" textfield, difference between the local and global behavior of the given sequence may be detected. If they

[1] Mathematica 6 was not yet available during the CreaComp project, but in principle the new dynamic objects make the implementations even easier.

click the button "static/dynamic", they could change between an animation and a static plot of the graph of the sequence. Moreover, they could switch back and forth between the representation of the range and the graph of the sequence by clicking on the "graph/range" button. The latter representation can be useful for exploring notions as bounds and accumulation points. During the experiment, the instructor should suggest sequence types and emphasize that with the present tool only finitely many sequence elements can be represented and studied at a time.

Usually widgets provide immediate feedback for students. Immediate feedback has a crucial educational role. For instance, experiments can be desigened where the user can investigate the role of parameters. As wee see above, changing only one element of the widget, e.g., entering a new number or polynomial into an input field, all other elements of the main panel will immediately change as well according to the content of the modified textfield. Thus, graphical representation of a one parameter family of functions can be studied, or solution of systems of parametric equations can be investigated. Another useful application of widgets is the immediate assessment of student's activity and knowledge. A widget type can ask the user to give objects with a given property. All trials of the user are immediately assessed and a feedback in form of checkmark or achieved points is provided (see later Example 5, where an appropriate "δ" has to be given). Kortenkamp [Kort04] reports about the difficulties of designing good experiments. A well designed experiment can contribute and lead to a conjecture and to a proof of it, and sometimes they eliminate a need for proof, because during the experiment one finds a counterexample which falsifies the conjecture.

5.3.2 Computational Tools

Several researchers pointed out, see [Sang06], that CAS can be extremely compressive: a lot of commands produce an output without any intermediate results and details. Thus it leaves the user in a certain unsatisfactory state if he wants to know more about the transformation rules used or how the result of the computation is finally gained. The compressive outputs of the standard CAS computational commands are fine for applications but certainly not appropriate for theory exploration, i.e., for introducing new mathematical notions and methods.
To overcome this, Theorema computational tools offer a fine control of the manipulation of symbolic expressions. Only by explicit user request will be a broader set of rules applied (see Example 2).

Example 2: Computations

Definition["E3", any[n], seq[n] = $\sum_{k=0}^{n} \frac{1}{k!}$] [In]
UseAlso[Definition["E3"]] [In]

Compute[$\langle \text{seq}[n]|_{n=1,\cdots,5} \rangle$, built − in → {Built − in["Quantifiers"]}] [In]

$\langle \frac{1}{0!} + \frac{1}{1!}, \quad \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!}, \quad \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3},$

$\frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!}, \quad \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!}\rangle$ [Out]

Compute[$\langle \text{seq}[n]|_{n=1,\cdots,5}\rangle$, built − in → {Built − in["Quantifiers"], Built − in["Numbers"]}] [In]

$\langle 2, \frac{5}{2}, \frac{8}{3}, \frac{65}{24}, \frac{163}{60}\rangle$ [Out]

In Example 2, a sequence is defined and after the definition we make the definition available in a computational session. With the Theorema standard *Compute* command, we demonstrate two small computations. Although the argument of the Compute command in both cases is the same, the two output are different. The set of transformation rules can be specified by changing the built-in option of the top level command Compute. In the second case, more built-in transformation rules are applied beyond the sequence definition and the rules for quantifiers. Only the built-in simplification rules for numbers provide "the usual", compressed form for the sequence entries. The first uncompressed form is useful to study the sum quantifier which is also new to many students in a introductory calculus course.

5.3.3 Reasoning Tools

The Theorema system provides tools both for formalizing mathematical statements using the language of predicate logic and for formal deductions. In Example 3, wee see a possible definition of a function and the notion of local continuity. Afterwards, using the predicate and function symbols defined, a simple proposition is formulated. Each formula gets a label such that we later can refer to it.

Example 3: Definitions, Propositions

Definition["square", any[x], sq[x] = x^2]

Definition["lcont", any[f, a], with[$a \in \mathbf{R}$],

$$\text{LCont}[f,a] \Leftrightarrow \underset{\substack{\epsilon\in\mathbf{R}\\ \epsilon>0}}{\forall}\ \underset{\substack{\delta\in\mathbf{R}\\ \delta>0}}{\exists}\ \underset{\substack{x\in\mathbf{R}\\ |x-a|<\delta}}{\forall}\ |f[x]-f[a]|<\epsilon$$

Proposition["sqiscont1", LCont[sq,1]]

For reasoning the top level command is *Prove*. Similar to the top level command for computing, the Prove command also has several options: One can specify by which prover and relative to which knowledge base should the current goal formula be proven.

Prove[Proposition["sqiscont1"],

using→{Definition["lcont"],Definition["square"]},by→TmaAnalysisProver]

After each call we get a proof or a deduction attempt in a separate Mathematica notebook. For each inference steps an explanatory text is also attached as we will see in the elaborated examples in 5.4

In Theorema, there are several general purpose provers are available for classical logic problems: PropositionalProver, PredicateProver, ResolutionProver, InductionProver etc. However, we think that for supporting practical reasoning in a specific domain like elementary analysis, purely logical techniques will not suffice. Therefore, combined methods like the PCS method introduced by Buchberger [Buch00a] or the S-Decomposition strategy with extended quantifier elimination [Jebe01, Vajd09a] supports the investigation of proving problems in the CreaComp elementary analysis units. For a further description of the Theorema formal language and reasoners we refer to [Buch00a, Wind01, WBR06, MSW07].

5.4 Problem Types and Proofs

5.4.1 Problems Involving One Alternating Quantifier Block ($\forall\exists$)

Investigation of Fixpoints and Cycles

Problem Specification 1
Definition["two-cycle", any$[f, a, b]$, with$[a \in \mathbb{R} \wedge b \in \mathbb{R}]$,

$$\text{HasTwoCycle}[f, a, b] \Longleftrightarrow \underset{\substack{x\in\mathbb{R}\\ a\leq x\wedge x\leq b}}{\exists} ((f[f[x]] = x) \wedge f[x] \neq x)\Bigg]$$

Definition["logmap4", any$[x]$, f1$[x] = 4x(1-x)$]

Proposition["logmap4-II", HasTwoCycle[f1, 0, 1]]
Remarks The definition of the main predicate (HasTwoCycle) contains only one existential quantifier. We investigate a particular instance of the logistic map family $f_r[x] = rx(1-x)$ [Weiss, Shaw06] in the sequel and we restrict our attention to the bounded interval [0,1]. We say that x is fixpoint if $f[x] = x$. The map f has a two-cycle, if there exists an $x \in [0, 1]$ such that second iterative of f applied to x is x itself and x is not a fixpoint. It is known that depending on r, f_r has n-cycles ($n = 1, 2, 3, \cdots$), and to explicitly characterize the set of the values r for which f_r has an n-cycle is computationally difficult. The study of the problem can lead to such interesting mathematical phenomena as bifurcation.
Proof

Prove:
(Proposition (logmap4-II)) HasTwoCycle[f1, 0, 1],
under the assumptions:
(Definition (two-cycle))

$$\underset{\substack{f,a,b\\ a\in\mathbb{R}\wedge b\in\mathbb{R}}}{\forall}\left(\text{HasTwoCycle}[f, a, b] \Leftrightarrow \underset{\substack{x\in\mathbb{R}\\ a\leq x\wedge x\leq b}}{\exists} ((f[f[x\ ]] = x) \wedge \neg(f[x] = x))\right),$$

(Definition (logmap4)) $\underset{x}{\forall}(\text{f1}[x] = 4 * x * (1-x))$.

Formula (Proposition (logmap4-II)), using (Definition (two-cycle)), is implied by:

$0 \in \mathbb{R} \wedge 1 \in \mathbb{R} \wedge \underset{\substack{x\in\mathbb{R}\\ 0\le x\wedge x\le 1}}{\exists} ((\mathrm{f1}[\mathrm{f1}[x]] = x) \wedge \neg(\mathrm{f1}[x] = x)),$

which, using (Definition (logmap4)), is implied by:

$0 \in \mathbb{R} \wedge 1 \in \mathbb{R} \wedge \underset{\substack{x\in\mathbb{R}\\ 0\le x\wedge x\le 1}}{\exists} ((4 * \mathrm{f1}[x] * (1 - \mathrm{f1}[x]) = x) \wedge \neg(4 * x * (1 - x) = x)),$

which, using (Definition (logmap4)), is implied by:

(1) $0 \in \mathbb{R} \wedge 1 \in \mathbb{R} \wedge$

$\underset{\substack{x\in\mathbb{R}\\ 0\le x\wedge x\le 1}}{\exists} ((4*(4*x*(1-x))*(1-4*x*(1-x)) = x) \wedge \neg(4*x*(1-x) = x)).$

We prove the individual conjunctive parts of (1):

Proof of (1.1) $0 \in \mathbb{R}$:

Type condition in (1.1) was checked.

Proof of (1.2) $1 \in \mathbb{R}$:

Type condition in (1.2) was checked.

Proof of (1.3)

$\underset{\substack{x\in\mathbb{R}\\ 0\le x\wedge x\le 1}}{\exists} ((4*(4*x*(1-x))*(1-4*x*(1-x)) = x) \wedge \neg(4*x*(1-x) = x))$:

We have to find x^* such that

(2) $x^* \in \mathbb{R} \wedge 0 \le x^* \wedge x^* \le 1$

$\wedge\, (4 * (4 * x^* * (1 - x^*)) * (1 - 4 * x^* * (1 - x^*)) = x^*) \wedge \neg\, (4 * x^* * (1 - x^*) = x^*).$

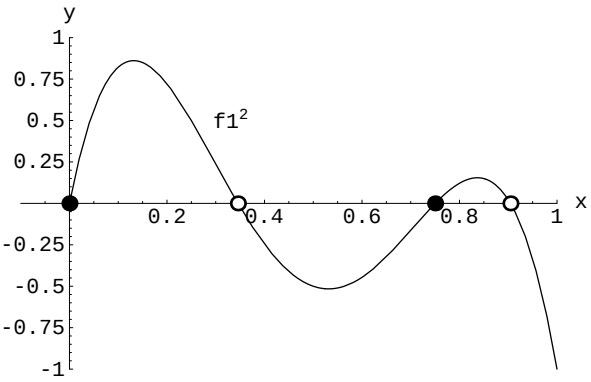
Using the Cylindrical Decomposition method, goal (2) can be solved for x^* . The method yields a sample solution

$x^* \leftarrow \frac{1}{8} * \left(5 + (-1) * 5^{\frac{1}{2}}\right).$

Hence formula (2) is solved, and we are done. ♠

Explanation to the Deduction

The deduction begins with two rewriting steps: Defintions for the logistic map and for the twocycle are unfolded (1). A direct reduction leads to an existentially quantified first order formula. The metavariable $x^*$is introduced (2). The metavariable x^* is instantiated by using the algebraic method. The next picture shows the real roots of the second iterate of f1: $0, \frac{1}{8}(5-\sqrt{5}), \frac{3}{4}, \frac{1}{8}(5+\sqrt{5})$. It turns out that two of the four roots will be a solution of the real constraint system and the lesser ($\frac{1}{8}(5-\sqrt{5})$, denoted by the first white disk from the left) is picked up for the witness in the proof.



Let us note that there is no universal and one existential quantifier in the first quantifier block in the reduced problem. The number of quantifer blocks is one. Thus, this is the easiest instance of the problem type $\forall\exists$.

5.4.2 Problems Involving Two Alternating Quantifier Blocks ($\forall\exists\forall\exists$)

Investigation of Boundedness

Problem Specification 2

Definition $\left["\text{bounded}", \text{any}[f], \text{Bounded}[f] \Longleftrightarrow \underset{K\in\mathbb{R}}{\exists}\underset{x\in\mathbb{R}}{\forall}|f[x]| \leq K\right]$

Definition $\left["\text{f1}", \text{any}[x], f[x] = \frac{x}{x^2+1}\right]$

Proposition["ratbounded", Bounded[f1]]

Remarks The proposition states here that the rational function f1 is globally bounded.

Proof

Prove:

(Proposition (ratbounded))Bounded[f1],

under the assumptions:

(Definition (bounded))$\underset{f}{\forall}\left(\text{Bounded}[f] \Leftrightarrow \underset{K\in\mathbb{R}}{\exists}\underset{x\in\mathbb{R}}{\forall}(|\text{f1}[x]| \leq K)\right)$,

(Definition (f1))$\underset{x}{\forall}\left(\text{f1}[x] = \frac{x}{x^2+1}\right)$.

Formula (Proposition (ratbounded)), using (Definition (bounded)), is implied by:

(1)$\underset{K\in\mathbb{R}}{\exists}\underset{x\in\mathbb{R}}{\forall}(|\text{f1}[x]| \leq K)$.

We have to find K^* such that

(2)$K^* \in \mathbb{R} \wedge \underset{x\in\mathbb{R}}{\forall}(|\text{f1}[x]| \leq K^*)$.

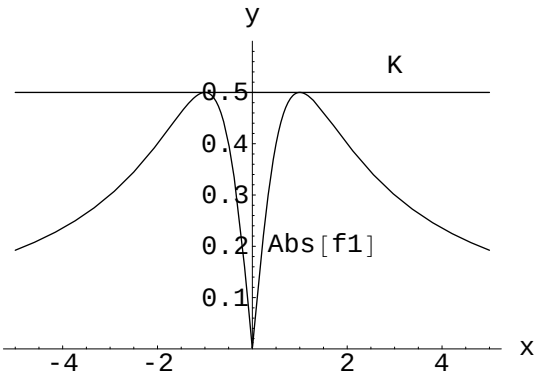
Formula (2), using (Definition (f1)), is implied by:

(3)$K^* \in \mathbb{R} \wedge \underset{x\in\mathbb{R}}{\forall}\left(\left|\frac{x}{x^2+1}\right| \leq K^*\right)$.

Based on the algebraic methods, in goal (3) we take the following instantiation for K^* as a guess:

$K^* \leftarrow \frac{1}{2}$.
Hence goal (3) would be implied by
(4) $\underset{x\in\mathbb{R}}{\forall}\left(\left|\frac{x}{x^2+1}\right| \leq \frac{1}{2}\right)$.
We assume
(5)$x_0 \in \mathbb{R}$,
and show
(6)$\left|\frac{x_0}{{x_0}^2+1}\right| \leq \frac{1}{2}$.
Using the Cylindrical Decomposition method, goal (6) can be proven. ♠

Explanation to the Deduction
The reduced problem has two quantifier blocks. In the first block there is no universal and one existential, in the second block there is one universal and no existential quantifier. In fact the witness term ($\frac{1}{2}$) is the best bound for the function f1.



Problem Specification 3

(Definition (bounded): bnd:)$\underset{f}{\forall}\left(\text{Bounded}[f] :\Leftrightarrow \underset{\substack{B\in\mathbb{R}\\B>0}}{\exists}\underset{a\in\mathbb{R}}{\forall}(|f[a]| \leq B)\right)$,
(Definition (sum of functions): $\oplus$:) $\underset{f,g,x}{\forall}((f \oplus g)[x]{:=}f[x] + g[x])$.
(Proposition (bounded sum): bnd+)
Bounded$[f] \wedge$ Bounded$[g] \Rightarrow$ Bounded$[f \oplus g]$,

Remarks This proposition states that the sum of two bounded functions is bounded. Thus here we investigate once more the boundedness of functions, but this proving problem is a different from Problem 1 and 2. This proving problem cannot be reduced directly just by unfolding the definitions of boundedness and the operators to a solving problem. An additional step is needed for replacing terms containing function symbols.

Proof

Prove:
(Proposition (bounded sum): bnd+)
Bounded$[f] \wedge$ Bounded$[g] \Rightarrow$ Bounded$[f \oplus g]$,
under the assumptions:

(Definition (bounded): bnd:)$\underset{f}{\forall}\left(\text{Bounded}[f] \nRightarrow \underset{\substack{B\in\mathbb{R}\\B>0}}{\exists}\underset{a\in\mathbb{R}}{\forall}(|f[a]|\leq B)\right)$,

(Definition (sum of functions): $\oplus$:)$\underset{f,g,x}{\forall}((f\oplus g)[x]{:=}f[x]+g[x])$.

For proving (Proposition (bounded sum): bnd+), we assume:

(2)Bounded$[f]$,

(3)Bounded$[g]$,

and we prove:

(1)Bounded$[f\oplus g]$.

Using the definition (Definition (bounded): bnd:), we expand the goal (1) and the assumptions (2) and (3) into:

(4)$\underset{\substack{B\in\mathbb{R}\\B>0}}{\exists}\underset{a\in\mathbb{R}}{\forall}(|(f\oplus g)[a]|\leq B)$

(5)$\underset{\substack{B\in\mathbb{R}\\B>0}}{\exists}\underset{a\in\mathbb{R}}{\forall}(|f[a]|\leq B)$

(6)$\underset{\substack{B\in\mathbb{R}\\B>0}}{\exists}\underset{a\in\mathbb{R}}{\forall}(|g[a]|\leq B)$

Using (Definition (sum of functions): $\oplus$:), the goal (4) is transformed into:

(7)$\underset{\substack{B\in\mathbb{R}\\B>0}}{\exists}\underset{a\in\mathbb{R}}{\forall}(|f[a]+g[a]|\leq B)$.

By the assumptions (5) and (6), we can take appropriate constants $\{\{B_1,B_1\in\mathbb{R},B_1>0\},\{B_2,B_2\in\mathbb{R},B_2>0\}\}$ such that:

(9)$B_1\in\mathbb{R}\wedge\left(B_1>0\wedge\underset{a\in\mathbb{R}}{\forall}(|f[a]|\leq B_1)\right)$

(10)$B_2\in\mathbb{R}\wedge\left(B_2>0\wedge\underset{a\in\mathbb{R}}{\forall}(|g[a]|\leq B_2)\right)$

For proving (7), we have to find appropriate values $\{\{B_0^*,B_0^*\in\mathbb{R},B_0^*>0\}\}$ such that:

(8)$B_0^*\in\mathbb{R}\wedge\left(B_0^*>0\wedge\underset{a\in\mathbb{R}}{\forall}(|f[a]+g[a]|\leq B_0^*)\right)$.

We decompose the proof into three branches.

1. Main branch: we assume:

(9.3)$\underset{a\in\mathbb{R}}{\forall}(|f[a]|\leq B_1)$

(10.3)$\underset{a\in\mathbb{R}}{\forall}(|g[a]|\leq B_2)$

and we prove:

(8.3)$\underset{a\in\mathbb{R}}{\forall}(|f[a]+g[a]|\leq B_0^*)$

For proving (8.3), we replace the universal variables by arbitrary constants $\{\{a_0,a_0\in\mathbb{R}\}\}$, and we prove:

(11)$a_0\in\mathbb{R}\Rightarrow|f[a_0]+g[a_0]|\leq B_0^*$.

We instantiate the assumptions (9.3) and (10.3) with appropriate values $\{\{a_1^*,a_1^*\in\mathbb{R}\},\{a_2^*,a_2^*\in\mathbb{R}\}\}$ (to be found later):

(12)$a_1^*\in\mathbb{R}\Rightarrow|f[a_1^*]|\leq B_1$

(13)$a_2^*\in\mathbb{R}\Rightarrow|g[a_2^*]|\leq B_2$

We decompose the proof into two branches.

1. Main branch: we assume:
(12.3)$|f\,[a_1^*]| \leq B_1$
(13.3)$|g\,[a_2^*]| \leq B_2$ and we prove:
(11.3)$|f\,[a_0] + g\,[a_0]| \leq B_0^*$

In order to prove (11.3), by using the assumptions (12.3) and (13.3), it suffices to prove:
(15)$|f\,[a_1^*]| \leq B_1 \wedge |g\,[a_2^*]| \leq B_2 \Rightarrow |f\,[a_0] + g\,[a_0]| \leq B_0^*$.

We eliminate the non-algebraic function symbols from (15), using the replacements $\{a_1^* \to a_0, f\,[a_0] \to x_1, a_2^* \to a_0, g\,[a_0] \to x_2\}$:
(16)$|x_1| \leq B_1 \wedge |x_2| \leq B_2 \Rightarrow |x_1 + x_2| \leq B_0^*$.

We transform (16) by adding the appropriate quantifiers:
(Conjecture−1)
$\underset{\substack{B1\in\mathbb{R}\\B1>0}}{\forall}\ \underset{\substack{B2\in\mathbb{R}\\B2>0}}{\forall}\ \underset{\substack{B0\in\mathbb{R}\\B0>0}}{\exists}\ \underset{x2\in\mathbb{R}}{\forall}\ \underset{x1\in\mathbb{R}}{\forall}(|x1| \leq B1 \wedge |x2| \leq B2 \Rightarrow |x1 + x2| \leq B0)$.

By QCSS, we prove the formula (Conjecture−1) and we find the witnesses: $\{B_0^* \to B_1 + B_2\}$.

2. Type branch: we assume: [...]

Explanation to the Deduction
Using the definitions, the predicate Bounded and the function operator is eliminated. The reduction attempt into a standard real constraint problem cannot succeed, because the last goal formula contains such terms like $f[a_0]$. Those have to be handled in an extra step. Formula (16) shows the generated conjecture after term-replacements. The generated conjceture Conjecture-1 is a quantified constraint problem. This problem is passed to the Solver by the Prover. Conjecture-1 is valid and the QCCS solver gives back in addition the witness $B_1 + B_2$. That witnesses is propagated by the Prover to the other branches of the proof tree (type and assumption branches here are not shown, because only standard checking steps remain).

5.4.3 Problems Involving Three Alternating Quantifier Blocks (∀∃∀∃∀∃)

Investigation of Limits

Problem Specification
Definition["converges", any[f],
Converges[f] $\Longleftrightarrow$

$$\left[\underset{\substack{a\in\mathbb{R}\\\epsilon>0}}{\exists}\ \underset{\epsilon\in\mathbb{R}}{\forall}\ \underset{M\in\mathbb{R}}{\exists}\ \underset{x\in\mathbb{R}}{\forall}((x \geq M) \Longrightarrow (|f[x] - a| < \epsilon))\right]$$

Definition $\left["\text{f1}", \text{any}[x], \text{f1}[x] = \frac{1}{x}\right]$
Proposition["ratconv", Converges[f1]]
Remarks This proving problem investages the limit of a rational function at infinity.
Proof

Prove:

(Proposition (ratconv))Converges[f1],

under the assumptions:

(Definition (converges))$\underset{f}{\forall}\left(\text{Converges}[f] \Leftrightarrow \underset{\substack{a\in\mathbb{R}\\ \epsilon>0}}{\exists}\underset{\epsilon\in\mathbb{R}}{\forall}\underset{M\in\mathbb{R}}{\exists}\underset{x\in\mathbb{R}}{\forall}(x \geq M \Rightarrow |f[x]-a| < \epsilon)\right)$,

(Definition (f1))$\underset{x}{\forall}\left(\text{f1}[x] = \frac{1}{x}\right)$.

Formula (Proposition (ratconv)), using (Definition (converges)), is implied by:

(1) $\underset{\substack{a\in\mathbb{R}\\ \epsilon>0}}{\exists}\underset{\epsilon\in\mathbb{R}}{\forall}\underset{M\in\mathbb{R}}{\exists}\underset{x\in\mathbb{R}}{\forall}(x \geq M \Rightarrow |\text{f1}[x]-a| < \epsilon)$.

We have to find a^* such that

(2)$a^* \in \mathbb{R} \wedge \underset{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}}{\forall}\underset{M\in\mathbb{R}}{\exists}\underset{x\in\mathbb{R}}{\forall}(x \geq M \Rightarrow |\text{f1}[x]-a^*| < \epsilon)$.

Formula (2), using (Definition (f1)), is implied by:

(3)$a^* \in \mathbb{R} \wedge \underset{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}}{\forall}\underset{M\in\mathbb{R}}{\exists}\underset{x\in\mathbb{R}}{\forall}\left(x \geq M \Rightarrow \left|\frac{1}{x}-a^*\right| < \epsilon\right)$.

Based on the algebraic methods, in goal (3) we take the following instantiation for a^* as a guess:

$a^* \leftarrow 0$.

Hence goal (3) would be implied by

(4) $\underset{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}}{\forall}\underset{M\in\mathbb{R}}{\exists}\underset{x\in\mathbb{R}}{\forall}\left(x \geq M \Rightarrow \left|\frac{1}{x}-0\right| < \epsilon\right)$.

We assume

(5)$\epsilon_0 > 0 \wedge \epsilon_0 \in \mathbb{R}$,

and show

(6) $\underset{M\in\mathbb{R}}{\exists}\underset{x\in\mathbb{R}}{\forall}\left(x \geq M \Rightarrow \left|\frac{1}{x}-0\right| < \epsilon_0\right)$.

We have to find M^* such that

(7)$M^* \in \mathbb{R} \wedge \underset{x\in\mathbb{R}}{\forall}\left(x \geq M^* \Rightarrow \left|\frac{1}{x}-0\right| < \epsilon_0\right)$.

Based on the algebraic methods, in goal (7) we take the following instantiation for M^* as a guess:

$M^* \leftarrow 1+{\epsilon_0}^{-1}$.

Hence goal (7) would be implied by

(8) $\underset{x\in\mathbb{R}}{\forall}\left(x \geq 1+{\epsilon_0}^{-1} \Rightarrow \left|\frac{1}{x}-0\right| < \epsilon_0\right)$.

We assume

(9)$x_0 \in \mathbb{R}$,

and show

(10)$x_0 \geq 1+{\epsilon_0}^{-1} \Rightarrow \left|\frac{1}{x_0}-0\right| < \epsilon_0$.

We prove (10) by the deduction rule.

We assume

(11)$x_0 \geq 1+{\epsilon_0}^{-1}$

and show

(12)$\left|\frac{1}{x_0}-0\right| < \epsilon_0$.

Using the Cylindrical Decomposition method, goal (12) can be proven. ♠

Explanation to the Deduction This is a direct reducible problem with three quantifier blocks. Note that witnesses for a and for M are generated too. Finally the problem is reduced to the checking problem. The validity of the formula (12) using the assumptions (5) and (11) is obtained by the algebraic method.

5.5 Direct ERQE Problems with the QCSS Solver

5.5.1 Introduction

In the previous subsection we presented several complete proof presentations. Proof problems were reduced by pure logical inference steps to real constraint solving problems. The deduction is finished by solving the constraint problem. In addition, the solver generated witnesses for the instantiation of the metavariables. Thus, after reducing the proving problem to a general constraint solving problem, we reduced the problem to a quantifier-free checking problem. The final check can be done by the algebraic method or based on universal certificates as described in 4.2.3. Thus, we illustrated, how the constraint solver was integrated to a domain specific prover paying attention to didactical issues.

From this point on we switch to a different presentation in the remaining part of the current chapter. As was mentioned at the beginning of 4.3, the author of the thesis implemented the QCSS solver in Mathematica which can also be used as a standalone application. Thus, we give some appropriate first order sentences as inputs for the solver and illustriate the functionalilties of the solver. We always say something about the context in which the input formula can occur. The input formula in the next problems can be seen as the result of the reduction of a proving problem similar to the ones discussed in 5.4. All problems intend to highlight a specific aspect of a basic analysis concept such as limit, continuity, Lipschitz-continuity, uniform continuity, maximum point, approximation, optimization, etc. The structure of the presentiation is as follows:

- The *input* closed formula is given.
- The *call* to the solver is presented. In some cases here we emphasize the role of the solver options.
- Output *witnesses* are shown.
- *Explanation* to the formula, the options and the solving method closes the demonstration of a problem.

Before the examples two remarks are in order. The first one is about the syntax of input formulae. The language of order rings can be extended simply by adding new function symbols like Min, Max, Abs and by function expressions which are defined piecewise. For instance, one can pre-process the atomic formula $|t| > 0$ by replacing it with $(t > 0 \vee t < 0)$. With a little more care even rational function expressions and squareroots can be added with an extra

condition for their domains. The second remark is about the form of the presentation: Every input and output formula is copied here in the same form as it appears in the computer algebra nootebook.

5.5.2 Local Continuity

Input

Lemma $\Big[$"LC1", $\underset{x,\epsilon\in\mathbb{R}}{\forall}\big((\epsilon \geq 0 \wedge |x-1| < \text{Min}[1,\epsilon/3]) \Longrightarrow \big(\big|x^2-1^2\big| < \epsilon\big)\big)$ $\Big]$

Call

QuantifiedConstraintSystemSolver[Lemma["LC1"], Mode → "NoWitness"]

Output

{True, {}}

Explanation

In this problem no witnesses are generated. Assume that local continuity of the function $f[x] = x^2$ is investigated at $x = 1$. Moreover, assume that one uses the Cauchy definition and constructed a suitable δ-candidate for a positive $\epsilon : \delta[\epsilon] = \text{Min}[1, \epsilon/3]$. Correctness of the candidate can be verified by the solver using the formula above.

5.5.3 Extraneous Point

Input

Lemma["ExtraneousPoint",

$\underset{x,y\in\mathbb{R}}{\exists}(x > -63 \wedge$

$\big(8954912x^3 - 1777440x^2y + 1710485868x^2 + 44100xy^2 -$

$223957440xy + 108895082184x + 27y^4 + 2778300y^2 - 7054659360y +$

$2310620648364 = 0))]$

Call

QuantifiedConstraintSystemSolver[Lemma["ExtraneousPoint"]]

Output

$\big\{\text{True}, \big\{x \to \frac{-30758091}{559682}, y \to \frac{42875}{529}\big\}\big\}$

Explanation

This formula is taken from algebraic geometry, see e.g. [Seid04]. Assume that a parametric curve C is given by $x[t] = -6t^4 - 63$ and $y[t] = 92t^3 + 70t^2$. We can search for the smallest variety on the plane which contains the points of the curve. By resultant computation we obtain that the answer is defined by the bivariate polynomial $p[x, y]$, where the coefficents of p is given in the input formula on the left hand side of the equation. However, doing explicit implicitization by quantifer elimination, it turns out that the real variety V is bigger than the set of points on the curve. Therefore there exists a point in V, which does not belong to the curve C. Such a point is found by the solver.

5.5.4 Two-Cycles

Input

Lemma $\left[\text{"L-II"}, \underset{\substack{r\in\mathbb{R}\\ r>3}}{\forall}\ \underset{\substack{x\in\mathbb{R}\\ 0\le x\le 1}}{\exists} \left(\left(r^2x(1-x)(1-rx(1-x))=x\right)\wedge\left(rx(1-x)\right)\neq x\right)\right]$

Call

QuantifiedConstraintSystemSolver[Lemma["L-II"]]

Output $\left\{\text{True}, \left\{x\to \frac{1+r}{2r}+\left(\frac{-1}{2}\right)\frac{-3+r^2+(-2)r^{\frac{1}{2}}}{r^2}\right\}\right\}$

Explanation

This input formula formulates that the standard logistic map with parameter r has a 2-cycle, if $r > 3$. Thus, it is a parametric version of the problem 5.4.1. Validity of the input formula is checked and a witness is generated by the Solver. In a very similar manner the 3-cycles and 4-cycles of the same map or fixponts and n-cycles of the cubic logistic map $g_[x] = rx(1 - x^2)$ can be investigated, see [Weiss, Shaw06].

5.5.5 Lipschitz Continuity

Input

Lemma $\left[\text{"LIPC"}, \underset{K\in\mathbb{R}}{\exists}\ \underset{x,y\in\mathbb{R}}{\forall}(-3\le x\le 7\wedge -3\le y\le 7)\Longrightarrow \left(\left|x^2-y^2\right|\le K|x-y|\right)\right]$

Call

QuantifiedConstraintSystemSolver[Lemma["LIPC"]]

Output

$\{\text{True}, \{K\to 14\}\}$

Explanation

This input formula corresponds to the statement that the square function $f[x] = x^2$ is Lipschitz continuous in the closed bounded interval $I = [-3, 7]$. A specific instance for K is called a Lipschitz constant. Thus, here $K = 14$ is Lipschitz constant for f in I. The function f is not Lipschitz continuous and not uniformly continuous on the whole real line.

5.5.6 Parametric Optimization

Input

Lemma $\left[\text{"pgo"},\ \underset{c\in\mathbb{R}}{\forall}\ \underset{\text{x0}\in\mathbb{R}}{\exists}\ \underset{x\in\mathbb{R}}{\forall}\left(-\text{x0}^4+c\text{x0}^2\ge -x^4+cx^2\right)\quad \text{"max"}\right]$

Call

QuantifiedConstraintSystemSolver[Lemma["pgo"]]

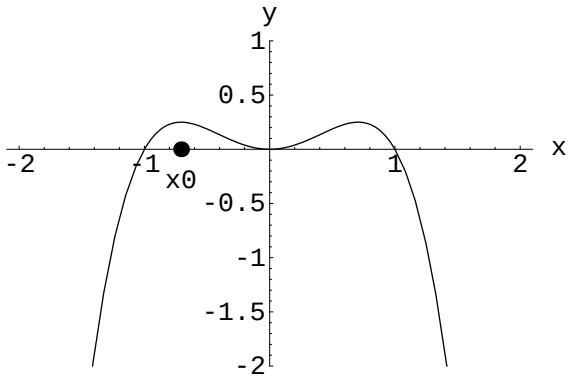
Output

$\left\{\text{True}, \left\{\text{x0}\to \left\|0\Leftarrow c\le 0, \frac{-c^{\frac{1}{2}}}{2^{\frac{1}{2}}}\Leftarrow c>0\right\|\right\}\right\}$

Explanation

The input formula states that a parametric quartic has a global maximum. In fact there are two maximum points for $c > 0$ and zero is the unique maximum

point for $c \leq 0$. The witness is given in the form of a case disctinction. The image below shows the lesser maximum point for the case $c = 1$.



5.5.7 Chebyshev Polynomials

Input

Lemma["CHEB2",

$$\underset{a,b\in\mathbb{R}}{\exists}\underset{x,\mathrm{a1},\mathrm{b1}\in\mathbb{R}}{\forall}\underset{-1\leq x\leq 1}{\underset{y\in\mathbb{R}}{\exists}}\left(\left(-1\leq y\leq 1\wedge\left(x^2-(ax+b)\right)^2\leq\left(y^2-(\mathrm{a1}\ y+\mathrm{b1})\right)^2\right)\right)\Bigg]$$

Call

QuantifiedConstraintSystemSolver[Lemma["CHEB2"],

OnlyOuterWitness→ True]

Output

$\left\{\text{True},\left\{a\to 0,b\to\frac{1}{2}\right\}\right\}$

Explanation

The input formula corresponds to the instance of the Solotareff approximation problem family, see the description in C.2. Here the polynomial $p[x] = x^2$ is approximated in $[-1, 1]$ by linear polynomials. The best appoximation by using the supremum norm is the polynomial $q[x] = (0)x + (\frac{1}{2})$. The solver computes the unknown coefficients (a, b) and gives specific values back as witnesses. A solver option guarantee that only the outermost wintnesses are returned. This problem can be seen in a different way. One can say that we are searching for a monic quadratic which is closest to the zero polynominal in $[-1, 1]$. The result is $x^2 - \frac{1}{2}$, which is in fact the second Chebyshev polynomial of the first kind.

5.5.8 Limit

Input

$$\text{Lemma}\left["\lim",\ \underset{\lambda\in\mathbb{R}}{\exists}\underset{\epsilon>0}{\underset{\epsilon\in\mathbb{R}}{\forall}}\underset{\delta>0}{\underset{\delta\in\mathbb{R}}{\exists}}\underset{x\neq 4}{\underset{x\in\mathbb{R}}{\forall}}\left(|x-4|<\delta\right)\Longrightarrow\left(\left|\left(\frac{\sqrt{2x+1}-3}{\sqrt{x}-2}\right)-\lambda\right|<\epsilon\right)\right]$$

Call
QuantifiedConstraintSystemSolver[Lemma["lim"], OnlyOuterWitness → True]
Output
$\{\text{True}, \{\lambda \to \frac{4}{3}\}\}$
Explanation
The input formula formulates that the function $f[x] = \frac{\sqrt{2x+1}-3}{\sqrt{x}-2}$ has limit at $x = 4$. Once more, a witness only for the outermost block is generated. Thus, we see that by definition, the limit is $\lambda = \frac{4}{3}$.

5.5.9 Parametric Limit

Input
Lemma["plim",

$$\underset{a,b\in\mathbb{R}}{\forall}\underset{\lambda\in\mathbb{R}}{\exists}\underset{\epsilon>0}{\underset{\epsilon\in\mathbb{R}}{\forall}}\underset{M\in\mathbb{R}}{\exists}\underset{x\in\mathbb{R}}{\forall}(x \geq M) \Longrightarrow \left(\left|\left(\sqrt{(x+a)(x+b)} - x\right) - \lambda\right| < \epsilon\right) \quad \text{"Inf"} \Bigg]$$

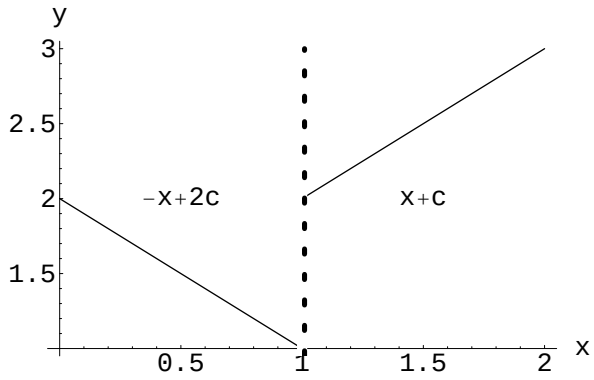
Call
QuantifiedConstraintSystemSolver[Lemma["plim"], OnlyOuterWitness → True]
Output
$\{\text{True}, \{\lambda \to \frac{a+b}{2}\}\}$
Explanation
Here we investigate the limit of a parametric function at infinity. The limit exists and is depending on the parameteres: $\lambda = \frac{a+b}{2}$.

5.5.10 Local Continuity for a Piecewise Function

Input Lemma["LC2",

$$\underset{a,c\in\mathbb{R}}{\exists}\underset{\epsilon>0}{\underset{\epsilon\in\mathbb{R}}{\forall}}\underset{\delta>0}{\underset{\delta\in\mathbb{R}}{\exists}}\underset{x\in\mathbb{R}}{\forall}(|x-1| < \delta) \Longrightarrow \left(\left|\left(\begin{cases} -x+2c & x \leq 1 \\ x+c & x > 1 \end{cases}\right) - a\right| < \epsilon\right) \Bigg]$$

Call
QuantifiedConstraintSystemSolver[Lemma["LC2"], OnlyOuterWitness → True]
Output $\{\text{True}, \{a \to 3, c \to 2\}\}$
Explanation Here we demonstrate a problem which involves a piecewise polynomial. Note that the defining expression of the function is given in grid form. In every row there is term t and a condition C in this order. This polynomial depends on the paramater c. We have to choose the parameter c that the function has a limit at 1. Thus, we have to link two linear functions in such a way that the (left) limit of the first function is the same as the (right) limit of the second function. The solver finds the unique parameter value and the limit of the function only by applying the definiton for the limit.



5.5.11 Uniform Continuity

Input

$$\text{Lemma}\left["\text{UC}", \underset{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}}{\exists}\ \underset{\substack{\delta\in\mathbb{R}\\ 0<\delta<1}}{\forall}\ \underset{x,y\in\mathbb{R}}{\exists}\left((|x-y|<\delta)\wedge\left(\left|x^2-y^2\right|\geq\epsilon\right)\right)\right]$$

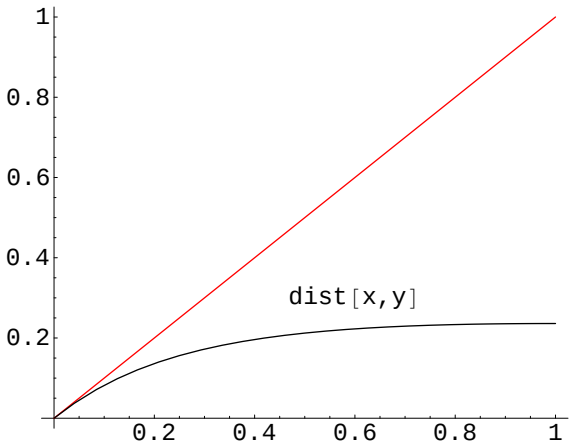
Call

QuantifiedConstraintSystemSolver[Lemma["UC"]]

Output $\left\{\text{True}, \left\{\epsilon\to 1, x\to -\frac{(1+\delta)^2}{2\delta}, y\to -\sqrt{1+\frac{(1+\delta)^4}{4\delta^2}}\right\}\right\}$

Explanation

The input formula formulates that the function $f[x] = x^2$ is not uniformly continuous on the real line, see [Vajd09b]. First, witness term for ϵ is generated: $\epsilon \to 1$. Afterward, by backsubstitution of the $\epsilon-$witness, witnesses for x and y are generated. In fact δ here is restricted ($\delta < 1$) to save place in the presentation. The Solver succeeds without the restriction, too, but generates (finitely many) different witness terms for the different δ-branches, e.g. we will have a different witness term for $\delta < 1$ and for $\delta > 3$. Thus, these witness terms are given in the form of a case disctinction, as in the example 5.5.6. It is interesting to observe that the distance between the generated x, y-pairs is indeed less than δ and the distance of their squares is exactly 1.



5.5.12 Lemma for Lim+

Input

Lemma["LIM+",

$$\underset{a1,a2\in\mathbb{R}}{\forall}\ \underset{a0\in\mathbb{R}}{\exists}\ \underset{\substack{\epsilon1,\epsilon2\in\mathbb{R}\\ \epsilon1>0\wedge\epsilon2>0}}{\forall}\ \underset{\epsilon0\in\mathbb{R}\epsilon0>0}{\exists}\ \underset{x1,x2\in\mathbb{R}}{\forall}((|x1-a1|<\epsilon1\wedge|x2-a2|<\epsilon2)\Longrightarrow(|(x1+x2)-a0|<\epsilon0))]$$

Call

QuantifiedConstraintSystemSolver[Lemma["LIM+"]]

Output

$\{\text{True},\{a0\to a1+a2,\epsilon1\to\frac{\epsilon0}{2},\epsilon2\to\frac{\epsilon0}{2}\}\}$

Explanation

The input formula is generated by the conjecture generator of the S-Decomposition prover for the LIM+ problem, see [Vajd09a]. Since the default options are set, witness terms both for the first and second quantifier blocks will be generated.

Chapter 6

Related Work

6.1 Challenge Problems

Already in the early nineties Bledsoe collected some challenge problems for automated reasoning in elementary analysis [Bled90]. It contained for example our example, CONT+ as well. The TPTP library (Thousands of Problems for Theorem Provers) [SuSu98] is a standard library of test problems for automated theorem proving, including a section also for analysis problems. Problems involving continuity of particular functions or Cont+, or CONT* are considered to be difficult for those resultion based systems, because they have no built-in computational facilities, only the inference rules for standard logical calculi. Furthermore, the proving problems are presented in such a way that one is interested to prove an isolated proposition from a given field. In contrast to this, Buchberger proposed a systematic theory exploration paradigm[Buch99]: Here one is interested to prove a whole group of propositions from a given field by invesitigating the possible interactions of the introduced functions and predicates. The problems are systematically layered and structured and the next level proofs are built on top of the already avaialable propositions.

From the nineties, researchers realized that for reasoning in practice the combination of computation (solvers) and inferencing (provers) is fruitful way for attacking the challenge problems.

6.2 Verify+Reduce

We find a description of an interactive calculus prover for continuity properties in [SuTa89]. This was one of the first attempts to integrate deduction and algebra. The formal system is many sorted classical logic. The system has 5 sorts: reals, pairs of reals, univariate and bivariate real functions and the powerset of the reals. Axioms for sets and ordered fields and for sets and functions are present. We should emphasize that with the system only interactive proving is possible: In a deduction, one has to type in a formula containing inequalities and the

module for the ordered fields (OF) try to prove it. If it succeeds, the formula entered is added to the assumption list. For instance, if one enters the formula $\epsilon/2 > 0$ and $\epsilon > 0$ is already in the list of assumptions, then after validity-check, the formula $\epsilon/2 > 0$ is added to the assumptions. Two main examples are given in the paper describing the system: CONT+ and MVT. MVT stands for the mean value theorem of univariate calculus. The latter is proven based on explicitly stated, additional lemmata.

6.3 Analytica

The Analytica prover was designed to prove theorems in elementary analysis and it was also implemented on the top of Mathematica [BCZh98] as Theorema. Analytica consists of four different phases: Skolemization, simplification, inference and rewriting; However, in contrast to the Theorema provers, proving is not possible for problems which involve quantifiers. Rather, lemmata of "rewrite style" are assumed as a starting point for quantifier free proof problems. Thus, this prover cannot handle problems like CONT+, or CONT*, but was able to prove some remarkable results like the proof of Weierstrass' example of a continuous, nowhere-differentiable function.

6.4 Omega+CoSIE

In Saarbruecken, The ActiveMath group is also interested in providing e-learning environments. ActiveMath is a web-based, multi-lingual user-adaptive learning system for mathematics[1]. Erica Melis investigated the Limit-Domain [Meli98]. They apply proof planning for solving proof problems in elementary analysis. The proof planning operates at a more abstract level as the level of standard level of logic calculus inferences rules. They use operators which represent complex inference rules, also control-rules and domain-specific constraint solvers. They developed constraint solvers which meet the particular requirements for proof-planning [ZiMe04]. Since in [Meli98] a linear constraint solver over the reals was used, the author does not know whether they could handle nonlinear (polynomial) inequality systems generally.

6.5 Mathpert

The work presented in [Bees98] by Beeson is very close to PCS spirit. Gentzen calculus and computational methods for quantifier-free problems are combined in order to prove elementary analysis proofs. Metavariables are introduced and handled exactly in the same way as we used it. For the reduction of the proving situation many built-in operations are used. Typical operation is the Factor-Bounding, which is a similar rule for handling inequalities containing products,

[1] http://www.activemath.org

as one of the control rules of Omega (ComplexEstimate). However the final witnesses will be not generated by CAD, but rather other operations like inequality chaining or upper and lower bound estimates are responsible to come up with the solving terms. They present the proof for the uniform continuity of the function $f[x] = x^3$ and $g[x] = \sqrt{x}$, but the latter uses the mean value theorem. For the discussion about the sense of doing so, see [Bees98, p. 70]

6.6 Automation of Logic Group in MPII

We do not close this section by describing another proving system which uses algebraic techniques for achieving efficient reasoning in practice, but point to a theoretical approach which can be interesting for future extensions of our proving system. V. Sofronie-Stokkermans from the Max Planck Institue for Informatics in Saarbruecken has studied reasoning in local theory extensions [Sofr05]. She shows that efficient hiearchic reasoning is possible for special types of extensions of a base theory.

Assume that the base theory is the theory of reals R. One can consider the elementary analysis proving problems, like the problem of proving Lipschitz continuity of the sum of two Lipshitz continuous functions, as an extension of the theory of reals R with new function symbols satisfying certain axioms.

The approach reduces the proving problem to the unsatisfiability of real constraints gained from the initial clauses by Skolemization, flattening and relational translation. Flattening eliminates the function symbols by introducing new variables and thus, similar to the technique described by us in 4.4.2 for generating conjectures from failing proofs.

If the extension is local, then the initial proof situation is valid if and only if the constructed real constraint system is unsatisfiable. The unsatisfiability of the nonlinear real constraints are checked by RQE. In practice, for real contraints associated with the Lipschitz continuity example, the REDLOG system[DoSt97] was used by the author[Sofr05].

Appendix A

RQE Based on CAD: Sketch of the Initial Algorithm

A.1 Introduction

As we already briefly noticed at the beginning of Section 3, the basic idea of the Collins' real quantifier elimination algorithm is as follows: Wlog we can assume that the arbitrarily quantified input formula ϕ is in prenex normal form and the right hand sides of the equalities and inequalities occurring in the quantifier-free matrix of ϕ are reduced to 0. We extract all the multivariate polynomials from the formula and if r is the number of variables, then we decompose the r-space into disjoint connected subsets such that each extracted polynomial has a constant sign (positive, zero or negative) on each subset. The decomposition will have two important properties: it will be *algebraic* and *cylindrical*. This decomposition permits us to decide the truth conditions of the quantified input formula by using finitely many sample points (we pick samples from each component of the decomposition). The algorithm is referred as Cylindrical Algebraic Decomposition (or with the abbreviation CAD) in the literature. The algorithm consists of three main phases: projection, base case (real root isolation and the decomposition of the real line) and lifting. The algorithm is recursive: In order to get the desired decomposition of $\mathbf{R^r}$, we need to construct decompositions of $\mathbf{R}^{r-1}, \cdots, \mathbf{R}^2, \mathbf{R}^1$ one after the other. We will see that exact symbolic computation is essential. The algorithm is doubly exponential in the number of variables r; fixing r, it is polynomial in the number of the involved polynomials and in the maximum degree of the involved polynomials.
After this subsection, we will be able to completely solve a RQE problem like

$$\exists_y (x^2 + y^2 \leq 1 \wedge y > x).$$

sectionTechnical Preliminaries: Computation with Algebraic Numbers

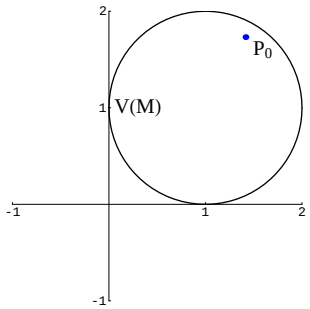
Let $\mathbf{Q}$ be the field of rational numbers, and $\overline{\mathbf{Q}}$ the field of real algebraic numbers. We start the discussion with the following subproblem. Given a multivariate polynomial $M \in \mathbf{Q}[x_1, \cdots, x_r]$, and a point $P_0 \in \overline{\mathbf{Q}}^r$. Determine the sign of M at point $P_0 = (\alpha_1, \cdots, \alpha_r)$, where each α_i is algebraic, i.e., $\alpha_i \in \overline{\mathbf{Q}}$. The solution of the subproblem will be crucial in the third, lifting phase of the CAD-algorithm.

Solution. Each coordinate of P_0 can be represented with a defining polynomial and with an isolating interval. We give a top-down approach for solving the problem.

- Step 1: Compute a primitive element γ, i.e., $\mathbf{Q}(\gamma) = \mathbf{Q}(\alpha_1) \cdots (\alpha_r)$ and representations $(A_i[\gamma])$ for the α_i's. Additionally, let f be a minimal polynomial of and (a, b) be an isolating interval for γ.
- Step 2: Construct the univariate polynomial $g[x] = M[A_1[x], \cdots, A_r[x]]$.
- Step 3: Construct a generalized Sturm sequence for f and g.
- Step 4: Compute the sign variation in (a, b) to determine the sign of M at P_0.

Example. Let $M = (x-1)^2 + (y-1)^2 - 1, P_0 = (\sqrt{2}, \sqrt{3})$.

Step 1: Since the representations for α_1 and α_2 are $\{p = x^2 - 2, (1, 2)\}$ and $\{q = x^2 - 3, (1, 2)\}$ resp., the primitive element can be chosen to $\gamma = \alpha_1 + \alpha_2 = \sqrt{2} + \sqrt{3}$, where $f[x] = x^4 - 10x^2 + 1$ is the minimal polynomial and $(a, b) = (1, 5)$ is an isolating interval for $= \gamma$. $A_1[\gamma] = \frac{1}{2}(\gamma^3 - 9\gamma)$ and $A_2[\gamma] = -\frac{1}{2}(\gamma^3 - 11\gamma)$ are the representations for α_1 and α_2.
Step 2: $g[x] = \frac{1}{2}x^6 - 10x^4 + \frac{101}{2}x^2 - 2x + 1$.
Step 3: We notice that f and g are relatively prime and squarefree. Generalized Sturm sequence for f and g: $\{x^4 - 10x^2 + 1, 2x^9 - 50x^7 + 402x^5 - 8x^4 - 1006x^3 + 40x^2 - 20x, -x^4 + 10x^2 - 1, -24x^3 + 40x^2 + 120x - 8, -\frac{20}{9}x^2 + 8x + \frac{4}{9}, \frac{1296}{25}x + \frac{432}{25}, \frac{200}{81}\}$
Step 4: $SV(a) : \{-, -, +, +, +, +, +\}$; $SV(b) : \{+, +, -, -, -, +, +\}$. $SV(a) - SV(b) = -1$, thus M is **negative** at P_0.



Remarks. For computing primitive elements we can use resultants [Wink96, p. 144], or Gröbner Basis [AdLo94, p. 97]. For generalized Sturm-sequences we refer to [Weis06, p. 90].

A.2 Definitions – CAD

Definition (Decomposition) We call a nonempty connected subset of $\mathbf{R}^r$ a region. A decomposition D of $X \subseteq \mathbf{R}^r$ is a finite disjunct collection of regions such that the union of the disjunct regions is X ($D = \{D_1, \cdots, D_\mu\}$; $\cup_i D_i = X$). We call an element of the decomposition a cell. The sample point of a cell is its arbitrary element.

Definition (Sign Invariance) Let A be a finite set of polynomials (with integral coefficients). We say that that a decomposition is A-invariant, if all $a \in A$ has constant sign on each cell.

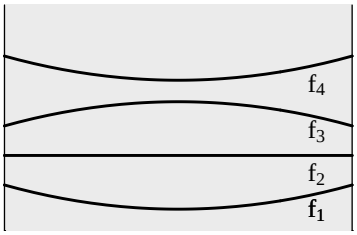
Definition (Cylinder) A cylinder over a region R is $Z(R) = R \times \mathbf{R}$.

Definition (Stack) Let $f_1, \cdots, f_k$ be real continuous functions defined on the region R. The decomposition of $Z(R)$ determined by the functions $f_1 < \cdots < f_k$ consists of (f_{i-1}, f_i)-sectors and f_i sections. Such a decompositon is also called a stack determined by the f_i's over the region R. (cf. figure below).

Definition (Cylindrical Decomposition) We say that the decomposition D of $\mathbf{R}^r$ is cylindrical, if either $r = 1$ and $D = \{D_1, \cdots D_{2\nu+1}\}$, where $D_{2i} = \{\alpha_i\}$, $\alpha_1 < \cdots < \alpha_\nu$, $\alpha_i \in \mathbf{R}$ and $D_{2i+1} = (\alpha_i, \alpha_{i+1})$ or
$r > 1$ and there is a cylindrical decomposition $D' = \{D_1, \cdots, D_\mu\}$ of $\mathbf{R}^{r-1}$ such that $D = \{D_{1,1} \cdots, D_{1,2\nu_1+1}, \cdots, D_{\mu,1} \cdots, D_{\mu,2\nu_\mu+1}\}$ and for each index i $\{D_{i,1} \cdots, D_{i,2\nu_i+1}\}$ is a stack over D_i.

Definition (Algebraic Decomposition) A decomposition is said to be algebraic if the defining functions are all algebraic functions, i.e., solutions of polynomial equations.

Definition (CAD) A decomposition of $\mathbf{R}^r$ is a cylindrical algebraic decomposition, if it is cylindrical and algebraic.

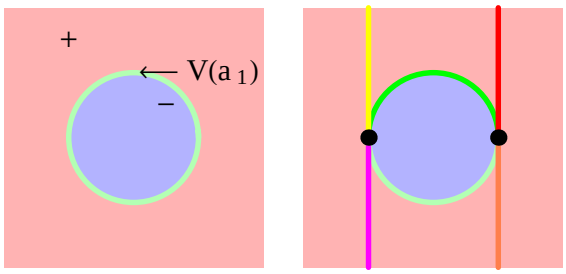


Stack Construction

Remark. Fortunately, today there are already quite a few books and articles discussing the CAD algorithm at elementary level. The author of the current thesis found – among many others – the following materials useful for studying the basic algorithm: [ACC98, Brow04, Dave85, Jirs95, Mish93, Weis06, Wink96]. Moreover, the implementations available in QEPCAD [Brow03], Mathematica [Strz99] and REDLOG [DoSt97] made the experimentations with the method very enjoyable.

A.3 Example

The next example shows two algebraic decompositions of the 2D space.



Let us consider first the figure on the left hand side. The first decomposition is induced only by one polynomial: $a_1[x, y] = x^2 + y^2 - 1$ ($A = \{a_1\}$). This can be considered as a *natural* decomposition, since it slices the plane into such cells, on which the sign of the polynomial a_1 is constant and the connected subsets are maximal in the sense that one cannot enlarge any of the cells without violating the sign invariance property. This decomposition is algebraic, but not cylindrical. The decomposition of the right hand side is cylindrical. We have 5 stacks there and the decomposition consists of 13 cells (notice that the two points, the four half-infinite line segments and the two half of $V(a_1)$ without their endpoints are separate cells of the decomposition; the remaining 5 cells are two dimensional). How can we make a natural decomposition induced by a_1 cylindrical? Roughly speaking, if we add the polynomials $a_2[x, y] = x + 1$ and $a_3[x, y] = x - 1$ to A then the natural decomposition induced by $A' = \{a_1, a_2, a_3\}$ will be exactly the decomposition on the right hand side, thus, it will be cylindrical as well. The polynomials in A' define implicitly a cylindrical algebraic decomposition. To define later the CAD explicitly, we will construct sample points for each cell and give the signs of each a_i on every cell. But first let us solve the problem in 1D to see if we can learn something from the solution for the solution in higher dimensions.

A.4 Base Case

Let us assume that the set of univariate polynomials $A = \{a_1, \cdots, a_n\}$ ($a_i \in \mathbf{Z}[x]$) are given. For the construction of the A-invariant decomposition of the 1D space it is sufficient to consider a squarefree basis $A' = \{a'_1, \cdots, a'_m\}$ for A. After separating the real roots of $pa = a'_1 \cdots a'_m$, the decomposition is available: take the distinct roots in ascending order, and generate indices, samples, and signs of the polynomials as follows:

- If pa has ν real roots $\alpha_1 < \cdots < \alpha_\nu$, then the cells consisting the distinct roots will get even indices starting from 2, and the 1-dimensional, open intervals get odd indices starting from 1 from the left. Assume that ith root is separated by the open interval (l_i, r_i) with rational endpoints. If

the root is rational the closed interval $[\alpha_i, \alpha_i]$ can be considered as an isolating interval as well.

- If we have $D = \{D_1, D_2, \cdots, D_{2\nu+1}\} = \{(-\infty, \alpha_1), \{\alpha_1\}, \cdots, (\alpha_\nu, \infty)\}$, take samples $l_1 - 1, \alpha_1, \frac{r_1+l_2}{2}, \cdots, r_\nu + 1$.
- Determining the sign of a polynomial a'_j on a cell is easy, because it changes sign if and only if the cell D_i contains one of its real root.

For real root isolation for univariate polynomials over $\mathbf{Z}$ one can use the classical Sturm sequences or methods based on sign evaluation of the derivatives[CoLo76].

A.5 Example

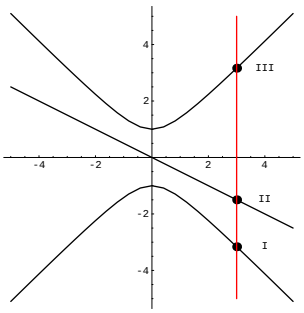
Assume $A = \{a_1, a_2, a_3\} = \{2x^2-1, x-1, x+1\}$. They are already pw. relatively prime and squarefree. Thus $pa = (2x^2 - 1)(x + 1)(x - 1)$ and it has four real roots, i.e., $a_1 = -1, a_2 = -\frac{1}{\sqrt{2}}, \alpha_3 = \frac{1}{\sqrt{2}}, \alpha_4 = 1$. The decomposition consists of 9 cells. The following table sums up the informations for the decomposition.

	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
	$(-\infty, -1)$	$\{-1\}$	$(-1, -\frac{1}{\sqrt{2}})$	$\{-\frac{1}{\sqrt{2}}\}$	$(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$	$\{\frac{1}{\sqrt{2}}\}$	$(\frac{1}{\sqrt{2}}, 1)$	$\{1\}$	$(1, \infty)$
S :	-2	-1	$-\frac{7}{8}$	$-\frac{1}{\sqrt{2}}$	0	$\frac{1}{\sqrt{2}}$	$\frac{7}{8}$	1	2
a_1 :	$+$	$+$	$+$	0	$-$	0	$+$	$+$	$+$
a_2 :	$-$	0	$+$	$+$	$+$	$+$	$+$	$+$	$+$
a_3 :	$-$	$-$	$-$	$-$	$-$	$-$	$-$	0	$+$

Remarks. In the example above two of the four roots were rational, but this was a mere luck. In general, if we decompose $\mathbf{R}^r$, a sample point from an n-dimensional cell has $r - n$ algebraic coordinates. The two other roots in the example were not rational, but they were expressible with radicals. If this is not the case, we have to represent the real roots with defining polynomials and isolating intervals (see A.1). We see that one important ingredient of the method for the decomposition is that we are able to enumerate all the (finitely many) real roots of the polynomials in A in a natural order. How can we generalize that for higher dimensions? This will lead us to the notion of delineability and to a decomposition where the cells are arranged in a certain cylindrical manner.

A.6 Delineability

Consider the polynomial set $A = \{a_1, a_2\} = \{y^2 - x^2 - 1, 2y + x\}$. It is clear that there are infinite points with coordinates $(\hat{x}, \hat{y})$ in the plane such that $a_1[\hat{x}, \hat{y}] = 0$ or $a_2[\hat{x}, \hat{y}] = 0$. But for a fixed x_0, there are only finitely many y's, such that $a_1[x_0, y] = 0$ or $a_2[x_0, y] = 0$. Moreover, the corresponding real y's can be enumarated 'from below to above':



It is said that the real roots of the polynomials in A are *delineable* over the real line. We notice that no matter, how we pick a point x_0 from the x-axes, the number of the corresponding real roots over x_0 is constant, i.e., it is always 3. The points in $V = V(a_1) \cup V(a_2)$ gives rise to a stack construction naturally. The cylinder over $\mathbf{R}$ can be seen as the union of the disjoint portions of V and the two dimensional sectors between them.

In general, the roots of a polynomial set A consisting of bivariate polynomials with integral coefficients will be not delineable over $\mathbf{R}$, but a decomposition of a line into smaller portions (points and intervals) can always be achieved such that the roots of A will be delineable over the smaller portions. If we reconsider the first example (A.3), we see that roots of a_1 is not delineable over $\mathbf{R}$, but they are delinable over the sets $(-\infty, -1), \{-1\}, (-1, 1), \{1\}$ and $(1, \infty)$ as the picture to the right shows. The five stacks defines a cylindrical A-invariant decomposition, because the sign of a polynomial remains unchanged in a sector and in a section inside a cylinder.

Now how can we detect whether the roots are delinable over $\mathbf{R}$ and if not, how can we construct such a decomposition of the line into smaller portions? The projection phase of the CAD algorithm answers the question. Roughly, delineability can be only violated in 2D, if the algebraic curves defined by the a_i's cross or they have self-crossings or vertical tangents or asymptotes. These critical points can be detected investigating the leading coefficients, discriminants and resultants. From the technical point of view, the case where some of the polynomials in A vanishes completely over a region R, has to be handled separately, but we will not go into the details here (see. e.g., [ACC98, p. 141])

A.7 Projection

For the sake of simplicity, we consider here only the 2D problem. For higher dimensions, one would need subresultant computations[ACC98, Wink96]. The solution in 2D is somewhat simpler. As before, let us assume that the finite polynomial set $A = \{a_1, \cdots, a_s\}$ ($a_i \in \mathbf{Z}[x, y]$) is given and consider all a_i as a univariate polynomial over $\mathbf{Z}[x]$ (y is the main variable). To achieve an A-invariant cylindrical decomposition, we form

- the leading coefficient of each polynomial,
- the discriminant of each polynomial,

- and the resultant of each polynomial pairs.

$$P(A) = \{\mathrm{LC}_y(a_i)\} \cup \{\mathrm{DISCR}_y(a_i)\} \cup \{\mathrm{RES}_y(a_i, a_j)\} \ (i \neq j)$$

$P(A)$ is said to be the (first) projection set, and set of their (irreducible) factors is the projection factor set. Superfluous elements of $P(A)$ (e.g. one with no real root) will be removed.

A.8 Example

Let $A = \{a_1, a_2, a_3\}$ where $a_1[x, y] = y^2+x^2-1, a_2[x, y] = y-x, a_3[x, y] = xy-2$.

$$P(A) = \{x\} \cup \{x^2 - 1\} \cup \{2x^2 - 1, x^2 - 2\}$$

There are seven real roots and the decomposition of $\mathbf{R}$ has 15 cells.

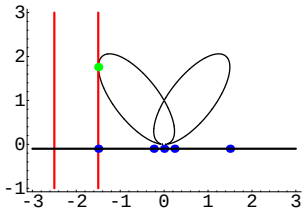
A.9 Lifting

As we discussed in A.1, computation with algebraic numbers will be crucial. The basic idea is that we will extend the decomposition D' of the real line $\mathbf{R}$ to a two dimensional CAD D. In the general case, the decomposition of $\mathbf{R}^2$ will be further extended, until we reach $\mathbf{R}^r$. Once more, for the sake of simplicity, we consider here only the 2D case. Thus, let us assume that we know the set $A = \{a_1, \cdots, a_n\}$ of the initial bivariate polynomials, the projection factor set and the induced decomposition of the real line D'. A sample point s_i of a cell D'_i in D' is either rational or algebraic. In both cases, we will construct the stack over each region D'_i using the sample point s_i. Backsubstitution of s_i into the variable x for each a_j yields univariate polynomials. The real roots of the those polynomials will define a 1D CAD over s_i. Thus, after projection the original higher dimensional problem reduced to many one dimensional problems.

If the sample s_i is rational then there is no technical difficulty because the coefficients of the univariate polynomials are rationals (integers) as well. If the s_i is algebraic then the coefficients of the resulting univariate polynomials can be algebraic and that can require expensive computations for determining primitive elements, isolating roots and evaluating signs.

A.10 Example

In the following example will we consider the CAD induced by the tacnode curve, i.e. $a_1[x, y] = y^4 - 2y^3 + y^2 - 3x^2y + 2x^4$.



After the first two phases of the CAD algorithm (projection and base case) the following informations are available:

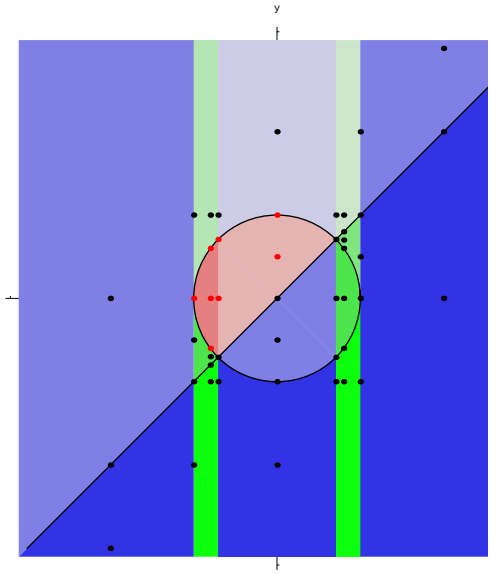
- the projection factors are $\mathrm{pf}_1[x] = x$ and $\mathrm{pf}_2[x] = 2048x^6 - 4608x^4 + 37x^2 + 12$,
- the second factor has four real roots $\alpha_1 < \cdots < \alpha_4$ ($\approx -1.5, -0.24, 0.24, 1.5$) with isolating intervals $(-\frac{3}{2}, -\frac{5}{4}), (-\frac{1}{4}, -\frac{7}{32}), (\frac{7}{32}, \frac{1}{4}), (\frac{5}{4}, \frac{3}{2})$,
- the sample points are $S = \{-\frac{5}{2}, \alpha_1, -\frac{3}{4}, \alpha_2, -\frac{7}{64}, 0, \frac{7}{64}, \alpha_3, \frac{3}{4}, \alpha_4, \frac{5}{2}\}$,
- $D' = \{(-\infty, \alpha_1), \{\alpha_1\}, (\alpha_1, \alpha_2), \{\alpha_2\}, (\alpha_2, 0), \{0, 0\}, \cdots, (\alpha_4, \infty)\}$. Thus the induced decomposition D' has 11 cells.

Now we illustrate the stack construction over a rational and over an algebraic point. To construct the stack over the first cell $(-\infty, \alpha_1)$, we take the sample $s_1 = -\frac{5}{2}$. $a_1[-\frac{5}{2}, y] = y^4 - 2y^3 + y^2 - \frac{75}{4}y + \frac{625}{8}$. This polynomial has no real root. The whole cylinder consists only one cell, a sample is $s_{1,1} = (-\frac{5}{2}, 0)$. a_1 has positive sign in $D_{1,1}$.
Now take $s_2 = \alpha_1$. Taking $q = \mathrm{res}_x[a_1, \mathrm{pf}_2]$, we get a univariate polynomial with five real roots $\beta_1 < \cdots < \beta_5$, among them with the real roots of $a_1[\alpha_1, y] \in \mathbf{Q}(\alpha_1)[y]$. It turns out, as the picture also indicates, that from the five real roots of q only one is the real root of the polynomial $a_1[\alpha_1, y]$ as well: the one with the isolating interval $(\frac{7}{4}, 2)$, i.e., β_5. Thus the second stack consists of 3 cells with samples $s_{2,1} = (\alpha_1, \frac{3}{4}), s_{2,2} = (\alpha_1, \beta_5), s_{2,3} = (\alpha_1, 3)$. a_1 is positive, 0, and positive on the cells $D_{2,1}, D_{2,2}$ and $D_{2,3}$ resp.
The construction of the remaining stacks are similar. There are 55 cells and 11 stacks in the decomposition D alltogether.

A.11 Formula Construction

For solving the RQE problem completely using CAD, we are left with the final problem of constructing a quantifier-free formula for the input formula ϕ. Although the constructed CAD induced by the polynomials involved in the input formula would immediately make it possible to handle all the universal and existential quantifiers in one stroke, we consider here the case where only one existential quantifier should be eliminated.

The formula in the introductory example in A.1 has exactly one existential quantifier. After reducing the right hand side of the inequalities we see that involved polynomials are $a_1[x,y] = -y^2 - x^2 + 1, a_2[x,y] = y - x$, the projection factors are $2x^2 - 1, x + 1, x - 1$ (see A.5). The induced decomposition D' of $\mathbf{R}$ has 9 cells and D has 47 cells:



From the 47 cells only on the cells indexed by $(2,4),(3,4),(3,5),(3,6),(4,3),(4,4),(5,5),(5,6)$ is a_1 nonnegative and a_2 is positive. Now we project away the variable y, i.e., we take only the set of indices of the first coordinates: $\{2,3,4,5\}$. Can we define the cells indexed by $2,3,4,5$ in D'? An obvious answer is $-1 \leq x \wedge x < \frac{1}{\sqrt{2}}$, but this defining formula is not contained in the original signature, i.e., we cannot refer to an indexed root of a univariate polynomial like the second root of the polynomial $2x^2 - 1$ $\left(\frac{1}{\sqrt{2}}\right)$ in the language of L_{or}. However, one can show that taking all the (higher) derivatives of the polynomials contained in the projection factor set, one can define all the cells with the aid of signs of the polynomials in the extended projection set (here we add the polynomial $p[x] = x$ to the set) in L_{or}, e.g. the following formula is already compatible with the original signature and equivalent to the existential ϕ:

$$(2x^2 - 1 < 0) \vee (2x^2 - 1 = 0 \wedge x < 0) \vee (2x^2 - 1 > 0 \wedge x < 0 \wedge x + 1 \geq 0).$$

Thus the RQE problem is solved. However, we note that solutions expressed in the extended language is important for solving the extended quantifier elimination problem.

A.12 Final Remarks

The original algorithm by Collins is now 35 years old [Coll75]. In fact, initially the CAD and the RQE was two distinct algorithms. The first complete implementation of the CAD method was carried out by Collins' Ph.D. student, S. D. Arnon. In the past decades several researchers tried to improve the original CAD-based RQE algorithm. These improvements are various type: they include the partial CAD by Hong, better projection operators and root isolation or fast algorithms for computing with algebraic numbers. Validated numerical computations a parallelism are also considered recently.

Appendix B

Systems for Practical Quantifier Elimination

We will briefly present the available systems which perform real quantifier elimination. The author is familiar with the following three systems:

- QEPCAD,
- REDLOG,
- Mathematica.

We choose a QE example which is tractable with any of the systems. The input formula of the QE problem is a particular case of the Solotareff second problem discussed in Appendix C; namely, the QE formula is:

$$\underset{x,a_1}{\forall} \; (-1 \leq x \leq 1 \Rightarrow \underset{y}{\exists} \; (-1 \leq y \leq 1 \wedge (x^2 + x - a)^2 \leq (y^2 + y - a_1)^2))$$

B.1 QEPCAD B

QEPCAD stands for Quantifier Elimination by Partial Cylindrical Algebraic Decomposition.

- **Version:** QEPCAD B 1.48
- **Authors:** G.E. Collins, H. Hong, C. Brown and others
- **Availability:** open source, downloadbale from
  `http://www.cs.usna.edu/ qepcad/B/QEPCAD.html`

- **System Description:** QEPCAD is an implementation of quantifier elimination by partial cylindrical algebraic decomposition due originally to Hoon Hong, and subsequently added on to by many others. It is an interactive command-line program written in C/C++, and based on the SACLIB library. QEPCAD B version 1.x is a substantial departure from the original QEPCAD developed by Chris Brown. QEPCAD and the SACLIB library are the result of a program of research by George Collins and his PhD students that has spanned several decades.

Problem solving with the program.
Input: `(Ax)(Aa1)(Ey)[[-1<=x /\ x<=1] ==>`
`[-1<=y /\ y<=1 /\ (x^2+x-a)^2<=(y^2+y-a1)^2]].`
Output: `8a-7=0`

In fact the user has to give the input formula in prenex form and additionally needs to specify a variable order before the elimination starts (a,x,a1,y). Detailed information is provided by QEPCAD after the projection phase concerning the projection polynomials and their factors.

```
d-proj-factors
```

```
P_1,1=4a+1, P_1,2=8a+3, ...
```

If the extension phase is done, the solution formula is provided by both the standard and in the extended Tarskian language. Besides the solution formula, informations about the cells, stacks, sample points can be extracted by specific commands.

```
d-stat
```

```
Counts for Truth--Invariant CAD Construction Phase:
 Level   1   2    3     4    Total
Stacks   1  37  233  2563   2834
[...]
```

A restricted 2D variant of the program (CAD2D) replaces the symbolic computations by validated numerics for speeding up computations.

B.2 REDLOG

REDLOG stands for Reduce logic system.

- **Version:** REDLOG 3.0
- **Authors:** V. Weispfenning, T. Sturm, A. Dolzmann and others
- **Availability:** open source, downloadbale from

  `http://www.algebra.fim.uni-passau.de/ redlog/downloads/`
- **System Description:** REDLOG provides an extension of the computer algebra system Reduce to a computer logic system implementing symbolic algorithms on first-order formulas wrt. temporarily fixed first-order

languages and theories. REDLOG is designed for working with several languages and theories in the sense of first-order logic. Both a language and a theory make up a context. In addition, a context determines the internal representation of terms. It contains several quantifier elimination algorithms (rlqe).

Problem solving with the program.

Input: `f:=all(x, (-1<=x and x<=1) impl all(a1, ex(y, -1<=y and y<=1 and (x^2+x-a)^2<=(y^2+y-a1)^2))); rlqe(f);`

Output: `8*a-7=0`

After starting the computer algebra system REDUCE one loads the REDLOG package by the command

`load_package redlog;`

The ordered fields context (Ordered Field Standard Form) can be set by

`rlset ofsf;`

Besides the standard real quantifier elimination call above (rlqe), one can explicitly request a CAD-based elimination (rlcad) or a Hermitian quantifier elimination (rlhqe). The REDLOG system provides an integrated approach to real quantifier elimination. If degree condititions are satisfied by the current variable of the input formula, rlqe starts the eliminiation with the virtual substitution (VS) method. If VS is not applicable, then it applies the general partial CAD construction. A heuristic for the optimal variable order is part of the rlqe algorithm, too. Fine tuning for the elimination is done by REDLOG switches. Extended quantifier elimination is avaliabale by using the command rlqea. The answers are given for the variabels in the outermost existential quantifier block. Here is variation of the above problem with answer:

Input: `rlqea((ex,a,f);`

Output: {{`true`,{`a=7/8`}}}

Finally, it is important to mention that integer quantifier elimination is also implemented in REDLOG. The context for Presburger arithemtic (PASF) should be set for performing integer quantifier elimination with REDLOG.

B.3 Mathematica

- **Version:** Mathematica 6.0
- **Authors:** A. Strzebonski
- **Availability:** commercial, `http://wolfram.com`
- **System Description:** Mathematica is one of the best general purpose computer algebra systems. It contains quantifier elimination algorithms for RCF (Resolve).

Problem solving with the program.
Input:

```
f=ForAll[x,-1<=x<=1, ForAll[a1, Exists[y, -1<=y<=1,
(x^2+x-a)^2<=(y^2+y-a1)^2]]]; Resolve[f,{a},Reals]
```

Output: `a=7/8`

As Redlog, Mathematica also provides an integrated approach for real quantifier elimination. Both Virtual Substitution (VS) and CAD-based elimination methods are available. The default settings can be changed by modifying system options. The Resolve command has its own options, but the real key for influencing the performance of the elimination technique is to modify the Inequality solving options. This may be interesting if the next variable can be eliminated by different methods. With the settings below one forces Mathematica to use only CAD for the elimination.

```
Developer‘SetSystemOptions[
"InequalitySolvingOptions" -> "CAD" -> True];
Developer‘SetSystemOptions[
"InequalitySolvingOptions" -> "LinearQE" -> False];
Developer‘SetSystemOptions[
"InequalitySolvingOptions" -> "QuadraticQE" -> False];
```

LinearQE and QuadraticQE here stands for the Virtual Substitution method. With this setting we forbid the usage of VS. In the defualt settings, Mathematica investigates the input formula of the problem and it decides about the method, too. Mathematica has a very effective implementation of the quantifier elimination methods. However, solution formulae are constructed only in the extended Tarskian language.

Appendix C

The Solotareff Approximation Problem

C.1 Introduction

Although our interests is in practical reasoning in elementary analysis, several other application field of quntifier elimination exist. Collins in [Coll98] enumerates among many others robot motion planning, geometry proving, investigation of stability of differential schemes, algebraic curve display, etc. We investigate here only one specific application field, a polynomial approximation problem. We try to learn from the the success and failure of applying the method for examples of that field. We are trying to take into consideration the observed phenomena by the *integration* of the QE methodology into reasoning systems.

C.2 Solotareff Second Problem

C.2.1 Description and Formulations

Solotareff[1] second problem is polynomial approximation problem, see [Coll95, Kalt00, Laza06], where we approximate a real polynomial of degree n with polynomials having degree (at most) $(n-2)$ in the closed interval $[-1,1]$, using the uniform (supremum) norm. The natural question which arises in this setup, whether there exists a best approximation polynomial which satisfies the given conditions. It is maybe astonishing at the very first glance, that the problem can be expressed using the language L_{or} (ordered rings) and thus it is tractable with the quantifier elimination methodology.
To show this, let us begin the following symbolization of the problem:

[1] Other transcriptions like *Zolotarev* is also used

$$\underset{\overline{Q}}{\forall}\,\underset{x}{\forall}\,(-1 \leq x \leq 1) \Rightarrow \|P - Q\| \leq \|P - \overline{Q}\|$$

where P is given polynomial to approximate, Q is the best polynomial which satisfies the approximation conditions and $\overline{Q}$ ranges over the set of all polynomials with degree $(n-2)$. The supremum norm ($\|\cdot\|$) is obviously not part of the signature of the language L_{or} and we are not allowed to quantify in this first order language over polynomials, but it is easy to see, that the condition $\|A\| \leq \|B\|$ is equivalent to the following:

$$\underset{x}{\forall}\ (-1 \leq x \leq 1) \Rightarrow \underset{y}{\exists}\,(-1 \leq y \leq 1) \wedge |A[x]| \leq |B[y]|$$

Furthermore, eliminating the absolute value (e.g. $|x| \leq |y| \Leftrightarrow x^2 \leq y^2$), and introducing $3n-1$ fresh variables ($(n+1)$ bound and $2(n-1)$ free variables for representing the coefficients of P and Q, and resp.), we gain the sought for formalization of the problem. Another simple thought lead us to the conclusion that for arbitrary n, it is sufficient to approximate the following one parameter polynomial family of the form: $x^n + r \cdot x^{n-1}$ $(r \geq 0)$.
So let us formulate the quantifier elimination problem which belongs to the Solotareff approximation problem for $n = 2$:

$$r \geq 0 \wedge \underset{x,a_1}{\forall}\ (-1 \leq x \leq 1 \Rightarrow \underset{y}{\exists}\ (-1 \leq y \leq 1 \wedge (x^2 + rx - a)^2 \leq (y^2 + ry - a_1)^2))$$

We note that using this formalization[2] , the degree two problem involves 5 variables and the degree three problem has 7 variables. However, if we use a different approach and formalization[3] , the number of variables can be reduced, thus making the problem easier.

As one might immediately think, we have to pay a certain price for the reduction of the number of the variables: user interaction is needed in several places, because the second approach solves the optimization problem in several subsequent steps. The second approach asks for the minimum value first and only after finding the minimum, gives the coefficients of the best approximation polynomial in a second phase (by substitution and simplification). Moreover, it also operates directly on the semi-algebraic sets which corresponds to the input formula, namely it takes the inf and sup of the set w.r.t. a specified variable at the suitable phase.
So let us see the input formula which belongs to the Solotareff approximation problem for $n = 2$, using the second approach:

$$\underset{x}{\exists}\ (m > 0 \wedge -1 \leq x \leq 1 \wedge (m = x^2 + rx - (a) \vee -m = x^2 + rx - (a)))$$

[2]This is due to G. E. Collins
[3]This is due to C. Brown

Fixing r, a, here m describes the range of the polynomial $x^2 + rx - a$ in absolute value, i.e., all possible deviations from zero in $[-1, 1]$. After eliminating x, we have a description of a semi-algebraic subset of the 3-space (a, r, m) and we take the sup w.r.t. the variable m. Eliminating (existentially) the variables a, we left with 2-dimensional semi-algebraic set. Taking its inf w.r.t. m , we know the smallest possible value for m which belongs to a fixed r. Backsubstituting the value of m, we get a functional relationship between r and a.
Comparing the two approaches, it is clear that the number of variables reduced for $n = 2$ by 1 and for $n = 3$ by 2.
Before we report about the computations, I would like to emphasize the following items. I was able to solve the general problem for $n = 2$ with the first formalization and for $n = 2, 3$ with the second approach, without any further mathematical knowledge about the best approximation (uniqueness, derivatives, number of extrema etc. of the best polynomial). With the first formalization I was able to solve particular problems for $n = 3$, i.e. for fixed r, for certain intervals like $[\frac{1}{2}, 1]$ or $(100, \infty)$ but failed in general. At the end of subsection we briefly come back on the issue what we can learn from the answers/results, how additional mathematical knowledge can make the problem, which is still a QE problem, easier.

C.2.2 Computations

Problem Degree 2; $n = 2, r \geq 0$ **(general problem) (1A)**

Formulation: First
No. of problems: 1
System(s): Mathematica 5.2
Input (ϕ):
$r \geq 0 \wedge \underset{x,a_1}{\forall} \ (-1 \leq x \leq 1 \Rightarrow \underset{y}{\exists} \ (-1 \leq y \leq 1 \wedge (x^2 + rx - a)^2 \leq (y^2 + ry - a1)^2))$

Output (ψ): $(0 \leq r \leq 2 \wedge a = \frac{1}{8}(4 + 4r - r^2)) \vee (r > 2 \wedge a = 1)$

Call: Resolve[$\phi, \{r, a\}$, Reals]
Remarks: No subproblems, no user interaction. However if we divide the problem into two and eliminate the variables y and a_1 first and only afterwards the variable x, a considerable speedup is achieved. This observation will be crucial for solving the problem with the first formalization for particular degree three polynomials. A different formulation for the general $n = 3$-problem is needed up to my knowledge.
Timing: Mma 5.2, Intel Xeon E5345, 2.33GHZ, 27sec

Problem Degree 2; $n = 2, r \geq 0$ **(general problem) (1B)**

Formulation: Second
No. of problems: 3

System(s): QEPCAD 1.46
Input 1 (ϕ_1):
$\underset{x}{\exists}\ (m > 0 \wedge -1 \leq x \leq 1 \wedge (m = x^2 + rx - (a) \vee -m = x^2 + rx - (a)))$

Assumptions: $r \geq 0$
Output 1 (ψ_1): $m-r-a+1 \geq 0 \wedge m-r+a-1 \geq 0 \wedge ((r-2 > 0 \wedge m-r-a+1 = 0) \vee (r-2 \leq 0 \wedge 4m-r^2-4a = 0) \vee (r-2 > 0 \wedge m-r+a-1 = 0) \wedge (4m-r^2-4a > 0 \wedge m-r+a-1 = 0))$
Variable Order: a, r, m, x
Remark. Lim sup w.r.t m was taken after eliminating x.

Input 2 (ϕ_2): $\underset{a}{\exists}\ \psi_1$

Assumptions: $r \geq 0$
Output 2 (ψ_2): $m - r \geq 0 \wedge 8m - r^2 - 4r - 4 \leq 0 \wedge ((r - 2 \geq 0 \wedge m - r = 0) \vee (r - 2 < 0 \wedge 8m - r^2 - 4r - 4 = 0))$
Variable order r, m, a
Remarks. Lim inf w.r.t m was taken after eliminating a. With the assumption, ψ_2 is equivalent to $(0 \leq r \leq 2 \wedge m = \frac{1}{8}(4 + 4r + r^2)) \vee (r > 2 \wedge m = r)$
Remarks: Several subproblems, user interaction.

Input 3a (ϕ_{3a}): $(0 \leq r \leq 2 \wedge \psi_1/.m \rightarrow \frac{1}{8}(4 + 4r + r^2))$
Output 3a (ψ_{3a}): $0 \leq r \leq 2 \wedge 8a + r^2 - 4r - 4 = 0$
Input 3b (ϕ_{3b}): $(r > 2 \wedge \psi_1/.m \rightarrow r)$
Output 3b(ψ_{3b}): $r - 2 > 0 \wedge a - 1 = 0$
Timing: QEPCAD 1.46, P4 2.8GHZ < 3sec

Problem Degree Three; solving special cases; $n = 3 : r = 0, r = 1, r = 2$ **(particular problems) (2A)**

Formulation: First
No. of problems: 2
System(s): Mathematica 5.2
Input 1($\phi_1, r = 0$) :
$\underset{a_1,b_1}{\forall}\ (-1 \leq x \leq 1 \Rightarrow \underset{y}{\exists}\ (-1 \leq y \leq 1 \wedge (x^3 - (ax + b))^2 \leq (y^3 - (a_1 y + b_1))^2))$

Output 1(ψ_1): $(-1 \leq x \leq 1 \wedge \frac{1}{4}(4x^3 - 4ax - 1) \leq b \leq \frac{1}{4}(4x^3 - 4ax + 1)) \vee x > 1$
Call: Resolve[ϕ_1, Reals]

Input 2 (ϕ_2): $\underset{x}{\forall}\ \psi_1$

Output 2 (ψ_2): $a = \frac{3}{4} \wedge b = 0$
Remarks. A different formulation for the general $n = 3$-problem is needed up to my knowledge.
Timing: Mma 5.2, Intel Xeon E5345, 2.33GHZ, 19sec

Input 1($\phi_1, r = 1$):

$\underset{a_1,b_1}{\forall}\ (-1 \le x \le 1 \Rightarrow \underset{y}{\exists}\ (-1 \le y \le 1 \wedge (x^3+x^2-(ax+b))^2 \le (y^3+y^2-(a_1y+$

$b_1))^2))$
[...]
Output 2 (ψ_2): $a = 1 \wedge b = \frac{11}{27}$
Remark. Details are omitted
Timing: Mma 5.2, Intel Xeon E5345, 2.33GHZ, 62sec

Input 1($\phi_1, r = 2$):
$\underset{a_1,b_1}{\forall}\ (-1 \le x \le 1 \Rightarrow \underset{y}{\exists}\ (-1 \le y \le 1 \wedge (x^3+2x^2-(ax+b))^2 \le (y^3+2y^2-$

$(a_1y+b_1))^2))$
[...]
Output 2 (ψ_2): $a = 1 \wedge b = \frac{1}{27}(44 - 7\sqrt{7})$
Remark. Details are omitted.
Timing: Mma 5.2, Intel Xeon E5345, 2.33GHZ, 566sec

Problem Degree Three; solving special cases; $n = 3 : r = 0, r = 1, r = 2$ **(particular problems) (2B)**

Formulation: Second
No. of problems: 3
System(s): QEPCAD 1.46
Input 1 ($\phi_1, r = 0$):
$\underset{x}{\exists}\ (m > 0 \wedge -1 \le x \le 1 \wedge (m = x^3 - (ax+b) \vee -m = x^3 - (ax+b))$

Output 1 (ψ_1): $m+b-a+1 \ge 0 \wedge m-b+a-1 \ge 0 \wedge m+b+a-1 \ge 0 \wedge m-b-a+1 \ge 0 \wedge ((a-3 \ge 0 \wedge m-b-a+1 = 0) \vee (a-3 \ge 0 \wedge m+b-a+1 = 0) \vee (a-3 < 0 \wedge 27m^2+54bm+27b^2-4a^3 > 0 \wedge 27m^2-54bm+27b^2-4a^3 = 0) \vee (a-3 < 0 \wedge 27m^2-54bm+27b^2-4a^3 \ge 0 \wedge 27m^2+54bm+27b^2-4a^3 = 0) \vee (27m^2-54bm+27b^2-4a^3 \ge 0 \wedge m-b+a-1 = 0) \wedge (27m^2+54bm+27b^2-4a^3 > 0 \wedge m+b+a-1 = 0))$
Variable Order: a, b, m, x
Input 2 (ϕ_2): $\underset{a,b}{\exists}\ \psi_1$

Output 2 (ψ_2): $4m - 1 = 0$
Variable Order: a, b, m
Input 3 (ϕ_3): $\psi_1/.m \to \frac{1}{4}$
Output 3 (ψ_3): $4a - 3 = 0 \wedge 4b - 4a + 3 = 0$
Remark: Output is the same as $a = \frac{3}{4},\ b = 0$.
Timing: QEPCAD 1.46, P4 2.8GHZ < 5sec

Input 1 ($\phi_1, r = 1$):
$\underset{x}{\exists}\ (m > 0 \wedge -1 \le x \le 1 \wedge (m = x^3+x^2-(ax+b) \vee -m = x^3+x^2-(ax+b))$

[...]
Output 3 (ψ_3): $a - 1 = 0 \wedge 27b - 27a + 16 = 0$

Remark: Details are omitted, output is the same as $a = 1,\ b = \frac{11}{27}$.
Timing: QEPCAD 1.46, P4 2.8GHZ < 7sec

Input 1 ($\phi_1, r = 2$):
$\exists_x\ (m > 0 \wedge -1 \leq x \leq 1 \wedge (m = x^3 + 2x^2 - (ax + b) \vee -m = x^3 + 2x^2 - (ax + b))$

[...]
Output 3 (ψ_3): $a - 1 = 0 \wedge 27b - 44 < 0 \wedge 27b^2 - 88b + 59 = 0$
Remark: Details are omitted, output is the same as $a = 1,\ b = \frac{1}{27}(44 - 7\sqrt{7})$.
Timing: QEPCAD 1.46, P4 2.8GHZ 10sec

Problem Degree Three; solving the general problem $n = 3$

Using the second formulation I was able to solve the general problem breaking down the original problem into two cases: I) $0 < r < 1$, II) $r > 1$. In fact, the first case is easier, because the solutions ($a[r]$, $b[r]$) here are polynomial functions, which is not true for the second case. The solution is achieved by combining QEPCAD and Mathematica and the computational time was more than 60 minutes in total. Without reporting the computational details, the following table summarizes the results:

r	m	a	b
$(0,1)$	$\frac{1}{108}(r^3 + 9r^2 + 27r + 27)$	$\frac{1}{4}(-r^2 + 2r + 3)$	$\frac{1}{108}(-r^3 + 18r^2 + 27r)$
$(1,\infty)$	$\frac{1}{27}(-r^3 + 9r + \sqrt{r^6 + 9r^4 + 27r^2 + 27})$	1	$\frac{1}{27}(r^3 + 18r - \sqrt{S})$

S is the same expression as the expression on the third row under the square root.

Adding Mathemtical Knowledge

The computational results obtained by the application of the QE methodology showed the power of the method. However, if we investigate the Solotareff second approximation problem for degree n in detail, it turns out[Coll95, Coll98] that the solution is unique and

a) if $0 \leq r < S_n = \tan^2\left(\frac{\pi}{2n}\right)$ then the coefficients of the solution are rational polynomials in r and they are easily computable in terms of the Chebyshev polynomials.

b) if $r > S_n$, then quantifier elimination is indeed needed, but in this case if E denotes the error function for the optimal polynomial, then there exists points u_i's $(-1 = u_0 < u_1 < \cdots < u_{n-1} < u_n = 1)$ such that the error is maximal in the u_i's, thus $E'[u_i] = 0$ $(i = 1, \cdots, n-1)$ and $E[1] = (-1)^{i-1}E[u_{n-i}]$. Using this additional mathematical background knowledge we can reformulate the general degree three QE problem for $r > 1$ with the first formalization:

$$\exists_u\ r > 1 \wedge -1 < u < 1 \wedge 1 + r - (a + b) = -1 + r - (-a + b) \wedge$$

$$u^3 + ru^2 - (au + b) = -(1 + r - (a + b)) \wedge 3u^2 + 2ru - a = 0.$$

The CAD based QE algorithm gives us the solution

$$a = 1 \wedge b = \frac{1}{27}(r^3 + 18r - \sqrt{r^6 + 9r^4 + 27r^2 + 27})$$

within a second (!), which is the same as we have above. Thus considerable speedup is gained by additional mathematical knowledge (equational constraints). We stop here but refer to [Coll95, Laza06] for further details, if the reader is interested.

C.3 Moral

- The first we can learn from the above elaborated examples that good formalization, the clever choice of the variable order and suitable decompositions of the initial elimination problems into subproblems can bring problems with five or six variables into the scope of the CAD-based method.
- If we design/apply a black box implementation for real quantifier elimination, these aspects must be taken into consideration. The designer should provide specific options for the user to influence the behavior of the general algorithm which does the quantifier elimination, produces witness terms or just decides the satisfiabilty or real polynomial systems. The default options should work for the simplest cases. However, it is important that the user can specify exactly the method for the problem, the time constraint and so on.
- Things get even more complicated, if besides the CAD-based method, other methods as virtual substitution or Hermitian quantifier elimantion is available in an integrated system. Under these assumptions, an experiment was carried out by the author and his colleague for a different problem type from robotics in Mathematica, see [MoVa08].
- Although we were not able to solve the general degree three problem with the first approach, the data that can be collected from investigating special cases, can still be useful. E.g. with the first formalization, we get for the values $r = 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1$, the following list of coefficents for the optimal approximating polynomials $p_r[x] = ax + b$:

  $$a = \left\{\frac{3}{4}, \frac{55}{64}, \frac{15}{16}, \frac{63}{64}, 1\right\} \; b = \left\{0, \frac{503}{6912}, \frac{143}{864}, \frac{71}{256}, \frac{11}{27}\right\}.$$

  Now we can have a polynomial Ansatz. We construct from the data the Lagrange polynomials $L_a[x] = \frac{1}{4}(-r^2 + 2r + 3)$, $L_b[x] = \frac{1}{108}(-r^3 + 18r^2 + 27r)$. Since there degrees are only two and three, one may conjecture that in fact, this polymomial pair will be the general solution for $r \in [0, 1]$. Checking this is already possible even with the first formulation. Checking is always easier than solving: here this is so, because the checking problem contains two less variables:

$$(0 \leq r \leq 1 \wedge -1 \leq x \leq 1) \Rightarrow \underset{y}{\exists} \ (-1 \leq y \leq 1 \wedge (x^3 + rx^2 - (\tfrac{1}{4}(-r^2 + 2r + 3)x + \tfrac{1}{108}(-r^3 + 18r^2 + 27r)))^2 \leq (y^3 + ry^2 - (a_1 y + b_1))^2) \rightarrow \text{True}$$

In a similar way, one can easily conjecture for the second case $r > 1$ that $a = 1$ (constant).

Curriculum Vitae

Affiliation

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, A-4040 Linz, Austria
Robert.Vajda@risc.uni-linz.ac.at

Personal Data

Date and Place of Birth: January 17, 1974, Szeged
Nationality: Hungarian

Education

2004–2009	Doctoral Studies in Symbolic Computation at RISC, JKU, Linz
2001–2004	Doctoral Studies in Mathematics, University of Szeged
1996–2001	Master Degree Studies in Mathematics and Philosophy, Univ. of Szeged
1992–1995	Studies in Mathematics and Computer Science, Univ. of Szeged
1992	High School Diploma (Matura; with distinction)

Teaching Experience

2008-2009	Assistant Professor at the University of Szeged, Analysis Department, Bolyai Institute
2001-2007	Assistant Teacher at the University of Szeged, Analysis Deptartment, Bolyai Institute
Subjects:	Calculus, Numerical Mathematics, Linear Algebra, Computer Supported Mathematical Modeling

Publications

R. Vajda and Z. Kovács: Interactive Web Portals in Mathematics. Teaching Mathematics and Computer Science 1(2) (2003), pp. 347-361, Debrecen

M. K. Németh and R. Vajda: Computer-Assisted Assessment of Mathematical Knowledge. In: Computer Algebra Systems and Dynamic Geometry Systems in Mathematics Teaching. Proceedings of the Sprout-Selecting Conference, 2004, pp. 113-127, Pécs

R. Vajda, T. Jebelean and B. Buchberger: Combining Logical and Algebraic Techniques for Natural Style Proving in Elementary Analysis. Mathematics and Computers in Simulation, Volume 79, Issue 8, April 2009, pp. 2310-2316, Elsevier; doi:10.1016/j.matcom.2008.11.002

R. Vajda: An e-Learning Environment for Elementary Analysis: Combining Computer Algebra, Graphics and Automated Reasoning. Teaching Mathematics and Computer Science 7(1) (2009), pp. 13-34, Debrecen

Acknowledgements

A very long path led to this thesis. Most of the work related to it was done in Linz, Szeged and of course, in Hagenberg. I thank all who were with or behind me in the long journey, even to those colleagues, friends and girlfriends, who could not wait until the final version of the thesis.

First of all, let me thank my advisor, *Bruno Buchberger*, for giving me the chance to be involved in the CreaComp project and the Theorema Group and the RISC Symbolic Computation doctoral programme. I feel myself lucky to have a master who has a clear vision about the connections among algorithmic mathematics, computer science and mathematics education. He gave me a research topic which particularly fitted to my interests. Throughout my Ph.D. study he always found time for inspiring discussions.

I thank my second advisor, *Wolfgang Schlöglmann* for reading through the thesis and giving me hints on the didactical aspects of the thesis.

I thank my colleague, *Tudor Jebelean* for teaching me automated reasoning and for the collaboration on the S-Decomposition method. He even found time during his journeys from Timisoara to Linz to stop at Szeged and have a discussion about automated reasoning with me.

Further, let me thank all the former and current members of The Theorema Group, especially Wolfgang Windsteiger, Nikolai Popov, Teimuraz Kutsia, Camelia Rosenkranz, Markus Rosenkranz, Florina Piroi, Adrian Craciun, Alexander Zapletal, Laura Kovacs and Gabor Kusper for their kind support.

Let me thank the broader RISC community for their patience and support. In spite of the different research interests, they accepted me as colleague. They explained all kinds of algorithms to me or helped me with translations and organisational items or just gave me a ride between Linz and Hagenberg. Especially I want to thank to Manuel Kauers, Veronika Pillwein, Brian Moore, Karoly Erdei, Karoly Bosa, Gabor Guta, Gabor Bodnar for their support during my stays in Linz and Hagenberg.

Finally, let me thank all of my colleagues in Szeged and around the world who were willing to help me during my master and Ph.D. study. I thank Janos Karsai who introduced me to the world of Mathematica. I thank Zoltan Kovacs for WMI, for the joint paper, for the shared moments in the common office and for more. I also thank Chris Brown, Thomas Sturm and Adam Strzebonski for answering all of my questions related to the quantifier elimination implementations in QEPCAD, REDLOG and Mathematica.

Speical thanks goes to my family members for their kind support and patience.

The work presented in this thesis was supported by the Upper Austrian Project CREACOMP, by the FWF Project SFB F013 and by the European Erasmus Teaching Staff Mobility Programme.

Bibliography

[AMWe08] M. Achatz, S. McCallum, V. Weispfenning. Deciding Polynomial-Exponential Problems. Proceedings of the ISSAC 2008 Conference, pp. 213-222, 2008, July, Hagenberg, Austria.

[AdLo94] W. W. Adams, P. Loustaunau. Introduction to Gröbner Bases. Graduate Studies in Mathematics, American Mathematical Society, Providence, R.I., 1994.

[AkPa07] B. Akbarpour, L. C. Paulson. Towards Automatic Proofs of Inequalities Involving Elementary Functions. Proceedings of the 4th Workshop of Pragmatics of Decision Procedures in Automated Reasoning, pp. 33-43, 2006, August, Seattle, USA.

[ACC98] D. S. Arnon, G. E. Collins, S. McCallum. Cylindrical Algebraic Decomposition I: The Basic Algorithm. In: Caviness-Johnson (eds): Quantifier Elimination and Cylindrical Algebraic Decomposition, pp. 136-151, Springer, 1998.

[BCZh98] A. Bauer, E. M. Clarke, X. Zhao. Analytica - An Experiment in Combining Theorem proving and Symbolic Computation. Journal of Automated Reasoning (JAR), 21(3), pp. 295-325, 1998.

[Bees89] M. Beeson. Logic and Computation in Mathpert: An Expert System for Learning Mathematics. Computers and Mathematics, pp. 202-214, Springer Verlag, 1989.

[Bees98] M. Beeson. Automatic Generation of Epsilon-Delta Proofs of Continuity. LNCS/LNAI 1476, pp. 67-83, Springer Verlag, 1998.

[Bees03] M. Beeson. the Mechanization of Mathematics. In: Teuscher C. (ed.) Alan Turing: Life and Legacy of a Great Thinker, Springer Verlag, Berlin, 2003.

[Bled90] W. Bledsoe. Challenge Problems in Elementary Analysis. Journal of Automated Reasoning (JAR) 6(3), 1990, pp. 341-359.

[Borw05] J. M. Borwein. The Experimental Mathematician: The Pleasure of Discovery an the Role of Proof. International Journal of Computers for Mathematical Learning, 10(2), Springer, 2005, pp. 75-108.

[BrMa07] A. R. Bradley, Z. Manna. The Theory of Computation. Decision Procedures with Applications to Verification. Springer, 2007.

[BrMc05] C. W. Brown, S. McCallum. On Using Bi-equational Constraints in CAD Construction. Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation, Beijing, China, pp. 76-83.

[Brow03] C. W. Brown. An Overview of QEPCAD B: A Tool for Real Quantifier Elimination and Formula Simplification. Journal of Japan Society for Symbolic and Algebraic Computation, 10(1):13-22, 2003. See `http://www.cs.usna.edu/~qepcad/B/QEPCAD.html`

[Brow04] C. W. Brown. ISSAC 2004 Tutorial: Cylindrical Algebraic Decomposition.

[BuLi81] B. Buchberger, F. Lichtenberger. Mathematik für Informatik I. Springer Verlag Berlin, 1981.

[Buch85] B. Buchberger. Editorial. Journal of Symbolic Computation (JSC) 1, pp. 1-6, 1985.

[Buch90] B. Buchberger. Should Students Learn Integration Rules? In: ACM SIGSAM Bulletin, Vol. 24/1, pp. 10-17, 1990.

[Buch91] B. Buchberger. Logic for Computer Science. Springer Verlag, Wien New York, 1991.

[BuHo91] B. Buchberger, H. Hong. Speeding-up Quantifier Elimination by Groebner Bases. Technical report no. 91-06 in RISC Report Series, University of Linz, Austria. February 1991.

[Buch96a] B. Buchberger. Symbolic Computation: Computer Algebra and Logic. In: Proceedings of FROCOS 1996 (1st International Workshop on Frontiers of Combining Systems), March 26-28, 1996, Munich, (F. Bader, K.U. Schulz eds.), Applied Logic Series, Vol.3, Kluwer Academic Publisher, Dordrecht - Boston - London, 1996, pp. 193-220.

[Buch96b] B. Buchberger. Mathematiksoftware und Mathematikunterricht: Ein Vorwort. In: Mathematikunterricht mit Computeralgebra-Systemen, H.Heugl, W. Klinger, J. Lechner (ed.), pp. 9-13. 1996. Addison-Wesley, Bonn - Reading

[Buch97a] B. Buchberger. Doing Mathematics by Computer? In: Advances in the Design of Symbolic Computation Systems, A. Miola, M. Temperini (ed.), pp. 2-20. 1997. Springer, RISC Book Series on Symbolic Computation.

[Buch98] B. Buchberger. Theorema: Theorem Proving for the Masses using Mathematica. Invited Talk at the 10 years Mathematica Symposium., Chicago, 1998.

[Buch99] B. Buchberger. Theorem Proving Versus Theory Exploration. Invited talk at the Calculemus Workshop, Univ. of Trento, 1999, Italy.

[Buch00a] B. Buchberger, C. Dupre, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru, W. Windsteiger: The Theorema Project: A Progress Report. In: Symbolic Computation and Automated Reasoning (Proceedings of CALCULEMUS 2000, Symposium on the Integration of Symbolic Computation and Mechanized Reasoning, August 6-7, 2000, St. Andrews, Scotland), M. Kerber and M. Kohlhase (eds.), A.K. Peters, Natick, Massachusetts, pp. 98-113.

[Buch00b] B. Buchberger. Computer-Algebra: Das Ende der Mathematik? Mitteilungen der Deutschen Mathematiker-Vereinigung, 2, pp. 16-19. 2000, Birkhaeuser Verlag AG, Basel, Switzerland

[Buch01] B. Buchberger. The PCS Prover in Theorema. Lecture Notes in Computer Science (LNCS) 2178, pp. 19-23, Springer Verlag, Berlin, 2001.

[Bund84] A. Bundy. The Computer Modelling of Mathematical Reasoning. Academic Press, New York, 1984.

[ClZh93] E. Clarke, X. Zhao. Analytica: A Theorem Prover for Mathematica. The Mathematica Journal 3(1), pp. 56-71, 1993.

[ChLe73] C. Chang, R. Lee. Symbolic Logic and Mechanical Theorem Proving. Academic Press, New York, 1973.

[Coll75] 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.

[Coll95] G. E. Collins. Application of Quantifier Elimination to Solotareff's Approximation Problem. Technical report no. 95-31 in RISC Report Series, University of Linz, Austria. March 1995.

[Coll98] G. E. Collins. Quantifier Elimination by Cylindrical Algebraic Decomposition - Twenty Years of Progress. In: Caviness-Johnson (eds): Quantifier Elimination and Cylindrical Algebraic Decomposition, pp. 8-23, Springer, 1998.

[CoLo76] G.E. Collins, R. Loos. Polynomial Real Root Isolation by Differentiation. Proceedings of the Third ACM Symposium on Symbolic and Algebraic Computation, pp. 15-25, August, 1976, New York, USA.

[Dave85] J. H. Davenport. Computer Algebra for Cylindrical Algebraic Decomposition. Report TRITA-NA-8511, The Royal Institute of Technology, Stockholm, 1985.

[DSW97] 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.

[DoSt97] A. Dolzmann, T. Sturm. Redlog: Computer Algebra Meets Computer Logic. ACM SIGSAM Bulletin, 31(2):2-9, June 1997. See `http://www.fmi.uni-passau.de/~redlog/`

[Dupr00] D. Dupre. Automated Theorem Proving by Integrating Proving, Solving and Computing. RISC, Johannes Kepler University of Linz. PhD Thesis. 2000.

[Gent34] G. Gentzen. Untersuchungen über das logische Schließen. Mathematische Zeitschrift 39, pp. 176-210, pp. 405-431, 1934-1935.

[GeKa06] Stefan Gerhold, Manuel Kauers. A Computer Proof of Turan's Inequality. Journal of Inequalities in Pure and Applied Mathematics 7(2), pp. 1-4. May 2006. Article 42.

[HKL96] H. Heugl, W. Klinger, J. Lechner: Mathematikunterricht mit Computeralgebra-Systemen. Addison-Wesley, Bonn, 1996.

[Jebe01] 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.

[Jirs95] M. Jirstand. Cylindrical Algebraic Decomposition - An Introduction. Technical Reports from the Automatic Control Group of Linköping, Linköping University, Sweden, October 18, 1995.

[Kalt00] E. Kaltofen. Challanges of Symbolic Computation: My Favorite Open Problems. Journal of Symbolic Computaion (JSC), 29(6), pp. 891-919, 2000.

[Kort04] U. Kortenkamp. Experimental Mathematics and Proofs in the Classroom. ZDM, 2004, Vol. 36(2), pp. 61-66.

[KrKr72] G. Kreisel, J. L. Krivine. Modelltheorie. Springer, Berlin, 1972.

[Kutz00] B. Kutzler. The Algebraic Calculator as a Pedagogical Tools for Teaching Mathematics. The International Journal of Computer Algebra in Mathematics Education, 7, pp. 5-24.

[Laza06] D. Lazard. Solving Kaltofen's Challenge on Zolotarev's Approximation Problem. In: J. Dumas (ed): ISSAC Proc. 2006 Internat. Symp. Symbolic Algebraic Comput., New-York, N. Y., 2006 ACM Press, pp. 196-203.

[LoWe93] R. Loos, V. Weispfenning. Applying Linear Quantifier Elimination. The Computer Journal, 36(5):450-462, 1993.

[Mark02] D. Marker. Model Theory: An Introduction, Springer, Berlin, 2002.

[MLHa05] S. McLaughlin, J. Harrison. A Proof-Producing Decision Procedure for Real Arithmetic, Springer LNCS 3632, pp. 295-314, 2005.

[MSW07] G. Mayrhofer, S. Saminger, W. Windsteiger. CreaComp: Experimental Formal Mathematics for the Classroom. In: Symbolic Computation and Education. Edited by Shangzhi Li, Dongming Wang, and Jing-Zhong Zhang. World Scientific Publishing Co., Singapore, New Jersey, 2007, pp. 94-114.

[Meli98] E. Melis. The Limit Domain. Artifical Intelligence Planning Systems (AIPS'98), 1998, pp. 199-207.

[Mish93] B. Mishra. Algorithmic Algebra. Texts and Monographs in Computer Science. Springer, 1993.

[MoVa08] B. Moore, R. Vajda. Some Experiments Using Quantifier Elimination to Prove the Non-existence of Dynamically Balanced Spherical Linkages, Technical Report 2008-10, SFB F013, Linz, Austria, September 2008.

[Pari03] P.A. Parrilo. Semidefinite Programming Relaxations for Semialgebraic Problems, Mathematical Programming Ser. B, Vol. 96, No.2, pp. 293-320, 2003.

[Pell99] F. J. Pelletier. A History of Natural Deduction and Elementary Logic Textbooks. In: J. Woods, B. Brown (eds) Logical Consequence: Rival Approaches, Vol. 1. (Oxford: Hermes Science Pubs) pp. 105-138.

[Piro04] F. Piroi. Tools for Using Automated Provers in Mathematical Theory Exploration PhD Thesis, RISC, Linz 2004.

[Poly65] G. Polya. Mathematical Discovery. On Understanding, Learning, and Teaching Problem Solving. Vol. I-II, New York, Wiley & Sons, 1964-65.

[Pres29] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt in Comptes Rendus du I congres de Mathématiciens des Pays Slaves. Warszawa. pp. 92-101, 1929.

[Roll03] H. Rolletschek. Decidable Logical Theories, Lecture Notes, RISC-Linz, pp.1-113, 2003.

[RoLo07] Eugenio Roanes-Lozano et al.: A General Purpose Mathematical GUI for Computer Algebra Systems, In: Mathematics and Computer in Simulation 2007 (in press)

[Sang06] C. Sangwin. Computer Algebra through a Proceptual Lens. Retirement as Process and Concept. Festschrift for Eddie Grey and David Tall, pp. 215-222.

[SchTe99] 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.

[Seid04] 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.

[Shaw06] W. T. Shaw. Complex Analysis with Mathematica. Cambridge University Press, 2006.

[Sofr05] V. Sofronie-Stokkermans. Hierarchic Reasoning in Local Theory Extensions. LNAI 3632, pp 219-234, Springer Verlag, 2005.

[SoNu04] R. Sommer, G. Nuckols. A Proof Environment for Teaching Mathematics. Journal of Automated Reasoning (JAR) 32(3), pp. 227-258, 2004.

[Strz99] A. Strzebonski. A Real Polynomial Decision Algorithm Using Arbitrary-Precision Floating Point Arithmetic, Reliable Comput., 5 (3), 1999, pp. 337-346; Developments in Reliable Computing (Tibor Csendes, ed.), New York: Kluwer Academic Publishers, 1999, pp. 337-346.

[Strz08] A. Strzebonski. Real Root Isolation for Exp-Log Functions. Proceedings of the ISSAC 2008 Conference, pp. 303-314, 2008, July, Hagenberg, Austria.

[Stur99] Real Quantifier Elimination in Geometry. PhD Thesis. FMI, Universität Passau, Passau, December 1999.

[Stur02] T. Sturm. Integration of Quantifier Elimination with Constraint Logic Programming, LNCS, 2385, Springer, 2002, pp. 75-107.

[SuTa89] P. Suppes, S. Takahashi. An Interactive Calculus Theorem-prover for Continuity Properties. Journal of Symbolic Computation (JSC), 1989, 7, pp. 573-590.

[SuSu98] G. Sutcliffe, C. B. Suttner. The TPTP Problem Library: CNF Release v1.2.1. Journal of Automated Reasoning (JAR) 21(2), pp. 177-203, 1998.

[Tars54] A. Tarski. Decision Method for Elementary Algebra and Geometry. In: Caviness-Johnson (eds): Quantifier Elimination and Cylindrical Decomposition pp.24-58.

[VaKo03] R. Vajda, Z. Kovács: Interactive Web Portals in Mathematics. Teaching Mathematics and Computer Science 1(2) (2003), pp. 347-361, Debrecen

[Vajd05] Finding Witness Terms in ETRCF by Qunatifier Elimination techniques. Talk given in the Omega-Theorema-Ultra Workshop in Saarbuecken, 2005. See `http://www.ags.uni-sb.de/ omega/workshops/`

[Vajd09a] R. Vajda, T. Jebelean, B. Buchberger. Combining Logical and Algebraic Techniques for Natural Style Proving in Elementary Analysis. Mathematics and Computers in Simulation 79 (2009), pp. 2310-2316, Elsevier; doi: 10.1016/j.matcom.2008.11.002

[Vajd09b] R. Vajda. Report on the CreaComp Project. Linking Computer Algebra and Automated Reasoning for Providing a New Learning Environment for Elementary Analysis. Teaching Mathematics and Computer Science 7(1) (2009), pp. 13-34, Debrecen

[WaBl87] Tie-Cheng Wang, W. W. Bledsoe. Hierarchical Deduction. In: Journal of Automated Reasoning (JAR) 3(1), 1987, pp.35-78.

[Weis97] V. Weispfenning. Quantifier Elimination for Real Algebra - The Quadratic Case and Beyond. Applicable Algebra in Engineering, Communication and Computing, 8(2):85-101, 1997.

[Weis99] V. Weispfenning. Mixed Real-Integer Linear Quantifier Elimination. ISSAC 1999, pp. 129-136.

[Weis06] V. Weispfenning. Elimination-Algorithmen mit Anwendungen, Passau, 2006 (Vorläufiger Entwurf)

[Weiss] E. W. Weisstein. Logistic Map. From MathWorld - A Wolfram Web Resource. http://mathworld.wolfram.com/LogisticMap.html

[Wind01] W. Windsteiger. A Set Theory Prover in Theorema: Implementation and Practical Applications. PhD Thesis, RISC, Linz, 2001.

[WBR06] W. Windsteiger, B. Buchberger, M. Rosenkranz. Theorema. In: The Seventeen Provers of the World, Freek Wiedijk (ed.), LNAI 3600, pp. 96-107. 2006. Springer Berlin Heidelberg New York.

[Wind07] W. Windsteiger. Towards Computer-Supported Proving in Maths Education. June 21, 2007. Contributed talk at CADGME.

[Wink96] F. Winkler. Polynomial Algorithms in Computer Algebra. Texts and Monographs in Symbolic Computation. Springer, 1996.

[Wolf99] S. Wolfram. The Mathematica Book. Wolfram Media Inc. Champaign, Illinois, USA and Cambridge University Press, 1999. See `http://www.wolfram.com/`

[ZiMe04] J. Zimmer, E. Melis. Constraint Solving for Proof Planning. Journal of Automated Reasoning (JAR) 33(1), pp. 51-88, 2004.