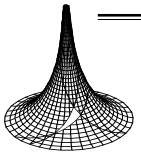


7/1 (2009), 13–34
tmcs@math.klte.hu
http://tmcs.math.klte.hu

Teaching
Mathematics and
Computer Science

An e-learning environment for elementary analysis: combining computer algebra, graphics and automated reasoning

Róbert Vajda

Abstract. CreaComp is a project at the University of Linz, which aims at producing computer-supported interactive learning units for several mathematical topics at introductory university level. The units are available as Mathematica notebooks. For student experimentation we provide computational, graphical and reasoning tools as well. This paper focuses on the elementary analysis units.

The computational and graphical tools of the CreaComp learning environment facilitate the exploration of new mathematical objects and their properties (e.g., boundedness, continuity, limits of real valued functions). Using the provided tools students should be able to collect empirical data systematically and come up with conjectures. A CreaComp component allows the formulation of precise conjectures and the investigatation of their validity. The Theorema system, which has been integrated into the CreaComp learning environment, 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. We demonstrate the learning situations and the provided tools through several examples.

Key words and phrases: computer algebra, elementary analysis, automated reasoning, extended quantifier elimination.

ZDM Subject Classification: U55, U75, E50.

1. Introduction

At the fifth CAME Symposium[1] several researchers pointed out, that although more and more examples are known where a computer algebra system

[1]http://www.lonklab.ac.uk/came/events/CAME5/

(CAS) could be exploited in a reasonable way for mathematics teaching and learning, typically their integration to the curriculum and the mathematical coursewares is slow. For instance, exercise books still contain the same old (CAS-free) exercises and methods, even without mentioning CAS, or some functionalities of CAS is 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 computer algebra 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. Even if this is accepted by the educator, 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.

One way to fill this gap is to turn to automated proving systems and adapt their services for the special needs of math education. At the University of Linz, a well known CAS, Mathematica [34] has been extended with reasoning capabilities: The Theorema proving system [8], [12] (www.theorema.org) 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. Thus, a reasonable integration and adaption of the services of a computer algebra and a reasoning system has led to the idea of the CreaComp learning environment.

The CreaComp project (2004–2006) [21], [33] 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. The units have a uniform structure and style. They contain standard static content such as

- the description of the educational goal,
- the definition of main notions,

- questions and explanations,
- summary.

Moreover, their interactive elements invite the students to explore the content of the unit by computer-supported tools.

Mathematica plays for the CreaComp learning 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.

This paper reports about the Elementary Analysis units. The author of the present paper not only developed the units, but took part also in the implementation of the graphical and reasoning tools. The paper is structured as follows: Section 2 describes the available tools in the CreaComp environment. Section 3 shows excerpts from the unit on Continuity of Real Values Functions. Section 4 reports about related and future work.

2. Tools in the CreaComp learning environment

2.1. Introduction

The most important novelty of the new learning environment is the presence of *reasoning* tools. Theorem proving systems are known for a while in computer science, but they can become attractive also for math education, only if

- they have a reasonable interface for inputing 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.

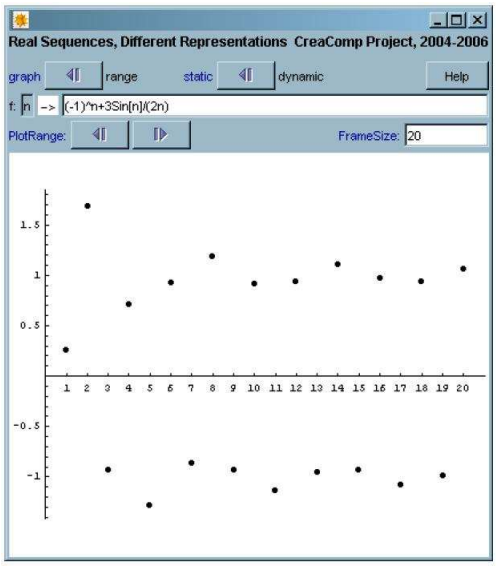
2.2. Graphical tools

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 provied by the author in his former work [28] and also by other researchers [24]. 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.[2]

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 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

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



Example 1. Widget for visualizing sequences

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 [19] 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.

2.3. Computational tools

Several researchers pointed out, see [25], 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 $\big[$"E3", any$[n]$, seq$[n] = \sum_{k=0}^{n} \frac{1}{k!}\big]$ [In]

UseAlso[Definition["E3"]] [In]

Compute$\big[\langle \text{seq}[n]|_{n=1,\ldots,5}\rangle,\ \text{built} - \text{in} \to \{\text{Built} - \text{in}[\text{"Quantifiers"}]\}\big]$ [In]

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

Compute$\big[\langle \text{seq}[n]|_{n=1,\ldots,5}\rangle,\ \text{built} - \text{in} \to \{\text{Built} - \text{in}[\text{"Quantifiers"}],$
$\text{Built} - \text{in}[\text{"Numbers"}]\}\big]$ [In]

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

2.4. 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 function definition and a definition for the notion of local continuity. Below, a simple proposition is formulated based on the definitions of the function and predicate symbols. Each formula has a label such that it can be referenced later in the educational unit.

EXAMPLE 3. Entering Definitions and Propositions in the CreaComp Environment

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

Definition["cont",any$[f, a]$, with$[a \in \mathbb{R}]$,

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

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

The top level command for reasoning 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. The following line shows how a call to a specific prover can be requested:

EXAMPLE 4. Prover Call

Prove[Proposition["sqiscont1"],
using→{Definition["cont"],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 (see Example 5, 6).

In Theorema, several general purpose provers are available for classical logic problems: PropositionalProver, PredicateProver, ResolutionProver, InductionProver etc. However, we share the opinion that purely logical techniques will not suffice for supporting practical reasoning in a specific domain like elementary analysis. We call the task of finding a correct deduction a *proving problem*. Here, a proving problem can be can be reduced by logical techniques to a solving problem over the reals or integers. Purely logical techniques cannot handle

efficiently the reduced problems, which are typically constraint satisfaction problems. Therefore, for handling proving problems, we implemented in the environment the standard logical inference rules and complemented them with algebraic techniques.

Combined methods like the PCS method introduced by Buchberger [10], [16] or the S-Decomposition strategy with extended quantifier elimination [17], [29] supports the investigation of proving problems in the CreaComp elementary analysis units. Extended quantifier elimination over reals enables to solve quantified parametric polynomial constraints, i.e., for finding witness terms for existentially quantified variables in the goal.

For a further description of the Theorema formal language and reasoners we refer to [8], [12], [32], [31], [21].

3. Case study

In the unit on Continuity of Real Function students learn about local continuity, continuity on an interval, and uniform continuity. The crucial observation is that if these notions are introduced formally following Cauchy then all of their defining formula contain a sequence of alternating quantifiers in the quantifier prefix. Reasoning with such formulae is difficult for students and for machines. For completing a simple proof the student should know [13], [18]

- how to apply definitions (rewriting),
- how to handle quantified variables in a proof problem, both on the premises and on the goal side (inferencing, quantifier-rules),
- the interaction of negation with the other logical connectives and with the quantifiers (inferencing, negation),
- how to solve algebraic inequalities (inequality solving).[3]

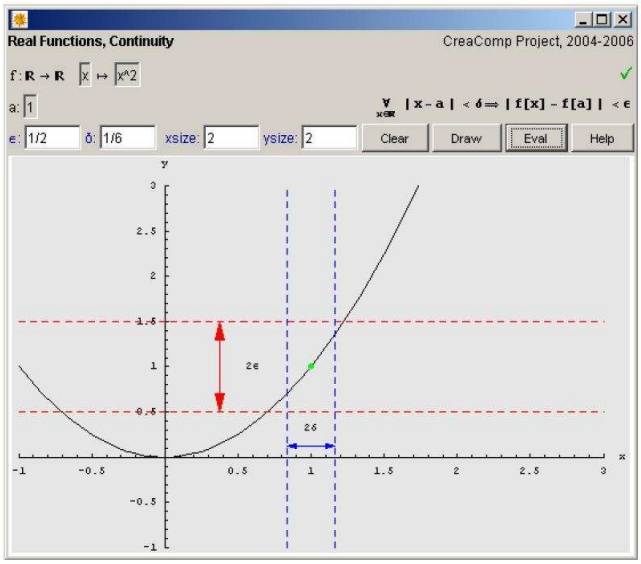
In the sequel we illustrate how the domain specific provers in the environment support understanding of the above steps. Three examples will be considered:

- The function 'Square' is continuous at 1.
- The function 'Square' is not uniformly continuous (on the whole real line).
- The Sum of two continuous functions is also continuous.

[3]solving here not necessarily means to find all possible solutions but only one sample solution

3.1. The function 'Square' is continuous at 1

Once more we emphasize that the formulation of the conjectures preceded by experimenting with other tools, e.g., for investigating the continuity of the square function, an $\epsilon - \delta$ widget can be exploited.



Example 5. Widget for local continuity

For the formalization of the first problem we refer back to Example 3 and for the prover call to Example 4.

Besides the definition for the function and the property investigated, no other formula is present in the initial knowledge base. Thus in the automated mode there is no guarantee that the student will get a proof for each (even valid) conjecture, but learning from a failing proof also gives us a lot of didactical possibilities [11]: The student may ask himself: Why did the prover stuck? Can I add something to the list of premises such that I can go further in the deduction? In case of a successful deduction, the overall structure and the application of specific inferences rules can be studied carefully by the students.

In this educational unit only one prover is used. Therefore, it can be set to default prover and then one does not even need to give it as an option for the prover call here. For selecting a specific reasoner from a bunch of available provers, one needs background knowledge about the inference rules and implementations.

Another level would be to allow the users to construct their own reasoner from available inference rules, but we do not consider those more sophisticated options here.

We now give the deduction generated by the system and explain the steps afterwards. We partitioned the deduction into five parts for a better overview (Part 1– Part 5). As part of the representation of the deduction attempt, the user sees the initial proof situation echoed (Part 1 in Example 6). Note that the subsequent pieces of the proof between the horizontal lines copied here exactly in the format produced automatically by the system.

EXAMPLE 6. First Deduction

Part 1

Prove:

Proposition(sqiscont1) $\text{Continuous}[\text{sq}, 1]$,

under the assumptions:

Definition (square) $\forall_x \text{sq}[x] = x^2$,

Definition (cont) $\forall_f \forall_{a\in\mathbb{R}} \text{Continuous}[f,a] \Leftrightarrow \forall_{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}} \exists_{\substack{\delta\in\mathbb{R}\\ \delta>0}} \forall_{\substack{x\in\mathbb{R}\\ |x-a|<\delta}} |f[x]-f[a]| < \epsilon$.

Part 2

Formula Proposition(sqiscont 1), using Definition(cont) is implied by:

$$\forall_{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}} \exists_{\substack{\delta\in\mathbb{R}\\ \delta>0}} \forall_{\substack{x\in\mathbb{R}\\ |x-1|<\delta}} |\,\text{sq}[x]-\text{sq}[1]| < \epsilon,$$

which using Definition (square) is implied by:

$$1\in\mathbb{R} \wedge \forall_{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}} \exists_{\substack{\delta\in\mathbb{R}\\ \delta>0}} \forall_{\substack{x\in\mathbb{R}\\ |x-1|<\delta}} |x^2-1^2| < \epsilon. \tag{1}$$

We prove the individual conjunctive parts of (1):

Part 3

Proof of (1.1) $1\in\mathbb{R}$:

Type condition in (1.1) was checked.

Proof of (1.2) $\forall_{\substack{\epsilon\in\mathbb{R}\\ \delta>0}} \exists_{\substack{\delta\in\mathbb{R}\\ \delta>0}} \forall_{\substack{x\in\mathbb{R}\\ |x-1|<\delta}} |x^2-1^2| < \epsilon$:

We assume

$$\epsilon_0\in\mathbb{R} \wedge \epsilon_0 > 0, \tag{2}$$

and prove

$$\exists_{\substack{\delta\in\mathbb{R}\\ \delta>0}} \forall_{\substack{x\in\mathbb{R}\\ |x-1|<\delta}} |x^2-1^2|<\epsilon_0. \tag{3}$$

We have to find δ^* such that

$$\delta^*\in\mathbb{R}\wedge\delta^*>0\wedge \forall_{\substack{x\in\mathbb{R}\\ |x-1|<\delta^*}} |x^2-1^2|<\epsilon_0. \tag{4}$$

Part 4

Based on the Cylindrical Decomposition method, we find suitable instantiations for δ^*. The method yields $\delta^* \leftarrow -1+(1+\epsilon_0)^{1/2}$.
Hence goal (4) would be implied by:

$$-1+(1+\epsilon_0)^{1/2}>0\wedge \forall_{\substack{x\in\mathbb{R}\\ |x-1|<-1+(1+\epsilon_0)^{1/2}}} |x^2-1^2|<\epsilon_0. \tag{5}$$

Part 5

We assume

$$x_0\in\mathbb{R}, \tag{6}$$

and prove

$$-1+(1+\epsilon_0)^{1/2}>0\wedge(|x_0-1|<-1+(1+\epsilon_0)^{1/2}\Rightarrow|x_0^2-1|<\epsilon_0). \tag{7}$$

[...]

All current goal formulae will be printed in red and the formulae in the knowledge base are printed in magenta.[4] The proof starts by applying the definitions. For the (universal) equivalence and equality in the definitions a rewrite rule will be applied (Part 1). In Part 2 rules for quantifiers are applied. Finally the proving problem is reduced to a solving problem. For handling formulae in the goal which have an existential quantifier at the leftmost position, a metavariable

[4]This and other features of the interactive notebook documents cannot be reproduced here by using black and white static text. Still, we distinguish the formulae in the knowledge base from the goal formulae by inverting the text and background colors.

for the existentially quantified variable is introduced (here δ^*). Each metavariable can be seen as a placeholder for a term which is not yet known. In the solving problem, a suitable δ^* has to be found, which satisfies a quantified constraint. The proof of branch (1.1) is simple: it is just typecheking based on built-in rules for reals. Part 4 is the most interesting part of the proof: A suitable δ^* is found using algebraic techniques. The proof is now basically done since we left with two (quantifier-free) inequality checkings (Part 5), but for didactical reasons, these inequalities can be investigated further using more elementary inference rules for inequalities.

For illustration, let us assume that we left with the (universal) polynomial problem $a + b = 1 \Rightarrow 2a^2 + 2b^2 \geq 1$. Algorithms in computational algebraic geometry [22], [23] can produce certificates like the polynomials of the right hand side of the identity $2a^2+2b^2-1 = (a+b-1)^2+(a-b)^2+2(a+b-1)$. Knowing those certificates, verifying nonnegativity becomes easy.

3.2. The function 'Square' is not uniformly continuous

Before we turn to the second proving problem, let us note that during the prover implementation, we followed and applied Buchberger's White Box-Black Box principle [4] for integrating algorithmic mathematics into math education. This means that by adjusting a prover option (Advanced/Didactical Mode), one can modify the granularity of the proof steps. We illustrate this in our second problem. First, for completeness, we give a compact deduction trace where we use the underlying algebraic technique as black box for handling quantified polynomial constraints over the reals (Advanced Mode, Example 7). Here the didactical situation is as follows: The primary topic is uniform continuity. Either students know already the algebraic algorithm for handling constraints over the reals (Cylindrical Decomposition) or they do not know the algorithm yet. In the second case, they just accept that there is a general technique for solving algebraic problems over the reals, but are not yet intertested in the details of it. In both cases, proving means in a strict sense here to get from the initial proving situtation to a quantified constraint satisfaction problem by purely predicate logic inference steps.

In contrast to the first version, we show afterwards the deduction for the same proving problem, except that we change one option of the prover, which is responsible for 'opening up' the Black Box of the algebraic technique (Example 8). This means that in this didactical scenario we would like to exploit the underlying algebraic technique (Cylindrical Decomposition) also for details of solving the

constraint problem. However, we do not come up with all possible certificates for the original algebraic algorithm [22], but rather we exploit the algorithm for generating witness terms for the existentially quantified variables in the quantified constraint problem.[5] We emphasize that this could be very important in the typical educational scenario where the freshman students meet at first time with elementary analysis problems and probably have no preliminary knowledge about advanced algebraic techniques. Here computer support is crucial, because the checking of inequalities over the reals can be an easier task than inequality solving (finding suitable solving terms). In addition, since the witness terms are typically not unique, the students can compare the witnesses found by themselves with the ones generated by the algebraic algorithm. In fact, the deduction in Example 6 was generated in the Didactical mode as well.

The proving problem here is that the 'Square' function is not uniformly continuous. The reader could also follow how the negation of a statement is handled by the reasoner: Negation is pushed inside the formulae and once more the quantified constraint problem is solved by algebraic techniques. Here are the two deductions for comparison. The first is in Advanced, the secondone is in Didactical mode:

EXAMPLE 7. Second Deduction, First Version

Prove:
Proposition(sqisnot ucont) $\neg$ UContinuous [sq],
under the assumptions:
Definition (square) $\underset{x}{\forall}\,\mathrm{sq}[x] = x^2$,

Definition (ucont) $\underset{f}{\forall}\,\mathrm{UContinuous}[f] \Leftrightarrow \underset{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}}{\forall}\ \underset{\substack{\delta\in\mathbb{R}\\ \delta>0}}{\exists}\ \underset{\substack{x,y\in\mathbb{R}\\ |x-y|<\delta}}{\forall}\ |f[x]-f[y]| < \epsilon$.

Formula Proposition (sqisnotucont) using Definition (ucont) is implied by:

$$(1)\qquad \neg\ \underset{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}}{\forall}\ \underset{\substack{\delta\in\mathbb{R}\\ \delta>0}}{\exists}\ \underset{\substack{x,y\in\mathbb{R}\\ |x-y|<\delta}}{\forall}\ |\,\mathrm{sq}[x]-\mathrm{sq}[y]| < \epsilon.$$

Pushing negation inside in (1), our new goal is:

$$(2)\qquad \underset{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}}{\exists}\ \underset{\substack{\delta\in\mathbb{R}\\ \delta>0}}{\forall}\ \underset{\substack{x,y\in\mathbb{R}\\ |x-y|<\delta}}{\exists}\ \neg(|\,\mathrm{sq}[x]-\mathrm{sq}[y]| < \epsilon).$$

[5]Thus the invention of the witness terms still remains black box but the handling of quantified polynomial constraint systems become whiter

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

$$\underset{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}}{\exists}\ \underset{\substack{\delta\in\mathbb{R}\\ \delta>0}}{\forall}\ \underset{\substack{x,y\in\mathbb{R}\\ |x-y|<\delta}}{\exists}\ \neg(|x^2-y^2|<\epsilon). \tag{3}$$

Formula (3) is proven by using algebraic techniques and we are done.

Example 8. Second Deduction, Second Version

Prove:
Proposition(sqisnot ucont) ¬ UContinuous[sq],
under the assumptions:
Definition (square) $\underset{x}{\forall}\,\mathrm{sq}[x]=x^2$,

Definition (ucont) $\underset{f}{\forall}\,\mathrm{UContinuous}\,[f] \Leftrightarrow \underset{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}}{\forall}\ \underset{\substack{\delta\in\mathbb{R}\\ \delta>0}}{\exists}\ \underset{\substack{x,y\in\mathbb{R}\\ |x-y|<\delta}}{\forall}\ |f[x]-f[y]|<\epsilon$.

Formula Proposition (sqisnotucont) using Definition (ucont) is implied by:

$$\neg\ \underset{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}}{\forall}\ \underset{\substack{\delta\in\mathbb{R}\\ \delta>0}}{\exists}\ \underset{\substack{x,y\in\mathbb{R}\\ |x-y|<\delta}}{\forall}\ |\,\mathrm{sq}[x]-\mathrm{sq}[y]|<\epsilon. \tag{1}$$

Pushing negation inside in (1), our new goal is:

$$\underset{\substack{\epsilon\in\mathbb{R}\\ \epsilon\in>0}}{\exists}\ \underset{\substack{\delta\in\mathbb{R}\\ \delta\in>0}}{\forall}\ \underset{\substack{x,y\in\mathbb{R}\\ |x-y|<\delta}}{\exists}\ \neg(|\,\mathrm{sq}[x]-\mathrm{sq}[y]|<\epsilon). \tag{2}$$

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

$$\underset{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}}{\exists}\ \underset{\substack{\delta\in\mathbb{R}\\ \delta>0}}{\forall}\ \underset{\substack{x,y\in\mathbb{R}\\ |x-y|<\delta}}{\exists}\ \neg(|x^2-y^2|<\epsilon). \tag{3}$$

We have to find an ϵ^* such that

$$\epsilon^*\in\mathbb{R}\wedge\epsilon^*>0\wedge\ \underset{\substack{\delta\in\mathbb{R}\\ \delta>0}}{\forall}\ \underset{\substack{x,y\in\mathbb{R}\\ |x-y|<\delta}}{\exists}\ \neg(|x^2-y^2|<\epsilon^*). \tag{4}$$

Based on the Cylindrical Decomposition method, we find suitable instantiations for ϵ^*. The method yields $\epsilon^*\leftarrow 1$. Hence goal (4) would be implied by:

$$1>0\wedge\ \underset{\substack{\delta\in\mathbb{R}\\ \delta>0}}{\forall}\ \underset{\substack{x,y\in\mathbb{R}\\ |x-y|<\delta}}{\exists}\ \neg(|x^2-y^2|<1). \tag{5}$$

We prove the individual conjuctive parts of (5):
Proof of (5.1) $1 > 0$:
Using algebraic methods (5.1) is proven.
Proof of (5.2) $\underset{\substack{\delta\in\mathbb{R}\\ \delta>0}}{\forall}\ \underset{\substack{x,y\in\mathbb{R}\\ |x-y|<\delta}}{\exists}\ \neg(|x^2-y^2|<1)$:
We assume

$$\delta_0 > 0 \wedge \delta_0 \in \mathbb{R}, \tag{6}$$

and show

$$\underset{\substack{x,y\in\mathbb{R}\\ |x-y|<\delta_0}}{\exists}\ \neg(|x^2-y^2|<1). \tag{7}$$

We have to find x^* and y^* such that

$$x^* \in \mathbb{R} \wedge y^* \in \mathbb{R} \wedge |x^*-y^*| < \delta_0 \wedge (|x^{*2}-y^{*2}|<1). \tag{8}$$

Using the Cylindrical Decomposition method, goal (8) can be solved for x^* and y^*. The method yields $x^* \leftarrow \frac{(1+\delta_0)^2}{-2\delta_0}$, $y^* \leftarrow -\sqrt{1+\frac{(1+\delta_0)^4}{4\delta_0^2}}$ for $(0<\delta_0\leq 1)$.[6]
Hence formula (8) is solved and we are done.

3.3. The sum of two continuous functions is continuous (CONT+)

This proof problem is attacked with the S-Decomposition strategy. The S-Decomposition strategy is suitable, if the main formulae in the knowledge base and in the goal have a similar structure. In addition to the definition applications and quantifier rules, a new step is needed here for reducing a problem to a pure real algebraic problem: Namely, we eliminate terms containing the function symbol f by introducing new variables (step (19) and (20) in Example 9 below).

Here we explain the method shortly. We cannot prove goal formula (19) directly, because the algebraic techniques cannot handle formulae containing terms with "unknown" function symbols like $f[a_0]$. The solution is based on the conjecture generation strategies described in Buchberger's Lazy Thinking paradigm [11]. Assume we are stuck with proving a formula G. Then we could recollect relevant formulae from the current knowledge base, say, $A_1, A_2, \cdots$. We slightly generalize the proof problem by introducing new variables for adequate terms and

[6]To save place, instances for $\delta_0 > 1$ are omitted

generating a conjecture in the form $A'_1 \wedge A'_2 \cdots \Rightarrow G'$. In case the generated conjecture can be proven, then the initial proving problem is solved as well. Here, in order to find suitable instantiations for the metavariables ϵ_1^*, ϵ_2^* by algebraic techniques, we reintroduce in Conjecture-1 the quantifiers in an appropriate order. The same happens when constructing Conjecture-2.

Since we already presented two complete deductions, here we skip some details and just focus on the most interesting parts.

EXAMPLE 9. Third Deduction

Prove:

Proposition(CONT+) Continuous$[f, a] \wedge$Continuous$[g, a] \Rightarrow$ Continuous$[f \oplus g, a]$, under the assumptions:

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

Definition(cont) $\underset{f}{\forall}\ \underset{a\in\mathbb{R}}{\forall}$ Continuous $[f, a] :\Leftrightarrow \underset{\substack{\epsilon\in\mathbb{R}\\ \epsilon>0}}{\forall}\ \underset{\substack{\delta\in\mathbb{R}\\ \delta>0}}{\exists}\ \underset{\substack{x\in\mathbb{R}\\ |x-a|<\delta}}{\forall} |f[x] - f[a]| < \epsilon$.

[The proof decomposes into three branches (main, types, conditions). For saving space, we do not show the next few lines of the automatically generated proof.]
Main branch: We assume:

(16.3) $$|f[y_1^*] - f[a]| < \epsilon_1^*,$$

(17.3) $$|g[y_2^*] - g[a]| < \epsilon_2^*,$$

and we prove:

(15.3) $$|f[y_0] + g[y_0] - (f[a] + g[a])| < \epsilon_0.$$

In order to prove (15.3), by using assumptions (16.3) and (17.3), it suffices to prove:

(19) $(|f[y_1^*] - f[a]| < \epsilon_1^* \wedge |g[y_2^*] - g[a]| < \epsilon_2^*) \Rightarrow |f[y_0] + g[y_0] - (f[a] + g[a])| < \epsilon_0.$

We eliminate the non-algebraic function symbols from (19), using the replacements
$y_1^* \leftarrow y_0$, $f[a] \leftarrow fa1$, $f[y_0] \leftarrow x_1$, $y_2^* \leftarrow y_0$, $g[a] \leftarrow ga1, g[y_0] \leftarrow x_2$:

(20) $$(|x1 - fa1| < \epsilon_1^* \wedge |x2 - ga1| < \epsilon_2^*) \Rightarrow |x1 + x2 - (fa1 + ga1)| < \epsilon_0.$$

We transform (20) by adding the appropriate quantifiers:

Conjecture-1 $\underset{fa1,ga1\in\mathbb{R}}{\forall}\ \underset{\substack{\epsilon 0\in\mathbb{R}\\ \epsilon 0>0}}{\forall}\ \underset{\epsilon 1>0\wedge\epsilon 2>0}{\exists}\ \underset{x1,x2\in\mathbb{R}}{\forall}$

$$(|x1-fa1|<\epsilon 1\wedge|x2-ga1|<\epsilon 2)\Rightarrow|x1+x2-(fa1+ga1)|<\epsilon 0.$$

By CAD, we prove Conjecture-1 and we find the witnesses: $\epsilon_1^*\rightarrow\frac{\epsilon_0}{2}$, $\epsilon_2^*\rightarrow\frac{\epsilon_0}{2}$.

2. Condition branch: We assume:

(15.2) $$|y_0-a|<\delta_0^*,$$

and we prove:

(13.3.1) $$|y_1^*-a|<\delta_1\wedge|y_2^*-a|<\delta_2.$$

In order to prove (13.3.1), by using assumption (15.2), it suffices to prove:

(21) $$|y_0-a|<\delta_0^*\Rightarrow(|y_1^*-a|<\delta_1\wedge|y_2^*-a|<\delta_2).$$

Using the already known substitutions, formula (21) becomes:

(22) $$|y_0-a|<\delta_0^*\Rightarrow(|y_0-a|<\delta_1\wedge|y_0-a|<\delta_2).$$

We transform (22) by adding appropriate quantifiers:

Conjecture-2 $\underset{a,\delta 1,\delta 2\in\mathbb{R}\ \delta 1>0\wedge\delta 2>0}{\forall}\ \underset{\substack{\delta 0\in\mathbb{R}\\ \delta 0>0}}{\exists}\ \underset{y0\in\mathbb{R}}{\forall}\ |y_0-a|<\delta 0\Rightarrow(|y_0-a|<\delta 1\wedge|y_0-a|<\delta 2).$

By CAD, we prove formula Conjecture-2 and we find witnesses: $\{\delta_0^*\leftarrow\overline{F_1}[\delta_1,\delta_2]\}$, where:

(A) $$\underset{\delta 1,\delta 2\in\mathbb{R}\ \delta 1>0\wedge\delta 2>0}{\forall}(\delta 2\leq\delta 1\Rightarrow\overline{F_1}[\delta 1,\delta 2]=\delta 2),$$

(B) $$\underset{\delta 1,\delta 2\in\mathbb{R}\ \delta 1>0\wedge\delta 2>0}{\forall}(\delta 2>\delta 1\Rightarrow\overline{F_1}[\delta 1,\delta 2]=\delta 1).$$

3. Type branch: [...]

4. Conclusions and related work

We presented an e-learning environment in which graphical, computational and reasoning tools support the exploration of notions in elementary analysis. We

emphasized the role of uniform language and software environment for all kind of mathematical activities. Experiments exploiting the graphical and computational tools facilitate gaining insights and formulating conjectures. After suitable formalization, conjectures are investigated by the reasoning tools. Proof presentation aims at showing deductions in a human readable format. This format is particularly important in the educational context. Typical deductions generated automatically by analysis provers were illustrated. The instructor's role remains crucial in the environment, because students need guidance for their experiments and to be able to analyze and understand the diverse feedbacks they got by using the tools.

The author hopes that the CreaComp environment contributes to students' learning. For students actively using the tools, links between the different representations of objects may become stronger. The same holds for the different phases of mathematical exploration like experimenting, conjecturing and reasoning. In the environment there is no need to hide simple, but important details of deductions: it minimizes the risk that student will copy or imitate the static canonical textbook proofs without interiorizing the universal scheme of proving and understanding the 'how and why' of reasoning. The disadvantage of the environment is that it needs extra time both on the student and instructor side. Although the units and tools are designed for education purposes, still, the specific syntax of the system should be understood and acquired beyond the natural language. A possible pitfall is that technicalities and the details of formal deduction system may shadow the primary topic of the educational unit (elementary analysis). There is a need for improving the control of granularity of the inference steps and the transition between formal and informal language which has been used for work in the current learning system.

Since the main novelty of the environment is the presence of reasoning tools, we are also interested which other learning systems offer similar reasoning tools.

We find a description of an interactive calculus prover for continuity properties in [27] and already in the early nineties Bledsoe collected some challenge problems for automated reasoning in elementary analysis [3]. It contained for example our third example, CONT+ as well. Meanwhile, many of the problems could be solved from the collection. Our methods and provers are now able to handle problems like the

- boundedness,
- limit,
- continuity,

- uniform continuity,
- convexity

of particular algebraic functions such as x^2, x^3, $1/x$ and also able to deal with problems such as CONT+, CONT*, LIM+, etc.

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.[7] Erica Melis investigated the Limit-Domain [20]. 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 [35]. Since in [20] a linear constraint solvers over the reals was used, the author does not know whether they could handle nonlinear (polynomial) inequality systems generally.

The Analytica prover was designed to prove theorems in elementary analysis and it was also implemented on top of Mathematica [2]. 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*.

Acknowledgements.

The author is indebted to Bruno Buchberger, Tudor Jebelean, Wolfgang Windsteiger and Zoltán Kovács for their useful suggestions and remarks. He also thanks the referee for providing constructive comments and suggestions.

References

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

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

[2] A. Bauer, E. M. Clarke and X. Zhao, Analytica – An experiment in combining theorem proving and symbolic computation, *Journal of Automated Reasoning (JAR)* **21**(3) (1998), 295–325.

[3] W. Bledsoe, Challenge problems in elementary analysis, *Journal of Automated Reasoning (JAR)* **6**(3) (1990), 341–359.

[4] B. Buchberger, Should students learn integration rules?, *ACM SIGSAM Bulletin* **24/1** (1990), 10–17.

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

[6] 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, 193–220.

[7] B. Buchberger, *Mathematiksoftware und Mathematikunterricht: Ein Vorwort*, In: Mathematikunterricht mit Computeralgebra-Systemen, (H. Heugl, W. Klinger, J. Lechner, eds.), Addison-Wesley, Bonn, 1996, 9–13.

[8] B. Buchberger, C. Dupre, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru and 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, 98–113.

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

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

[11] B. Buchberger, *Algorithm Invention and Verification by Lazy Thinking*, In: Proceedings of SYNASC'03. Analele Universitatii din Timisoara, SeriaMatematica - Informatica, XLI, Timisoara, 2003, 41–70.

[12] B. Buchberger, A. Craciun, T. Jebelean, L. Kovacs, T. Kutsia, K. Nakagawa, F. Piroi, N. Popov, J. Robu, M. Rosenkranz and W. Windsteiger, Theorema: Towards Computer-Aided Mathematical Theory Exploration, *Journal of Applied Logic* **4**(4) (2006), 470–504.

[13] B. Buchberger, *Mathematical Theory Exploration*, Invited Talk at IJCAR, Seattle, USA, 2006.

[14] E. Clarke and X. Zhao, Analytica: A theorem prover for mathematica, *The Mathematica Journal* **3**(1) (1993), 56–71.

[15] A. Dolzmann, T. Sturm and V.Weispfenning, *Real Quantifier Elimination in Practice*, MIP-9720, Universität Passau, December 1997, Algorithmic Algebra and Number Theory, (Matzat, B. H. and Greuel, G.-M. and Hiss, G., eds.), Springer, 1998, 221–247.

[16] D. Dupre, *Automated Theorem Proving by Integrating Proving*, Solving and Computing. PhD Thesis. RISC Technical Report no. 00–19, University of Linz, Austria, May 2000.

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

[18] W. Holsztinsky, Mathematical Analysis – Introduction for Experts, See: `http://www.alanisko.com/wlod/anFrames.html`.

[19] U. Kortenkamp, Experimental mathematics and proofs in the classroom, *ZDM* **36**(2) (2004), 61–66.

[20] E. Melis, The limit domain, *Artificial Intelligence Planning Systems (AIPS'98)* (1998), 199–207.

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

[22] S. McLaughlin and J. Harrison, *A Proof-Producing Decision Procedure for Real Arithmetic*, Springer LNCS 3632, 2005, 295–314.

[23] P. A. Parrilo, Semidefinite programming relaxations for semialgebraic problems, *Mathematical Programming* Ser. B, **96**, no. 2 (2003), 293–320.

[24] Eugenio Roanes-Lozano et al., A general purpose mathematical GUI for computer algebra systems, *Mathematics and Computer in Simulation* (2008), (in press).

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

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

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

[28] R. Vajda and Z. Kovács, Interactive web portals in mathematics, *Teaching Mathematics and Computer Science, Debrecen, Hungary* **1**(2) (2003), 347–361.

[29] R. Vajda, T. Jebelean and B. Buchberger, *Combining Logical and Algebraic Techniques for Natural Style Proving in Elementary Analysis*, Vol. 79, Mathematics and Computer in Simulation, 2009, 2310–2316, doi: 10.1016/j.matcom.2008.11.002.

[30] Tie-Cheng Wang and W. W. Bledsoe, Hierarchical Deduction, *Journal of Automated Reasoning (JAR)* **3**(1) (1987), 35–77.

[31] W. Windsteiger, B. Buchberger and M. Rosenkranz, *Theorema*, In: The Seventeen Provers of the World, (Freek Wiedijk, ed.), LNAI 3600, Springer, 2006, 96–107.

[32] W. Windsteiger, *A Set Theory Prover in Theorema: Implementation and Practical Applications*, PhD Thesis. RISC Technical Report no. 01-14, University of Linz, Austria, July 2001.

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

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

[35] J. Zimmer and E. Melis, Constraint Solving for Proof Planning, *Journal of Automated Reasoning (JAR)* **33**(1) (2004), 51–88.

RÓBERT VAJDA
BOLYAI INSTITUTE
UNIVERSITY OF SZEGED
SZEGED, HUNGARY

E-mail: `vajdar@math.u-szeged.hu`

(Received September, 2007)