# Techniques for Natural-style Proofs in Elementary Analysis*

WORK IN PROGRESS

Tudor Jebelean

RISC–Linz, JKU
www.risc.jku.at

**Abstract.** Combining methods from satisfiability checking with methods from symbolic computation promises to solve challenging problems in various areas of theory and application. We look at the basically equivalent problem of proving statements directly in a non–clausal setting, when additional information on the underlying domain is available in form of specific properties and algorithms. We demonstrate on a concrete example several heuristic techniques for the automation of natural–style proving of statements from elementary analysis. The purpose of this work in progress is to generate proofs similar to those produced by humans, by combining automated reasoning methods with techniques from computer algebra. Our techniques include: the S-decomposition method for formulae with alternating quantifiers, quantifier elimination by cylindrical algebraic decomposition, analysis of terms behaviour in zero, bounding the $\epsilon$-bounds, rewriting of expressions involving absolute value, algebraic manipulations, and identification of equal terms under unknown functions. These techniques are being implemented in the *Theorema* system and are able to construct automatically natural–style proofs for numerous examples including: convergence of sequences, limits and continuity of functions, uniform continuity, and other.

**Keywords:** Satisfiability Checking · Natural-style Proofs · Symbolic Computation.

## 1 Introduction

The need for natural[1]–style proofs (that is: similar to proofs produced by humans – see [2]) arises in various applications, as for instance in tutorials, demonstrations, and interactive teaching systems. Some authors argue for the use of natural style when the proof system is not completely automatic (e. g. interactive provers) because this facilitates the interaction with the human user.

* Partially supported by project “Satisfiability Checking and Symbolic Computation” (H2020-FETOPN-2015-CSA 712689).

[1] Here we do not mean *natural deduction* as described e. g. in [7].

When applied to problems over reals, *Satisfiability Modulo Theories (SMT)* solving combines techniques from Automated Reasoning and from Computer Algebra. From the point of view of Automated Reasoning, proving unsatisfiability of a set of clauses appears to be quite different from producing natural-style proofs. Indeed the proof systems are different (resolution on clauses[2] vs. some version of sequent calculus[3]), but they are essentially equivalent, relaying on equivalent transformations of formulae. Moreover, the most important steps in first–order proving, namely the instantiations of universally quantified formulae (which in natural–style proofs is also present as the equivalent operation of finding witnesses for existentially quantified goals), are actually the same or very similar. (For an illustration of instantiations in natural–style proofs see [4].) From the point of view of Computer Algebra, finding these instantiations is the most important operation, thus here again one can use the same techniques in SMT solving and natural–style proving. Therefore the techniques can be easily moved from one area to the other, because they are essentially equivalent.

In this paper we present our results on a class of proof problems which arise in elementary analysis. These are problems involving formulae with alternating quantifiers, which are difficult to solve by the purely logic approach, because this requires the use of a large number of formulae which express the necessary properties of numbers (naturals, integers, rationals, reals). We use the following techniques, which extend our previous work [8]:

- the S-decomposition method for formulae with alternating quantifiers [6],
- Quantifier Elimination by Cylindrical Algebraic Decomposition [3],
- analysis of terms behaviour in zero,
- bounding the $\epsilon$-bounds,
- rewriting of expressions involving absolute value,
- algebraic manipulations: solving, substitution, and simplification,
- identification of equal terms under unknown functions.

Our prover, implemented in the frame of the *Theorema* system [2], aims at producing natural–style proofs for simple theorems involving limits of sequences and of functions, continuity, uniform continuity, etc. An important aspect of the naturalness of the proof is the fact that the prover does not need to access a large collection of formulae (expressing the properties of the domains involved). Rather, the prover uses symbolic computation techniques from algebra in order to discover relevant terms and to check necessary conditions, and only needs as starting formulae the definitions of the main notions involved.

The prover can run either in interactive mode (the user has a choice of certain techniques at certain points) or fully automatic mode, because this is provided by the *Theorema* system. When in automatic mode, the proofs tested by us took at most 30 seconds.

[2] `en.wikipedia.org/wiki/Resolution_(logic)` (May 2018)
[3] `www.encyclopediaofmath.org/index.php/Sequent_calculus` (May 2018), `en.wikipedia.org/wiki/Sequent_calculus` (May 2018)

## 2 Example: Product of Convergent Sequences

We illustrate our method by the proof of the theorem *"The product of two convergent sequences is convergent."*, which is presented in detail on the next pages. The lines labeled [K1], ..., [K5] are not part of the proof, but only annotations for the purpose of this presentation: they indicate the key steps in the proof where special symbolic methods have to be used.

Note the specific structure of the quantified formulae: the quantifier has a first underline which declares the *type* of the variable, and possibly a second underline which declares a *condition* upon the quantified variable. These two conditions have a specific role during the decomposition of the proof using our method – see [1]. For space reasons, in this presentation we do not address the treatment of types. Note also that we follow the convention of *Mathematica* and *Theorema*, by denoting function and predicate application by square brackets instead of the traditional round parantheses.

The proof starts from the definition of the notion of convergent sequence and of the product of sequences, and decomposes the theorem into 3 statements: two assumptions (1), (2) and one goal (3).

The main structure of the proof follows from the S-decomposition method (see [6]): the quantifiers are removed from the 3 statements in parallel, using a combination of inference steps which decompose the proof into several branches, introduce Skolem constants[4], and require special terms (for instantiations or as witnesses). In the background the prover keeps certain quantified formulae which express the general structure of the proof and which are used at certain moments for finding the witnesses and the instantiation terms (as described in [1]).

At the beginning the 3 formulae are existential, in this situation S-decomposition is applied as follows:

- First for the assumptions (1), (2): introduce the Skolem constants $a_1, a_2$ instead of the quantified variables; assume that the type conditions under the quantifier hold;
- Second for the goal (3): *introduce the witness* $a_1 * a_2$ instead of the existentially quantified variable; prove additionally the type condition (not shown in the example) under the quantifier.

As an effect of this transformations the 3 formulae become universal (4), (5), (6), and S-decomposition proceeds as follows:

- First for the goal (6): introduce the Skolem constant $e_0$ instead of the quantified variable; assume that the conditions $e_0 \in \mathbb{R}$ and $e_0 > 0$ under the quantifier hold;
- Second for the assumptions (4), (5): *introduce the instantiation term* $e = \ldots$ instead of the universally quantified variable; prove additionally the conditions under the quantifier (the proof of the type condition is not shown in the example); and instantiate the assumptions.

[4] Skolem constants are new constant symbols introduced instead of quantified variables in certain situations: existential assumptions and universal goals.

After these transformations we obtain again 3 existentially quantified formulae and the cycle re-iterates. At every iteration of the proof cycle one needs a witness for the existential goal and an instantiation term for the universal assumptions: these are the difficult steps in the proof, for which we use special proof techiques based on symbolic computation.

***Definition:*** The sequence $f : \mathbb{N} \longrightarrow \mathbb{R}$ is **convergent** iff:

$$\underset{a\in\mathbb{R}}{\exists} \underset{\substack{e\in\mathbb{R} \\ e>0}}{\forall} \underset{M\in\mathbb{N}}{\exists} \underset{\substack{n\in\mathbb{N} \\ n\geq M}}{\forall} |f[n] - a| < e$$

***Theorem:*** **The product of convergent sequences is convergent.**

***Proof:***

We assume:

(1) $\underset{a\in\mathbb{R}}{\exists} \underset{\substack{e\in\mathbb{R} \\ e>0}}{\forall} \underset{M\in\mathbb{N}}{\exists} \underset{\substack{n\in\mathbb{N} \\ n\geq M}}{\forall} |f_1[n] - a| < e$

(2) $\underset{a\in\mathbb{R}}{\exists} \underset{\substack{e\in\mathbb{R} \\ e>0}}{\forall} \underset{M\in\mathbb{N}}{\exists} \underset{\substack{n\in\mathbb{N} \\ n\geq M}}{\forall} |f_2[n] - a| < e$

and we prove :

(3) $\underset{a\in\mathbb{R}}{\exists} \underset{\substack{e\in\mathbb{R} \\ e>0}}{\forall} \underset{M\in\mathbb{N}}{\exists} \underset{\substack{n\in\mathbb{N} \\ n\geq M}}{\forall} |\,(f_1[n] * f_2[n]) - a| < e$

By (1), (2) we can take $a_1$,$a_2 \in \mathbb{R}$ such that :

(4) $\underset{\substack{e\in\mathbb{R} \\ e>0}}{\forall} \underset{M\in\mathbb{N}}{\exists} \underset{\substack{n\in\mathbb{N} \\ n\geq M}}{\forall} |f_1[n] - a_1| < e$

(5) $\underset{\substack{e\in\mathbb{R} \\ e>0}}{\forall} \underset{M\in\mathbb{N}}{\exists} \underset{\substack{n\in\mathbb{N} \\ n\geq M}}{\forall} |f_2[n] - a_2| < e$

[K1] *Witness for the existential goal:* $a \rightarrow a_1 * a_2$

For proving (3) it is sufficient to prove :

(6) $\underset{\substack{e\in\mathbb{R} \\ e>0}}{\forall} \underset{M\in\mathbb{N}}{\exists} \underset{\substack{n\in\mathbb{N} \\ n\geq M}}{\forall} |\,(f_1[n] * f_2[n]) - (a_1 * a_2)\,| < e$

For proving (6) we take $e_0 \in \mathbb{R}$ arbitrary but fixed, we assume :

(7) $e_0 > 0$

and we prove :

(8) $\underset{M\in\mathbb{N}}{\exists} \underset{\substack{n\in\mathbb{N} \\ n\geq M}}{\forall} |\,(f_1[n] * f_2[n]) - (a_1 * a_2)\,| < e_0$

$[K2]$ *Instantiation term for universal assumptions:* $e \to \text{Min}\,[\ldots]$

We consider :

$e = \text{Min}\left[1, \frac{e_0}{|a_2|+|a_1|+1}\right]$

First we prove :

(9) $e > 0$

This follows from (7) and elementary properties of $\mathbb{R}$.
Using (9), from (4) and (5) we obtain :

(10) $\underset{M\in\mathbb{N}}{\exists}\ \underset{\substack{n\in\mathbb{N}\\ n\geq M}}{\forall} |f_1[n] - a_1| < e$

(11) $\underset{M\in\mathbb{N}}{\exists}\ \underset{\substack{n\in\mathbb{N}\\ n\geq M}}{\forall} |f_2[n] - a_2| < e$

By (10) and (11) we can take $M_1$,$M_2$$\in \mathbb{N}$ such that :

(12) $\underset{\substack{n\in\mathbb{N}\\ n\geq M_1}}{\forall} |f_1[n] - a_1| < e$

(13) $\underset{\substack{n\in\mathbb{N}\\ n\geq M_2}}{\forall} |f_2[n] - a_2| < e$

$[K3]$ *Witness for existential goal* : $M \to \text{Max}\,[M_1, M_2]$

In order to prove (8) it suffices to prove:

(14) $\underset{\substack{n\in\mathbb{N}\\ n\geq \text{Max}[M_1,M_2]}}{\forall} |\,(f_1[n] * f_2[n]) - (a_1 * a_2)\,| < e_0$

For proving (14) we take $n_0 \in \mathbb{N}$ arbitrary but fixed, we assume :

(15) $n_0 \geq \text{Max}\,[M_1, M_2]$

and we prove :

(16) $|\,(f_1\,[n_0] * f_2\,[n_0]) - (a_1 * a_2)\,| < e_0$

$[K4]$ *Instantiation term for universal assumptions* : $n \to n_0$

First we prove :

(17) $(n_0 \geq M_1) \wedge (n_0 \geq M_2)$

This follows from (15) and elementary properties of $\mathbb{R}$.
Using (17), from (12) and (13) we obtain:

(18) $|f_1\,[n_0] - a_1| < e$

(19) $|f_2\,[n_0] - a_2| < e$

$[K5]$ *Algebraic manipulations*

Using elementary properties of $\mathbb{R}$ we transform (16) into:

(20) $|a_1 * (f_2[n_0] - a_2) + a_2 * (f_1[n_0] - a_1) + (f_1[n_0] - a_1) * (f_2[n_0] - a_2)| < e_0$

Using elementary properties of $\mathbb{R}$, from (18) and (19) we obtain:

(21) $|a_1 * (f_2[n_0] - a_2) + a_2 * (f_1[n_0] - a_1) + (f_1[n_0] - a_1) * (f_2[n_0] - a_2)| \leq$

$\leq |a_1 * (f_2[n_0] - a_2)| + |a_2 * (f_1[n_0] - a_1)| + |(f_1[n_0] - a_1) * (f_2[n_0] - a_2)| =$

$= |a_1| * |f_2[n_0] - a_2| + |a_2| * |f_1[n_0] - a_1| + |f_1[n_0] - a_1| * |f_2[n_0] - a_2| <$

$< |a_1| * e + |a_2| * e + e * e \leq |a_1| * e + |a_2| * e + e = e * (|a_1| + |a_2| + 1) =$

$= \frac{e_0}{|a_2|+|a_1|+1} * (|a_1| + |a_2| + 1) = e_0$

which proves the goal.

# 3 Application of Special Techniques

We describe here how the special techniques mentioned in the introduction are used in the course of the proof presented above, namely at the key steps indicated with [K1], ..., [K5].

## 3.1 K1: Witness for Existential Goal

Here the prover must produce the witness $a_1 * a_2$ needed for the existential variable $a$ in the current goal (3). We use the well known technique of *metavariables* (see also [4]), that is we replace the existential variable by a new symbol, which is a name for the term which we need to find. This term will be found later, when the prover generates a certain simplified formula (see [8, 1]) which we call formula (A):

$$\forall_{a_1,a_2}\exists_{a_0}\forall_{e_0}(e_0 > 0 \Rightarrow \exists_{e_1,e_2}(e_1 > 0 \wedge e_2 > 0 \wedge$$
$$\forall_{x_1,x_2}(|x_1 - a_1| < e_1 \wedge |x_2 - a_2| < e_2) \Rightarrow |x_1 * x_2 - a_0| < e_0))$$

The value of the metavariable (standing for $a_0$) can be found using Quantifier Elimination (QE) by Cylindrical Algebraic Decomposition (CAD), as described in [1] – which works for the case of the *sum* of convergent sequences. However in the case of *product* the corresponding QE problem cannot be solved in a reasonable time by CAD (e. g. in *Mathematica*), and even if solved, it generates a very complicated expression which cannot be used for finding $a_0$.

In the proof above we used another, much simpler technique: *reasoning about terms behaviour in zero*. It is clear that the formula (A) expresses the behaviour of the polynomials in any (small) vicinity of zero. Since polynomials are continuous, this will also be their behaviour *in zero*. One can in fact prove that formula (A) is equivalent to the formula:

$$\forall_{a_1,a_2}\exists_{a_0}\forall_{x_1,x_2}(|x_1 - a_1| = 0 \wedge |x_2 - a_2| = 0) \Rightarrow |x_1 * x_2 - a_0| = 0$$

In our special case it is immediately clear that $a_0$ equals $a_1 * a_2$, but we implemented a more general method: the two LHS equations are solved for $x_1, x_2$, then the values are replaced in the RHS equation, which is then solved for $a_0$.

### 3.2 K2: Instantiantion Term for Universal Assumptions

Here the prover must produce an appropriate term for the instantiation of the assumptions (4) and (5). In the case of *sum* of sequences this is $e_0/2$ and is relatively easy to guess by a human prover. Similar to [K1], a metavariable is used instead of the unknown term, and this will correspond to the existential variables $e_1, e_2$ in formula (A).

Again it is possible to use QE by CAD, by treating the formula (A) after replacing $a_0$ with its value and removal of the quantifiers for $a_0, a_1, a_2$ – see [1]. This works in the case of sum, but again it does not work satisfactorily in the case of product.

In order to find this witness (we assume that it is the same for $e_1$ and $e_2$), we use algebraic manipulation (solving, substitution, and computation), as well as rewriting of terms under the absolute value function. This is probably the most interesting of the new techniques presented here, and is detailed below at [K5].

### 3.3 K3: Witness for the Existential Goal

The proof needs a witness for the existential variable $M$ in the goal (8). Similarly to the other key steps, the prover uses a metavariable and produces an appropriate quantified formula whose treatment by QE allows to infer the right term, as described in [1].

### 3.4 K4: Instantiation of Universal Assumptions

The assumptions (12) and (13) need to be instantiated with appropriate terms for the universally quantified variable $n$. Here we use the special heuristics: *identification of equal terms under unknown functions*.

Since $f_1$ and $f_2$ are universally quantified in the original formula, and later become arbitrary constants, we do not know anything about their behaviour. In the goal (16), $f_1$ and $f_2$ have argument $n_0$. Therefore it will be possible to use the assumptions (12) and (13) for proving (16) only if $f_1$ and $f_2$ are applied to the same argument. (This corresponds in fact to resolution in first order logic.) In the case of this proof the solution is to set $n$ to $n_0$, but even if the expressions are more complicated one can use equation solving, substitution, and computation in order to find more complicated terms. Moreover, after this instantiation we substitute $f_1[n_0]$ and $f_2[n_0]$ with new arbitrary constants (e. g. $x_1$, $x_2$, respectively): this makes our expressions polynomial and helps creating the formula (A).

### 3.5 K5: Algebraic Manipulations

The most challenging part is the automatic generation of the instantiation term needed at step [K2], which is performed by a heuristic combination of solving, substitution, and simplifying, as well as rewriting of expressions under the absolute value function.

Note the goal (16) has under the absolute value function the expression (call it $E_0$) corresponding to $x_1 * x_2 - a_1 * a_2$. Let us also name the absolute value arguments of the assumptions (18) and (19) as $E_1$ and $E_2$, respectively.

First we use the following heuristic principle: transform the goal expression $E_0$ such that it uses as much as possible $E_1$ and $E_2$, because about those we know that they are small. In order to do this we take new variables $y_1, y_2$, we solve the equations $y_1 = E_1$ and $y_2 = E_2$ for $x_1, x_2$, we substitute the solutions in $E_0$ and the result simplifies to: $a_1 * y_2 + a_2 * y_1 + y_1 * y_2$. This is the internal representation of the absolute value argument in the goal (20). Note that the transformation from (16) to (20) is relatively challenging even for a human prover.

The formula (21) is realized by *rewriting of the absolute value expressions*. Namely, we apply certain rewrite rules to expressions of the form $|E|$ and their combination, as well as to the metavariable $e$. Every rewrite rule transforms a (sub)term into one which is not smaller, so we are sure to obtain a greater or equal term. The final purpose of these transformations is to obtain a strictly positive ground term $t$ multiplied by the target metavariable (here $e$). Since we need a value for $e$ which fullfils $t * e \leq e_0$, we can set $e$ to $e_0/t$. The rewrite rules come from the elementary properties of the absolute value function: (e. g. $|u + v| \leq |u| + |v|$)) and from the principle of *bounding the $\epsilon$-bounds*: Since we are interested in the behaviour of the expressions in the immediated vicinity of zero, the bounds $(e, e_0, e_1, e_2)$ can be bound from above by any positive value. In the case of product (presented here), we also use the rule: $e * e \leq e$, that is we bound $e$ to 1. This is why the final expression of $e$ is the minimum between 1 and the expression found as above.

This method works of course for the case of sum of sequences.

In order to make it work for more complex expressions, namely rational functions, we use a second set of rules which decrease the term: in order to obtain a bound for $U/V$, increase $U$ and decrease $V$. Using this we obtain automatically appropriate bounds for the case of inverse of a sequence and for the case of fraction of two sequences.

Full detail of the techniques and of the examples are presented in [5].

### 3.6 Proving Simple Conditions

At certain places in the proof, the conditions upon certain quantified variables have to be proven. The prover does not display a proof of these simple statements, but just declares them to be consequences of "elementary properties of $\mathbb{R}$". (Such elementary properties are also invoked when developing formulae (20) and (21).) By "elementary properties" in this context we understand the properties of various constants (like 0, 1), functions (like Min, Max, absolute value, $+, -, *, /$), and predicates (like $=, <, \leq$) over reals and naturals, which are normally studied before the notions of limit, continuity, etc. and which are typically considered prerequisites for working in elementary analysis.

In the background, however, the prover uses *Mathematica* functions in order to check that these statements are correct. This happens for the subgoal (9) and will be treated after the instatiation term is found, by using QE on the formula $\forall_{e_0}(e_0 > 0 \Rightarrow e > 0)$ (where $e$ has the found value Min[...]), which returns *true* in *Mathematica*. The same procedure is used for the subgoal (17).

## 4 Conclusion and Further Work

The full automation of proofs in elementary analysis constitutes a very intersting application for the combination of logic and algebraic techniques, which is essentially equivalent to SMT solving (combining satisfiability checking and symbolic computation). Our experiments show that complete and efficient automation is possible by using certain heuristics in combination with complex algebraic algorithms.

Further work includes a systematic treatment of various formulae which appear in textbooks, and extension of the heuristics to more general types of formulae. In this way we hope to address the class of problems which are usually subject to SMT solving.

## References

1. Abraham, E., Jebelean, T.: Adapting Cylindrical Algebraic Decomposition for Proof Specific Tasks. In: Kusper, G. (ed.) ICAI 2017: 10th International Conference on Applied Informatics (2017), in print

2. Buchberger, B., Jebelean, T., Kutsia, T., Maletzky, A., Windsteiger, W.: Theorema 2.0: Computer-Assisted Natural-Style Mathematics. JFR **9**(1), 149–185 (2016)
3. Collins, G.E.: Quantier elimination for real closed fields by cylindrical algebraic decomposition. In: Automata Theory and Formal Languages. LNCS, vol. 33, pp. 134–183. Springer (1975)
4. Dramnesc, I., Jebelean, T.: Synthesis of list algorithms by mechanical proving. Journal of Symbolic Computation **69**, 61–92 (2015)
5. Jebelean, T.: Experiments with Automatic Proofs in Elementary Analysis. Tech. Rep. 18-06, Research Institute for Symbolic Computation (RISC), Johannes Kepler University Linz (April 2018)
6. Jebelean, T., Buchberger, B., Kutsia, T., Popov, N., Schreiner, W., Windsteiger, W.: Automated Reasoning. In: Buchberger, B., et al. (eds.) Hagenberg Research, pp. 63–101. Springer (2009)
7. Pelletier, F.J.: A history of natural deduction and elementary logic textbooks. Logical consequence: Rival approaches **1**, 105–138 (2000)
8. Vajda, R., Jebelean, T., Buchberger, B.: Combining Logical and Algebraic Techniques for Natural Style Proving in Elementary Analysis. Mathematics and Computers in Simulation **79**(8), 2310–2316 (April 2009)