System Description: The Cut-Elimination System CERES [*]

Matthias Baaz[1], Stefan Hetzl[2], Alexander Leitsch[2], Clemens Richter[2], Hendrik Spohr[2]

[1]Institute of Discrete Mathematics and Geometry (E104), Vienna University of Technology, Wiedner Hauptstraße 8-10, 1040 Vienna, Austria
`baaz@logic.at`

[2]Institute of Computer Languages (E185), Vienna University of Technology, Favoritenstraße 9, 1040 Vienna, Austria
{`hetzl|leitsch|richter|spohr`}`@logic.at`

Abstract

Cut-elimination is the most prominent form of proof transformation in logic. The elimination of cuts in formal proofs corresponds to the removal of intermediate statements (lemmas) in mathematical proofs. The cut-elimination method CERES (cut-elimination by resolution) works by constructing a set of clauses from a proof with cuts. Any resolution refutation of this set then serves as a skeleton of an **LK**-proof with only atomic cuts.

The use of resolution and the enormous size of (formalized) mathematically relevant proofs suggest an implementation able to handle rather complex cut-elimination problems. In this paper we present an implementation of CERES: the system `CERES`. It already implements an improvement based on an extension of **LK** to the calculus **LKDe** containing equality rules and rules for introduction of definitions which makes it easier to formalize and interpret mathematical proofs in **LK**. Furthermore it increases the efficiency of the cut-elimination method. The system `CERES` already performs well in handling quite large proofs.

1 Introduction

Proof analysis is a central mathematical activity which proves crucial to the development of mathematics. Indeed many mathematical concepts such as the notion of group or the notion of probability were introduced by analyzing existing arguments. In some sense the analysis and synthesis of proofs form the very core of mathematical progress [12, 13].

Cut-elimination introduced by Gentzen [8] is the most prominent form of proof transformation in logic and plays an important role in automating the analysis of mathematical proofs. The removal of cuts corresponds to the elimination of intermediate

[*]supported by the Austrian Science Fund (project no. P17995-N12)

statements (lemmas) from proofs resulting in a proof which is analytic in the sense, that all statements in the proof are subformulas of the result. Therefore, the proof of a combinatorial statement is converted into a purely combinatorial proof. Cut elimination is therefore an essential tool for the analysis of mathematical proofs, especially to make implicit parameters explicit. Cut free derivations allow for

- the extraction of Herbrand disjunctions, which can be used to establish bounds on existential quantifiers (e.g. Luckhardt's analysis of the Theorem of Roth [11]).
- the construction of interpolants, which allow for the replacement of implicit definitions by explicit definitions according to Beth's Theorem.
- the calculation of generalized variants of the end formula.

In a formal sense Girard's analysis of van der Waerden's theorem [9] is the application of cut-elimination to the proof of Fürstenberg/Weiss with the "perspective" of obtaining van der Waerden's proof. Indeed an application of a complex proof transformation like cut-elimination by humans requires a goal oriented strategy.

Note that cut-elimination is *non-unique*, i.e. there is no single cut-free proof which represents *the* analytic version of a proof with lemmas. Therefore the application of purely computational methods on existing proofs may even produce new interesting proofs. Indeed, it is non-uniqueness which makes computational experiments with cut-elimination interesting [3]. The experiments can be considered as a source for a base of proofs in formal format which provide different mathematical and computational information.

CERES [5, 6] is a cut-elimination method that is based on resolution. The method roughly works as follows: The structure of the proof containing cuts is mapped to an unsatisfiable set of clauses $\mathcal{C}$ (the *characteristic clause set*). A resolution refutation of $\mathcal{C}$, which is obtained using a first-order theorem prover, serves as a skeleton for the new proof which contains only atomic cuts. In a final step also these atomic cuts can be eliminated, provided the (atomic) axioms are valid sequents; but this step is of minor mathematical interest and of low complexity. In the system `CERES`[1] this method of cut-elimination has been implemented. The system is capable of dealing with formal proofs in an extended version of **LK**, among them also very large ones (i.e. proofs with more than 1500 nodes and cut-elimination has been done within 14 seconds).

The extension of CERES to **LKDe** a calculus containing definition introductions and equality rules [10] moves the system closer to real mathematical proofs. In particular, introduction of definitions and concepts is perhaps the most significant activity of a mathematician in structuring proofs and theories. By its high efficiency (the core of the method is first-order theorem proving by resolution and paramodulation) `CERES` might become a strong tool in *automated proof mining* and contribute to an experimental culture of *computer-aided proof analysis* in mathematics.

[1]available at `http://www.logic.at/ceres/`

2 The System CERES

The cut-elimination system CERES is written in ANSI-C++. There are two main tasks. One is to compute an unsatisfiable set of clauses characterizing the cut formulas. This is done by automatically extracting the so-called *characteristic clause set* CL(φ). The other is to generate a resolution refutation of CL(φ) by an external theorem prover[2], and to compute the necessary projections of φ w.r.t. the clauses in CL(φ) actually used for the refutation. A projection of φ modulo a clause $C \in \mathrm{CL}(\varphi)$ is gained by applying all inference rules of φ not operating on cut ancestors to the initial sequents of C. The properly instantiated projections are concatenated, using the refutation obtained by the theorem prover as a skeleton of a proof with only atomic cuts.

The extraction of the *characteristic clause set* CL(φ) from an **LKDe**-proof φ is defined inductively. For every node ν of φ either:

- If ν is an occurrence of an axiom S and S' is the subsequent of S containing only ancestors of cut formulas then $\mathcal{C}_\nu = \{ S' \}$.
- Let ν_1, ν_2 be the predecessors of ν in a binary inference then we distinguish:
  - The auxiliary formulas of ν_1, ν_2 are ancestors of cut formulas then $\mathcal{C}_\nu = \mathcal{C}_{\nu_1} \cup \mathcal{C}_{\nu_2}$.
  - Otherwise $\mathcal{C}_\nu = \mathcal{C}_{\nu_1} \times \mathcal{C}_{\nu_2}$, where $\mathcal{C} \times \mathcal{D} = \{ C \circ D \mid C \in \mathcal{C}, D \in \mathcal{D} \}$ and $C \circ D$ is the merge of the clauses C and D.
- Let ν' be the predecessor of ν in a unary inference then $\mathcal{C}_\nu = \mathcal{C}_{\nu'}$.

The characteristic clause set CL(φ) is defined as $\mathcal{C}_\nu$ where ν is the root node of φ.

The *definition rules* of **LKDe** directly correspond to the *extension principle* (see [7]) in predicate logic. It simply consists in introducing new predicate- and function symbols as abbreviations for formulas and terms. Mathematically this corresponds to the introduction of new concepts in theories. Let A be a first-order formula with the free variables $x_1, \ldots, x_k$ (denoted by $A(x_1, \ldots, x_k)$ and P be a *new* k-ary predicate symbol (corresponding to the formula A). Then the rules are:

$$\frac{A(t_1, \ldots, t_k), \Gamma \vdash \Delta}{P(t_1, \ldots, t_k), \Gamma \vdash \Delta}\ \mathrm{def}_P\colon l \qquad \frac{\Gamma \vdash \Delta, A(t_1, \ldots, t_k)}{\Gamma \vdash \Delta, P(t_1, \ldots, t_k)}\ \mathrm{def}_P\colon r$$

for arbitrary sequences of terms $t_1, \ldots, t_k$. There are also definition introduction rules for new function symbols which are of similar type. The *equality rules* are:

$$\frac{\Gamma_1 \vdash \Delta_1, s = t \quad A[s]_\Lambda, \Gamma_2 \vdash \Delta_2}{A[t]_\Lambda, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ =\colon l1 \qquad \frac{\Gamma_1 \vdash \Delta_1, t = s \quad A[s]_\Lambda, \Gamma_2 \vdash \Delta_2}{A[t]_\Lambda, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ =\colon l2$$

for inference on the left and

$$\frac{\Gamma_1 \vdash \Delta_1, s = t \quad \Gamma_2 \vdash \Delta_2, A[s]_\Lambda}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A[t]_\Lambda}\ =\colon r1 \qquad \frac{\Gamma_1 \vdash \Delta_1, t = s \quad \Gamma_2 \vdash \Delta_2, A[s]_\Lambda}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A[t]_\Lambda}\ =\colon r2$$

[2]The current version of CERES uses the automated theorem prover Otter (see `http://www-unix.mcs.anl.gov/AR/otter/`), but any refutational theorem prover based on resolution and paramodulation may be used.

on the right, where Λ denotes a set of positions of subterms where replacement of s by t has to be performed. We call $s = t$ the *active equation* of the rules. Note that, on atomic sequents, the rules coincide with *paramodulation* – under previous application of the most general unifier.

Concerning the extension of the CERES-method from **LK** to **LKDe**, equality rules appearing within the input proof are propagated to the projections like any other binary rules. During theorem proving equality is treated by paramodulation; its application within the final clausal refutation is then transformed to the appropriate equality rules in **LKDe**. The definition introductions do not require any other special treatment within `CERES` than all other unary rules; in particular, they have no influence on the theorem proving part.

Since the restriction to skolemized proofs is crucial to the CERES-method, the system also performs skolemization (according to Andrew's method [2]) on the input proof.

The system `CERES` expects an **LKDe** proof of a sequent S and a set of axioms as input, and, after validation of the input proof, computes a proof of S containing at most atomic-cuts. Input and output are formatted using the well known data representation language XML. This allows the use of arbitrary and well known utilities for editing, transformation and presentation and standardized programming libraries. To increase the performance and avoid redundancy, most parts of the proofs are internally represented as directed acyclic graphs. This representation turns out to be very handy, also for the internal unification algorithms.

The formal analysis of mathematical proofs (especially by a mathematician as a pre- and post-"processor") relies on a suitable format for the input and output of proofs, and on an appropriate aid in dealing with them. We developed an intermediary proof language connecting the language of mathematical proofs with **LKDe** and the compiler `HLK`[3] transforming proofs written in this higher-level language to **LKDe**. Furthermore we implemented a proof tool[4] with a graphical user interface, allowing a convenient input and analysis of the output of `CERES`. Thereby the integration of definition- and equality-rules into the underlying calculus plays an essential role in overlooking, understanding and analyzing complex mathematical proofs by humans.

3 Example

The example proof below is taken from [14]; it was formalized in **LK** and analyzed by a former version of CERES in the paper [3]. Here we use the extensions by equality rules and definition-introduction in **LKDe** to give a simpler formalization and analysis of the proof. The end-sequent formalizes the statement: on a tape with infinitely many cells which are all labelled by 0 or by 1 there are at least two cells labelled by the same number. $f(x) = 0$ expresses that the cell number x is labelled by 0. Indexing of cells is done by number terms defined over $0, 1$ and $+$. The proof φ below uses two lemmas: (1) there are infinitely many cells labelled by 0 and (2) there are infinitely many cells labelled by 1. The proof shows the statement by a case distinction: If there are infinitely many cells labelled by 0 then choose two of them, otherwise there are infinitely many

[3]available at `http://www.logic.at/hlk/`

[4]available at `http://www.logic.at/prooftool/`

cells labelled by 1 and we can then choose two of them. In each case there are two cells with the same value. These lemmas are eliminated by CERES and a more direct argument is obtained in the resulting proof φ'. Ancestors of the cuts in φ are indicated in boldface.

Let φ be the proof

$$\cfrac{\cfrac{\overset{(\tau)}{A \vdash \mathbf{I_0}, \mathbf{I_1}} \quad \overset{(\epsilon_0)}{\mathbf{I_0} \vdash \exists p \exists q (p \neq q \wedge f(p) = f(q))}}{A \vdash \exists p \exists q (p \neq q \wedge f(p) = f(q)), \mathbf{I_1}}\ cut \qquad \overset{(\epsilon_1)}{\mathbf{I_1} \vdash \exists p \exists q (p \neq q \wedge f(p) = f(q))}}{A \vdash \exists p \exists q (p \neq q \wedge f(p) = f(q))}\ cut$$

where $\tau =$

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\overset{(\tau')}{A \vdash \mathbf{f(n_0 + n_1) = 0}, \mathbf{f(n_1 + n_0) = 1}}}{A \vdash \mathbf{f(n_0 + n_1) = 0}, \exists \mathbf{k}.\mathbf{f(n_1 + k) = 1}}\ \exists r}{A \vdash \exists \mathbf{k}.\mathbf{f(n_0 + k) = 0}, \exists \mathbf{k}.\mathbf{f(n_1 + k) = 1}}\ \exists r}{A \vdash \exists \mathbf{k}.\mathbf{f(n_0 + k) = 0}, \forall \mathbf{n} \exists \mathbf{k}.\mathbf{f(n + k) = 1}}\ \forall{:}\, r}{A \vdash \forall \mathbf{n} \exists \mathbf{k}.\mathbf{f(n + k) = 0}, \forall \mathbf{n} \exists \mathbf{k}.\mathbf{f(n + k) = 1}}\ \forall{:}\, r}{A \vdash \mathbf{I_0}, \forall \mathbf{n} \exists \mathbf{k}.\mathbf{f(n + k) = 1}}\ \mathrm{def}_{I_0}{:}\, r}{A \vdash \mathbf{I_0}, \mathbf{I_1}}\ \mathrm{def}_{I_1}{:}\, r$$

For $\tau' =$

$$\cfrac{\cfrac{\cfrac{f(n_0 + n_1) = 0 \vdash \mathbf{f(n_0 + n_1) = 0} \qquad \cfrac{\overset{(\text{Axiom})}{\vdash n_1 + n_0 = n_0 + n_1} \quad f(n_1 + n_0) = 1 \vdash \mathbf{f(n_1 + n_0) = 1}}{f(n_0 + n_1) = 1 \vdash \mathbf{f(n_1 + n_0) = 1}}\ =: l1}{f(n_0 + n_1) = 0 \vee f(n_0 + n_1) = 1 \vdash \mathbf{f(n_0 + n_1) = 0}, \mathbf{f(n_1 + n_0) = 1}}\ \vee{:}\, l}{\forall x (f(x) = 0 \vee f(x) = 1) \vdash \mathbf{f(n_0 + n_1) = 0}, \mathbf{f(n_1 + n_0) = 1}}\ \forall{:}\, l}{A \vdash \mathbf{f(n_0 + n_1) = 0}, \mathbf{f(n_1 + n_0) = 1}}\ \mathrm{def}_A{:}\, l$$

And for $i = 1, 2$ we define the proofs $\epsilon_i =$

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\psi \quad \eta_i}{\mathbf{f(s) = i}, \mathbf{f(t) = i} \vdash s \neq t \wedge f(s) = f(t)}\ \wedge{:}\, r}{\mathbf{f(s) = i}, \mathbf{f(t) = i} \vdash \exists q (s \neq q \wedge f(s) = f(q))}\ \exists{:}\, r}{\mathbf{f(s) = i}, \mathbf{f(t) = i} \vdash \exists p \exists q (p \neq q \wedge f(p) = f(q))}\ \exists{:}\, r}{\mathbf{f(n_0 + k_0) = i}, \exists \mathbf{k}.\mathbf{f(((n_0 + k_0) + 1) + k) = i} \vdash \exists p \exists q (p \neq q \wedge f(p) = f(q))}\ \exists{:}\, l}{\mathbf{f(n_0 + k_0) = i}, \forall \mathbf{n} \exists \mathbf{k}.\mathbf{f(n + k) = i} \vdash \exists p \exists q (p \neq q \wedge f(p) = f(q))}\ \forall{:}\, l}{\exists \mathbf{k}.\mathbf{f(n_0 + k) = i}, \forall \mathbf{n} \exists \mathbf{k}.\mathbf{f(n + k) = i} \vdash \exists p \exists q (p \neq q \wedge f(p) = f(q))}\ \exists{:}\, l}{\forall \mathbf{n} \exists \mathbf{k}.\mathbf{f(n + k) = i}, \forall \mathbf{n} \exists \mathbf{k}.\mathbf{f(n + k) = i} \vdash \exists p \exists q (p \neq q \wedge f(p) = f(q))}\ \forall{:}\, l}{\forall \mathbf{n} \exists \mathbf{k}.\mathbf{f(n + k) = i} \vdash \exists p \exists q (p \neq q \wedge f(p) = f(q))}\ c{:}\, l}{\mathbf{I_i} \vdash \exists p \exists q (p \neq q \wedge f(p) = f(q))}\ \mathrm{def}_{I_i}{:}\, l$$

for $s = n_0 + k_0$, $t = ((n_0 + k_0) + 1) + k_1$, and the proofs

$\psi =$

$$\cfrac{\cfrac{\overset{(\text{axiom})}{\vdash (n_0 + k_0) + (1 + k_1) = ((n_0 + k_0) + 1) + k_1} \quad \overset{(\text{axiom})}{n_0 + k_0 = (n_0 + k_0) + (1 + k_1) \vdash}}{n_0 + k_0 = ((n_0 + k_0) + 1) + k_1 \vdash}\ =: l1}{\vdash n_0 + k_0 \neq ((n_0 + k_0) + 1) + k_1}\ \neg{:}\, r$$

and $\eta_i =$

$$\dfrac{\mathbf{f}(\mathbf{s}) = \mathbf{i} \vdash f(s) = i \qquad \dfrac{\mathbf{f}(\mathbf{t}) = \mathbf{i} \vdash f(t) = i \qquad \overset{\text{(axiom)}}{\vdash i = i}}{\mathbf{f}(\mathbf{t}) = \mathbf{i} \vdash i = f(t)}\ =: r2}{\mathbf{f}(\mathbf{s}) = \mathbf{i}, \mathbf{f}(\mathbf{t}) = \mathbf{i} \vdash f(s) = f(t)}\ =: r2$$

The characteristic clause set is (after variable renaming)

$$\begin{array}{rcl} \mathrm{CL}(\varphi) & = & \{\vdash f(x+y) = 0, f(y+x) = 1; \quad (C_1) \\ & & f(x+y) = 0, f(((x+y)+1)+z) = 0 \vdash; \quad (C_2) \\ & & f(x+y) = 1, f(((x+y)+1)+z) = 1 \vdash\} \quad (C_3). \end{array}$$

The axioms used for the proof are the standard axioms of type $A \vdash A$ and instances of $\vdash x = x$, of commutativity $\vdash x + y = y + x$, of associativity $\vdash (x + y) + z = x + (y + z)$, and of the axiom

$$x = x + (1 + y) \vdash,$$

expressing that $x + (1 + y) \neq x$ for all natural numbers x, y.

The comparison with the analysis of Urban's proof formulated in **LK** (without equality) [3] shows that the proof based on equality is much more transparent. In fact the set of characteristic clauses contains only 3 elements (instead of 5), which are also simpler. This also facilitates the refutation of the clause set and makes the output proof simpler and more transparent. On the other hand, the analysis below shows that the mathematical argument obtained by cut-elimination is the same as in [3].

The program Otter found the following refutation of $\mathrm{CL}(\varphi)$ (based on hyperresolution only — without equality inference):

The first hyperesolvent, based on the clash sequence $(C_2; C_1, C_1)$, is

$$\begin{array}{rcl} C_4 & = & \vdash f(y+x) = 1,\ f(z + ((x+y)+1)) = 1, \text{ with the intermediary clause} \\ D_1 & = & f(((x+y)+1)+z) = 0 \vdash f(y+x) = 1. \end{array}$$

The next clash is sequence is $(C_3; C_4, C_4)$ which gives C_5 with intermediary clause D_2, where:

$$\begin{array}{rcl} C_5 & = & \vdash f(v'+u') = 1,\ f(v+u) = 1, \\ D_2 & = & f(x+y) = 1 \vdash f(v+u) = 1. \end{array}$$

Factoring C_5 gives C_6: $\vdash f(v+u) = 1$ (which roughly expresses that all fields are labelled by 1). The final clash sequence $(C_3; C_6, C_6)$ obviously results in the empty clause $\vdash$ with intermediary clause D_3: $f(((x+y)+1)+z) = 1 \vdash$. The hyperresolution proof ψ_3 in form of a tree can be obtained from the following resolution trees, where C' and ψ' stand for renamed variants of C and of ψ, respectively:

$\psi_1 :=$

$$\dfrac{\dfrac{C_1 \quad C_2}{D_1}\ \text{res} \qquad C_1'}{C_4}\ \text{res}$$

$\psi_2 :=$

$$\dfrac{\dfrac{\dfrac{C_3' \quad \psi_1}{D_2}\,\text{res} \qquad \psi_1'}{C_5}\,\text{res}}{C_6}\,\textit{factor}$$

$\psi_3 :=$

$$\dfrac{\dfrac{\psi_2 \quad C_3''}{D_3}\,\text{res} \qquad \psi_2'}{\vdash}\,\text{res}$$

Instantiation of ψ_3 by the uniform most general unifier σ of all resolutions gives a deduction tree $\psi_3\sigma$ in **LKDe**; indeed, after application of σ, resolution becomes cut and factoring becomes contraction. The proof $\psi_3\sigma$ is the skeleton of an **LKDe**-proof of the end-sequent with only atomic cuts. Then the leaves of the tree $\psi_3\sigma$ have to be replaced by the proof projections. E.g., the clause C_1 is replaced by the proof $\varphi[C_1]$, where $s = n_0 + n_1$ and $t = n_1 + n_0$:

$$\dfrac{\dfrac{\dfrac{\dfrac{f(s)=0 \vdash f(s)=0 \qquad \dfrac{\overset{\text{(Axiom)}}{\vdash t=s} \quad f(t)=1 \vdash f(t)=1}{f(s)=1 \vdash f(t)=1}\,=: l}{f(s)=0 \lor f(s)=1 \vdash f(s)=0, f(t)=1}\,\lor: l}{\forall x(f(x)=0 \lor f(x)=1) \vdash f(s)=0, f(t)=1}\,\forall: l}{A \vdash f(s)=0, f(t)=1}\,\text{def}_A: l}{A \vdash \exists p \exists q(p \neq q \land f(p)=f(q)), f(s)=0, f(t)=1}\,w: r$$

Furthermore C_2 is replaced by the projection $\varphi[C_2]$ and C_3 by $\varphi[C_3]$, where (for $i = 0, 1$) $\varphi[C_{2+i}] =$

$$\dfrac{\dfrac{\dfrac{\dfrac{\psi \quad \eta_i}{f(s)=i, f(t)=i \vdash s \neq t \land f(s)=f(t)}\,\land: r}{f(s)=i, f(t)=i \vdash \exists q(s \neq q \land f(s)=f(q))}\,\exists: r}{f(s)=i, f(t)=i \vdash \exists p \exists q(p \neq q \land f(p)=f(q))}\,\exists: r}{f(s)=i, f(t)=i, A \vdash \exists p \exists q(p \neq q \land f(p)=f(q))}\,w: l$$

Note that ψ, η_0, η_1 are the same as in the definition of ϵ_0, ϵ_1 above.

By inserting the σ-instances of the projections into the resolution proof $\psi_3\sigma$ and performing some additional contractions, we eventually obtain the desired proof φ' of the end-sequent

$$A \vdash \exists p \exists q(p \neq q \land f(p) = f(q))$$

with only *atomic* cuts. φ' no longer uses the lemmas that infinitely many cells are labelled by 0 and by 1, respectively.

4 Intended Extensions of `CERES` And Future Work

We plan to develop the following extensions of `CERES`:

- As the cut-free proofs are often very large and difficult to interpret, we intend to provide the possibility to analyze certain characteristics of the cut-free proof (which are simpler than the proof itself). An important example are Herbrand sequents which may serve to extract bounds from proofs (see e.g. [11]). We plan to develop algorithms for extracting Herbrand sequents (also from proofs of non-prenex sequents as indicated in [4]) and for computing interpolants, which allow for the replacement of implicit definitions by explicit ones according to Beth's Theorem.

- In the present version `CERES` eliminates all cuts at once. But - for the application to real mathematical proofs human user interaction is required — in particular only interesting cuts (i.e. lemmas of mathematical importance) deserve to be eliminated, others should be integrated as additional axioms.

- As CERES requires the skolemization of the end-sequent the original proof must be transformed to Skolem form. We plan to develop an efficient *reversed-skolemization*-algorithm, which transforms the theorem to be proved into its original form.

- A great challenge in the formal analysis of mathematical proofs lies in providing a suitable format for the input and output of proofs. We plan to improve our intermediary proof language and to move closer to the "natural" language of mathematical proofs.

To demonstrate the abilities of `CERES` and the feasibility of formalizing and analyzing complex proofs of mathematical relevance, we currently investigate a well known proof of the infinity of primes using topology (which may be found in [1]). Our aim is to eliminate the topological concepts from the proof by means of CERES, breaking it down to a proof solely based on elementary number arithmetic. This way one can obtain new insights into the construction of prime numbers contained in the topological proof.

References

[1] M. Aigner, G. M. Ziegler: *Proofs from THE BOOK.* Springer, 1998.

[2] P. B. Andrews: Resolution in Type Theory, *Journal of Symbolic Logic*, 36, pp. 414–432, 1971.

[3] M. Baaz, S. Hetzl, A. Leitsch, C. Richter, H. Spohr: Cut-Elimination: Experiments with CERES, *Proc. LPAR 2004*, pp. 481–495, Springer, 2004.

[4] M. Baaz, A. Leitsch: On Skolemization and Proof Complexity, *Fundamenta Informaticae*, 20(4), pp. 353–379, 1994.

[5] M. Baaz, A. Leitsch: Cut-Elimination and Redundancy-Elimination by Resolution, *Journal of Symbolic Computation*, 29, pp. 149–176, 2000.

[6] M. Baaz, A. Leitsch: Towards a Clausal Analysis of Cut-Elimination, *Journal of Symbolic Computation*, 41, pp. 381–410, 2006.

[7] E. Eder: Relative complexities of first-order calculi, Vieweg, 1992.

[8] G. Gentzen: Untersuchungen über das logische Schließen, *Mathematische Zeitschrift*, 39, pp. 405–431, 1934–1935.

[9] J. Y. Girard: *Proof Theory and Logical Complexity*, in Studies in Proof Theory, Bibliopolis, Napoli, 1987.

[10] A. Leitsch, C. Richter: Equational Theories in CERES, unpublished (available at `http://www.logic.at/ceres/`), 2005.

[11] H. Luckhardt: Herbrand-Analysen zweier Beweise des Satzes von Roth: polynomiale Anzahlschranken, *The Journal of Symbolic Logic*, 54, pp. 234–263, 1989.

[12] G. Polya: *Mathematics and plausible reasoning, Volume I: Induction and Analogy in Mathematics*, Princeton University Press, Princeton, New Jersey, 1954.

[13] G. Polya: *Mathematics and plausible reasoning, Volume II: Patterns of Plausible Inference*, Princeton University Press, Princeton, New Jersey, 1954.

[14] C. Urban: *Classical Logic and Computation* Ph.D. Thesis, University of Cambridge Computer Laboratory, 2000.