Proof Transformation by CERES *

Matthias Baaz[1], Stefan Hetzl[2], Alexander Leitsch[2], Clemens Richter[2], and 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.
In this paper we present an extension of CERES to a calculus **LKDe** which is stronger than the Gentzen calculus **LK** (it contains rules for introduction of definitions and equality rules). This extension makes it much easier to formalize mathematical proofs and increases the performance of the cut-elimination method. The system CERES already proved efficient in handling very large proofs.

1 Introduction

Proof analysis is a central mathematical activity which has proved crucial to the development of mathematics. 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[13, 14].
Cut-elimination introduced by Gentzen [9] 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 statements (lemmas), 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 proofs, especially to make implicit parameters explicit. In particular, cut free derivations allow for:

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

– 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 the replacement of implicit definitions by explicit ones 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 [10] 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. In contrast, such a transformation can be done purely automatically, which also might result in unexpected and interesting results [3]. 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. Indeed, it is non-uniqueness which makes computational experiments with cut-elimination interesting. The experiments can be considered as a source for a base of proofs in formal format which provide different mathematical and computational information.
CERES [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 a clause term which evaluates 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 only. In the system `CERES`[3] this method of cut-elimination has been implemented. The system is capable of dealing with formal proofs in **LK**, among them also very large ones.
The extension of CERES to a calculus containing definition-introduction and equality rules moves the system closer to real mathematical proofs. 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.

2 Extensions of Gentzen's LK

Gentzen's **LK** is the original calculus for which cut-elimination was defined. The original version of CERES in based on **LK** and several variants of it (we just refer to [6] and [7]). In formalizing mathematical proofs it turns out that **LK** (and also natural deduction) are not sufficiently close to real mathematical inference. First of all, the calculus **LK** lacks an efficient handling of equality (in fact equality axioms have to be added to the end-sequent). Due to the importance of equality this defect was apparent to proof theorists; e.g. Takeuti [15] gave an extension of

[3] available at http://www.logic.at/ceres/

LK to a calculus **LKe**, adding atomic equality axioms to the standard axioms of the form $A \vdash A$. The advantage of **LKe** over **LK** is that no new axioms have to be added to the end-sequent; on the other hand, in presence of the equality axioms, full cut-elimination is no longer possible, but merely reduction to *atomic cut*. But still **LKe** uses the same rules as **LK** as equality is axiomatized. On the other hand, in formalizing mathematical proofs, using equality as a *rule* is much more natural and concise. For this reason we choose the most natural equality rule, which is strongly related to paramodulation in automated theorem proving. Our approach differs from this in [17], where a unary equality rule is used (which does not directly correspond to paramodulation). The *equality rules* are:

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

for inference on the left and

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

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.
In CERES it is crucial that all nonlogical rules (which also work on atomic sequents) correspond to clausal inference rules in automated deduction. While cut and contraction correspond to resolution (and factoring, dependent on the version of resolution), the equality rules $=: l1, =: l2, =: r1, =: r2$ correspond to paramodulation, which is the most efficient equality rule in automated deduction [12]. Indeed, when we compute the most general unifiers and apply them to the paramodulation rule, then it becomes one of the rules $=: l1, =: l2, =: r1, =: r2$.

Perhaps the most significant tool in structuring mathematical proofs is the introduction of new concepts (formally definition-introduction). Though the use of definition introductions can be simulated by cuts, this simulation is rather unnatural and has a negative effect on the CERES-algorithm as will be explained in section 3 (definition introduction is a unary rule, while cut is a binary one). The *definition rules* directly correspond to the *extension principle* (see [8]) in predicate logic. It simply consists in introducing new predicate- and function symbols as abbreviations for formulas and terms. 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:

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

for arbitrary sequences of terms $t_1, \ldots, t_k$. Definition introduction is a simple and very powerful tool in mathematical practice. Note that the introduction of important concepts and notations like groups, integrals etc. can be formally described by introduction of new symbols. There are also definition introduction rules for new function symbols which are of similar type.

The *axiom system* for **LKDe** may be an arbitrary set of atomic sequents containing the sequents $A \vdash A$ (for atomic formulas A) which is closed under substitution. The only axioms which have to be added for equality are $\vdash s = s$ where s is an arbitrary term. So every axiom system has to contain the axioms $A \vdash A$ and $\vdash s = s$.

The calculus **LKDe** is **LK** extended by the equality rules and by the (infinite set of) definition-introduction rules. Clearly these extensions do not increase the logical expressivity of the calculus, but they make him much more compact and natural. To illustrate the rules defined above we give a simple example. The aim is to prove the (obvious) theorem that a number divides the square of a number b if it divides b itself. In the formalization below a and b are constant symbols and the predicate symbol D stands for "divides" and is defined by

$$D(x, y) \leftrightarrow \exists z.x * z = y.$$

The active equations are written in boldface.

$$\dfrac{\dfrac{\vdash (\mathbf{a} * \mathbf{z_0}) * \mathbf{b} = \mathbf{a} * (\mathbf{z_0} * \mathbf{b}) \qquad \dfrac{a * z_0 = b \vdash \mathbf{a} * \mathbf{z_0} = \mathbf{b} \quad \vdash b * b = b * b}{a * z_0 = b \vdash (a * z_0) * b = b * b} =: r2}{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{a * z_0 = b \vdash a * (z_0 * b) = b * b}{a * z_0 = b \vdash \exists z.a * z = b * b} \exists r}{\exists z.a * z = b \vdash \exists z.a * z = b * b} \exists : l}{\exists z.a * z = b \vdash D(a, b * b)} \mathrm{def}_D : r}{D(a, b) \vdash D(a, b * b)} \mathrm{def}_D : l}{\vdash D(a, b) \to D(a, b * b)} \to : r} =: r1$$

The axioms of the proof are: (1) an instance of the associativity law, (2) the equational axiom $\vdash b * b = b * b$ and the tautological standard axiom $a * z_0 = b \vdash a * z_0 = b$.

3 CERES on LKDe

3.1 Definitions and Results

Though CERES has been defined for **LK** originally, the method is very flexible and can be applied to virtually any sequent calculus for classical logic. Indeed, the extensions defined above, can easily built in without affecting the clarity and efficiency of the method. The central idea of CERES consists in analyzing the proof first, extracting a so-called characteristic clause set from the proof, and then using a resolution refutation of this set to obtain a proof with only atomic cuts. We consider the proofs in **LKDe** as directed trees with nodes which are labelled by sequents, where the root is labelled by the end-sequent. According to the inference rules, we distinguish binary and unary nodes. In an inference

$$\dfrac{\nu_1 : S_1 \quad \nu_2 : S_2}{\nu : S}\ x$$

where ν is labelled by S, ν_1 by S_1 and ν_2 by S_2, we call ν_1, ν_2 *predecessors* of ν. Similarly ν' is predecessor of ν in a unary rule if ν' labels the premiss and ν the consequent. Then the *predecessor relation* is defined as the reflexive and transitive closure of the relation above. Every node is predecessor of the root, and the axioms have only themselves as predecessors. For a formal definition of the concepts we refer to [6] and [7]. A similar relation holds between *formula occurrences* in sequents. Instead of a formal definition we give an example.
Consider the rule:

$$\frac{\forall x.P(x) \vdash P(a) \quad \forall x.P(x) \vdash P(b)}{\forall x.P(x) \vdash P(a) \wedge P(b)} \wedge\!: r$$

The occurrences of $P(a)$ and $P(b)$ in the premiss are *ancestors* of the occurrence of $P(a) \wedge P(b)$ in the consequent. $P(a)$ and $P(b)$ are called *auxiliary formulas* of the inference, and $P(a) \wedge P(b)$ the *main formula*. $\forall x.P(x)$ in the premisses are ancestors of $\forall x.P(x)$ in the consequent. Again the *ancestor relation* is defined by reflexive transitive closure.

Let Ω be the set of all occurrences of cut-formulas in sequents of an **LKDe**-proof φ. The cut- formulas are not ancestors of the formulas in the end-sequent, but they might have ancestors in the axioms (if the cuts are not generated by weakening only). The construction of the characteristic clause set is based on the ancestors of the cuts in the axioms. Note that *clauses* are just defined as atomic sequents. We define a set of clauses $\mathcal{C}_\nu$ for every node ν in φ inductively:

- If ν is an occurrence of an axiom sequent $S(\nu)$, and S' is the subsequent of $S(\nu)$ containing only the ancestors of Ω then $\mathcal{C}_\nu = \{S'\}$.
- Let ν' be the predecessor of ν in a unary inference then $\mathcal{C}_\nu = \mathcal{C}_{\nu'}$.
- Let ν_1, ν_2 be the predecessors of ν in a binary inference. We distinguish two cases
  (a) The auxiliary formulas of ν_1, ν_2 are ancestors of Ω. Then

$$\mathcal{C}_\nu = \mathcal{C}_{\nu_1} \cup \mathcal{C}_{\nu_2}.$$

  (b) The auxiliary formulas of ν_1, ν_2 are not ancestors of Ω. Then

$$\mathcal{C}_\nu = \mathcal{C}_{\nu_1} \times \mathcal{C}_{\nu_2}.$$

  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.

The *characteristic clause set* $\mathrm{CL}(\varphi)$ of φ is defined as $\mathcal{C}_{\nu_0}$, where ν_0 is the root. The definition of $\mathrm{CL}(\varphi)$ is the same as the one used for **LK** since both they contain only unary and binary rules. Note that unary rules have no effect on the characteristic clause set.

Theorem 1. *Let φ be a proof in* **LKDe**. *Then the clause set* $\mathrm{CL}(\varphi)$ *is equationally unsatisfiable.*

Remark 1. A clause set $\mathcal{C}$ is equationally unsatisfiable if $\mathcal{C}$ does not have a model where $=$ is interpreted as equality over a domain.

Proof. The proof is essentially the same as in [6]. Let ν be a node in φ and $S'(\nu)$ the subsequent of $S(\nu)$ which consists of the ancestors of Ω (i.e. of a cut). It is shown by induction that $S'(\nu)$ is **LKDe**-derivable from $\mathcal{C}_\nu$. If ν_0 is the root then, clearly, $S'(\nu_0) = \vdash$ and the empty sequent $\vdash$ is **LKDe**-derivable from the axiom set $\mathcal{C}_{\nu_0}$, which is just $\mathrm{CL}(\varphi)$. As all inferences in **LKDe** are sound over equational interpretations (where new symbols introduced by definition introduction have to be interpreted according to the defining equivalence), $\mathrm{CL}(\varphi)$ is equationally unsatisfiable. Note that, without the rules $=:l$ and $=:r$, the set $\mathrm{CL}(\varphi)$ is just unsatisfiable. Clearly the rules $=:l$ and $=:r$ are sound only over equational interpretations. $\diamond$

Note that, for proving Theorem 1, we just need the soundness of **LKDe** not its completeness.

The next steps in CERES are

(1) the computation of the proof projections $\varphi[C]$ w.r.t. clauses $C \in \mathrm{CL}(\varphi)$,
(2) the refutation of the set $\mathrm{CL}(\varphi)$, resulting in an RP-tree γ, i.e. in a deduction tree defined by the inferences of resolution and paramodulation, and
(3) "inserting" the projections $\varphi[C]$ into the leaves of γ.

Step (1) is done like in CERES for **LK**, i.e. we skip in φ all inferences where the auxiliary resp. main formulas are ancestors of a cut. Instead of the end-sequent S we get $S \circ C$ for a $C \in \mathrm{CL}(\varphi)$. The construction does not differ from this in [6] as the form of the rules do not matter.
Step (2) consists in ordinary theorem proving by resolution and paramodulation (which is equationally complete). For refuting $\mathrm{CL}(\varphi)$ any first-order prover like Vampire [4], SPASS [5] or Otter [6] can be used. By the completeness of the methods we find a refutation tree γ as $\mathrm{CL}(\varphi)$ is unsatisfiable by Theorem 1.
Step (3) makes use of the fact that, after computation of the simultaneous most general unifier of the inferences in γ, the resulting tree γ' is actually a *derivation in* **LKDe**! Indeed, after computation of the simultaneous unifier, paramodulation becomes $=:l$ and $=:r$ and resolution becomes cut in **LKDe**. Note that the definition rules, like the logical rules, do not appear in γ'. Now for every leaf ν in γ', which is labelled by a clause C' (an instance of a clause $C \in \mathrm{CL}(\varphi)$) we insert the proof projection $\varphi[C']$. The result is a proof with only atomic cuts.

The proof projection is only sound if the proof φ is skolemized, i.e. there are no strong quantifiers (i.e. quantifiers with eigenvariable conditions) in the end-sequent. If φ is not skolemized a priori it can be transformed into a skolemized proof φ' in polynomial (at most quadratic) time; for details see [5].

[4] http://www.vampire.fm/
[5] http://spass.mpi-sb.mpg.de/
[6] http://www-unix.mcs.anl.gov/AR/otter/

3.2 An Example

The example below is taken from [16]; 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 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 two cells labelled by the same number. $f(x) = 0$ expresses that the cell nr. 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. 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

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

where $\tau =$

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

For $\tau' =$

$$\dfrac{f(n_0 + n_1) = 0 \vdash \mathbf{f(n_0 + n_1) = 0} \quad \dfrac{\dfrac{(\text{Axiom})}{\vdash n_1 + n_0 = n_0 + n_1} \quad f(n_1 + n_0) = 1 \vdash \mathbf{f(n_1 + n_0) = 1}}{f(n_0 + n_1) = 1 \vdash \mathbf{f(n_1 + n_0) = 1}}\ =: l1}{f(n_0 + n_1) = 0 \vee f(n_0 + n_1) = 1 \vdash \mathbf{f(n_0 + n_1) = 0, f(n_1 + n_0) = 1}}\ \vee : l$$

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

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

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

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

and $\eta_i =$

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

The characteristic clause set is (after variable renaming)

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

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 this one is much more transparent. In fact the set of characteristic clauses contains only 3 clauses (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 CL(φ) (based on hyperresolution only – without equality inference):
The first hyperesolvent, based on the clash sequence $(C_2; C_1, C_1)$, is

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

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

$$C_5 = \vdash f(v'+u') = 1,\ f(v+u) = 1,$$
$$D_2 = f(x+y) = 1 \vdash f(v+u) = 1.$$

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:

$$\dfrac{\dfrac{C_1 \quad C_2}{D_1}\ res \qquad C_1'}{\psi_1{:}\ C_4}\ res \qquad \dfrac{\dfrac{\dfrac{C_3' \quad \psi_1}{D_2}\ res \qquad \psi_1'}{C_5}\ res}{\psi_2{:}\ C_6}\ factor \qquad \dfrac{\dfrac{\psi_2 \quad C_3''}{D_3}\ res \qquad \psi_2'}{\psi_3{:}\ \vdash}\ res$$

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 $t = n_1 + n_0$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{f(s) = 0 \vdash f(s) = 0 \qquad \dfrac{\text{(Axiom)}}{\vdash t = s \quad f(t) = 1 \vdash f(t) = 1}}{f(s) = 1 \vdash f(t) = 1}\ {=}{:}\,l1}{f(s) = 0 \vee f(s) = 1 \vdash f(s) = 0, f(t) = 1}\ \vee{:}\,l}{\forall x(f(x) = 0 \vee f(x) = 1) \vdash f(s) = 0, f(t) = 1}\ \forall{:}\,l}{A \vdash f(s) = 0, f(t) = 1}\ \mathrm{def}_A{:}\,l}{A \vdash \exists p \exists q(p \neq q \wedge f(p) = f(q)), f(s) = 0, f(t) = 1}\ w{:}\,r$$

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

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

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

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

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

4 The System CERES

The cut-elimination system CERES is written in ANSI-C++. There are two main tasks. On one hand, to compute an unsatisfiable set of clauses characterizing the cut formulas. This is done by automatically extracting the so-called characteristic clause term from a proof φ and computing the resulting *characteristic clause set* $\mathrm{CL}(\varphi)$. On the other hand, to generate a resolution refutation of $\mathrm{CL}(\varphi)$ by an external theorem prover[7], and to compute the necessary projection schemes of φ w.r.t. the clauses in $\mathrm{CL}(\varphi)$ actually used for the refutation. The properly instantiated projection schemes are concatenated, using the refutation obtained by the theorem prover as a skeleton of a proof with only atomic cuts.
Concerning the extension of **LK** to **LKDe**, equality rules appearing within the input proof are propagated to the projection schemes as any other binary rules. During theorem proving equality is treated by paramodulation (which is closely related to the equality rules in **LKDe**); 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.

[7] 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.

The system CERES expects an **LKDe** proof of a sequent S and a set of axioms as input, and computes a proof of S containing at most non 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**. Furthermore we implemented a proof viewer and proof editor 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.

5 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 nonprenex sequents as indicated in [4]) and for computing interpolants.
- 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.
- In the present version CERES eliminates all cuts at once. But — for the application to real mathematical proofs — only interesting cuts (i.e. lemmas) deserve to be eliminated, others should simply remain or 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 *de-skolemization*-algorithm, which transforms the theorem to be proved into its original form.

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.

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, LPAR 2004, *Lecture Notes in Artificial Intelligence*, pp. 481-495, 2005.
4. M. Baaz, A. Leitsch: On skolemization and proof complexity, *Fundamenta Informaticae*, 20(4), pp. 353–379, 1994.
5. M. Baaz, A. Leitsch: Cut normal forms and proof complexity, *Annals of Pure and Applied Logic*, 97, pp. 127-177, 1999.
6. M. Baaz, A. Leitsch: Cut-Elimination and Redundancy-Elimination by Resolution, *Journal of Symbolic Computation*, 29, pp. 149-176, 2000.
7. M. Baaz, A. Leitsch: Towards a Clausal Analysis of Cut-Elimination, *Journal of Symbolic Computation*, 41, pp. 381–410, 2006.
8. E. Eder: Relative complexities of first-order calculi, Vieweg, 1992.
9. G. Gentzen: Untersuchungen über das logische Schließen, *Mathematische Zeitschrift*, 39, pp. 405–431, 1934–1935.
10. J.Y. Girard: Proof Theory and Logical Complexity, in *Studies in Proof Theory*, Bibliopolis, Napoli, 1987.
11. H. Luckhardt: Herbrand-Analysen zweier Beweise des Satzes von Roth: polynomiale Anzahlschranken. *The Journal of Symbolic Logic* 54, 234–263, 1989.
12. R. Nieuwenhuis, A. Rubio: Paramodulation-based Theorem Proving, in *Handbook of Automated Reasoning*, eds. J.A. Robinson, A. Voronkov, pp. 371–443, Elsevier, 2001.
13. G. Polya. *Mathematics and plausible reasoning, Volume I: Induction and Analogy in Mathematics*. Princeton University Press, Princeton, New Jersey, 1954.
14. G. Polya. *Mathematics and plausible reasoning, Volume II: Patterns of Plausible Inference*. Princeton University Press, Princeton, New Jersey, 1954.
15. G. Takeuti: Proof Theory, North-Holland, Amsterdam, 2nd edition, 1987.
16. C. Urban: *Classical Logic and Computation* Ph.D. Thesis, University of Cambridge Computer Laboratory, 2000.
17. A. Degtyarev and A. Voronkov: Equality Reasoning in Sequent-Based Calculi *Handbook of Automated Reasoning* vol. I, ed. by A. Robinson and A. Voronkov, chapter 10, pp. 611-706, Elsevier Science, 2001.