System Description: The Proof Transformation System CERES*

Tsvetan Dunchev, Alexander Leitsch, Tomer Libal,
Daniel Weller, Bruno Woltzenlogel Paleo

Institute of Computer Languages (E185),
Vienna University of Technology, Favoritenstraße 9,
1040 Vienna, Austria
{cdunchev|leitsch|shaolin|weller|bruno}@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 extracting a set of clauses from a proof with cuts. Any resolution refutation of this set then serves as a skeleton of an ACNF, an LK-proof with only atomic cuts.
The system CERES, an implementation of the CERES-method has been used successfully in analyzing nontrivial mathematical proofs (see [4]).In this paper we describe the main features of the CERES system with special emphasis on the extraction of Herbrand sequents and simplification methods on these sequents. We demonstrate the Herbrand sequent extraction and simplification by a mathematical example.

1 Introduction

Proof analysis is a central mathematical activity which proved 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.

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

The development of the method CERES (cut-elimination by resolution) was inspired by the idea to fully automate cut-elimination on real mathematical proofs, with the aim of obtaining new interesting elementary proofs. While a

* supported by the Austrian Science Fund (project no. P22028-N13)

fully automated treatment proved successful for mathematical proofs of moderate complexity (e.g. the "tape proof" [2] and the "lattice proof" [9]), more complex mathematical proofs required an interactive use of CERES; this way we successfully analyzed Fürstenberg's proof of the infinitude of primes (see [4]) and obtained Euclid's argument of prime construction. Even in its interactive use CERES proved to be superior to the reductive cut-elimination due to additional structural information provided by the characteristic clause set (see below).

CERES [5, 6] is a cut-elimination method that is based on resolution. The method roughly works as follows: From the input proof φ of a sequent S a clause term is extracted and evaluated to an unsatisfiable set of clauses $\mathrm{CL}(\varphi)$, the *characteristic clause set*. A resolution refutation γ of $\mathrm{CL}(\varphi)$, which is obtained using a first-order theorem prover, serves as a skeleton for an (atomic cut normal form) ACNF ψ, a proof of S which contains at most atomic cuts. This method of cut-elimination has been implemented in the system `CERES`[1]. The system is capable of dealing with formal proofs in an extended version **LKDe** of **LK**, among them also very large ones.

However, the large size of ACNFs, automatically generated by `CERES`, turned out problematic in practice. Indeed, the aim is not only to produce an ACNF ψ from φ, but also to interpret ψ as a *mathematical* proof. In fact, the huge sizes of output proofs result from an inherent redundancy of formal calculi. Less redundant representations of the underlying mathematical arguments can be obtained by extracting a *Herbrand sequent* $H(\psi)$ from an ACNF ψ (see [9] and [11]). Thereby, $H(\psi)$ is a sequent consisting of instances of the quantifier-free parts of the formulas in S (we assume that S is skolemized). Though Herbrand sequents proved clearly superior to ACNFs in the analysis by humans, further simplifications of these sequents turned out important in practice. In this system description we lay specific emphasis on the extraction and simplification of Herbrand sequents, and illustrate transformations by an example.

By its high efficiency (the core of the method is first-order theorem proving by resolution and paramodulation), and by automatically extracting crucial structural information from proofs (e.g. the characteristic clause set) `CERES` proved useful in *automated proof mining*, thus contributing to an experimental culture of *computer-aided proof analysis* in mathematics.

2 The System `CERES`

The cut-elimination system `CERES` is written in ANSI-C++. The core functionality of `CERES`[3] allows the user to input a first order proof φ and obtain an $\mathrm{ACNF}(\varphi)$. The core system also includes two additional tools: the compiler `hlk`[2] for the intermediary proof language HandyLK and the proof viewer ProofTool[3].

[1] available at `http://www.logic.at/ceres/`
[2] `http://www.logic.at/hlk`
[3] `http://www.logic.at/prooftool`

The system follows a uniform data representation model for proofs and sequents in the form of XML using the proofdatabase DTD[4].

This functionality is extended in the current system by various optimizations on the resulted proof. CERES allows the computation of a Herbrand-sequent of the theorem and applies to it several simplification algorithms. Due to the important role of resolution provers in CERES, the system interfaces now with two additional provers: Prover9[5] and ATP[6].

The execution cycle starts with the mathematician using HandyLK to produce a formal **LKDe**-proof φ. **LKDe** is an extension of **LK**[2] to include definition and equality rules. HandyLK produces a formal proof by focusing on essential information as input. In particular, propositional inferences and context formulas are not required. HandyLK also simplifies the writing of proofs in a tree form by supplying meta-variables denoting proofs. Finally, HandyLK enables the definition of proof schemata by supporting parameterization over meta-terms and meta-formulas.

Since the restriction to skolemized proofs is crucial to the CERES-method, the system includes a proof skolemization transformation sk following Andrew's method[1]. sk(φ) is then parsed by CERES to produce CL(φ).

CL(φ) is given as input to one of the three possible theorem provers (Otter, Prover9 and ATP) and a resolution refutation is extracted. Otter and its successor Prover9 are both very efficient resolution provers that produce the resolution tree as output. Because a fully automatic refutation is not always possible, the interactive prover ATP was developed. ATP is a basic resolution prover that supports interaction with the user as well as customizable refinements. Interaction was used in order to validate and complete (manually obtained) refutations while customization can be used in order to implement specific and more efficient refinements[12].

In the last phase, CERES maps the refutation into an **LKDe**-proof such that resolution steps are mapped into atomic cuts, paramodulation into equality rules, etc. Moreover, it inserts proof projections (which are cut-free parts of the input proof[5, 6]) in order to obtain an ACNF(φ).

For a convenient analysis of the results, the system is equiped with the proof viewer/editor ProofTool. ProofTool is capable of presenting all data objects used in the process: the original and skolemized proof, the profile, the refutation, the ACNF-proof and the simplified Herbrand-sequent.

Herbrand-sequent extraction and simplification. As a post-process, the system now supports the extraction of a Herbrand-sequent $H(\varphi)$ from ACNF(φ). This process transforms the ACNF(φ) into a proof in an intermediary calculus $\mathbf{LKe}_A$, in which weakly quantified formulas are being replaced by an array of their instances. By "inverting" the transformation, we obtain $H(\varphi)$, which is still valid and contains all the desired information in a more compact form. $H(\varphi)$ is simplified further by the application of three algorithms[7].

[4] http://www.logic.at/ceres/xml/5.0/proofdatabase.dtd

[5] http://www.cs.unm.edu/~mccune/prover9/

[6] http://www.logic.at/atp

The first simplification algorithm S_{use} removes formulas containing irrelevant information: formulas which were introduced either by weakening or as the side-formulas of the main formulas of some other inferences.

An algebraic simplification S_{alg} is performed on the resulted sequent by normalizing each term with regard to a user-defined set of rewriting rules.

The last simplification algorithm S_{log} strips logically irrelevant formulas. As a Herbrand-sequent is always valid with regard to a given theory, we negate $H(\varphi)$ and apply a resolution theorem prover in order to obtain a refutation γ of the background theory and $\neg H(\varphi)$. The logically irrelevant formulas are all those formulas not appearing as leaves in γ.

3 An Example

In this section, we will treat a simple example from lattice theory. There are several different, but equivalent, definitions of the notion of *lattice*. Usually, the equivalence of several statements is shown by proving a cycle of implications. While this approach is elegant, it has the drawback that it does not provide *direct* proofs between the statements. Using cut-elimination, direct proofs of the implications between any two of the statements can be obtained. Hence we will demonstrate how the `CERES` system can be used to automatically generate such a proof via cut-elimination, how the Herbrand sequent extracted from the resulting proof can be simplified, and how the simplified Herbrand sequent provides a minimal explicit construction which was implicit in the original proof.

Lattice definitions. We will consider three definitions of the notion of lattice: two are algebraic, using 3-tuples $\langle L, \cap, \cup \rangle$, while the third one depends on the notion of partially ordered set $\langle S, \leq \rangle$.

Definition 1 (Algebraic Lattices). *A* semi-lattice *is a set L together with an operation $\circ$ fulfilling for all $x, y, z \in L$*

$$x \circ y = y \circ x \quad \textit{and} \quad x \circ x = x \quad \textit{and} \quad (x \circ y) \circ z = x \circ (y \circ z).$$

A L1-lattice is a set L together with operations $\cap$ (meet) *and $\cup$* (join) *s.t. both $\langle L, \cap \rangle$ and $\langle L, \cup \rangle$ are semi-lattices and for all $x, y \in L$*

$$x \cap y = x \leftrightarrow x \cup y = y.$$

A L2-lattice is a set L together with operations $\cap$ and $\cup$ s.t. both $\langle L, \cap \rangle$ and $\langle L, \cup \rangle$ are semi-lattices which for all $x, y \in L$ obey the absorption laws

$$(x \cap y) \cup x = x \quad \textit{and} \quad (x \cup y) \cap x = x$$

Definition 2 (Partial Order). *A binary relation $\leq$ on a set S is called* partial order *if for all $x, y, z \in S$*

$$x \leq x \quad \textit{and} \quad (x \leq y \wedge y \leq x) \rightarrow x = y \quad \textit{and} \quad (x \leq y \wedge y \leq z) \rightarrow x \leq z.$$

Definition 3 (Lattices based on Partial Orders). *A L3-lattice is a partially ordered set $\langle S, \leq \rangle$ s.t. for all $x, y \in S$ there exist a greatest lower bound $\cap$ and a least upper bound $\cup$, i.e. for all $z \in S$*

$$x \cap y \leq x \wedge x \cap y \leq y \wedge (z \leq x \wedge z \leq y \rightarrow z \leq x \cap y) \text{ and}$$
$$x \leq x \cup y \wedge y \leq x \cup y \wedge (x \leq z \wedge y \leq z \rightarrow x \cup y \leq z).$$

It is well known that the above three definitions of lattice are equivalent. We will formalize the proofs of $L1 \rightarrow L3$ and $L3 \rightarrow L2$ in order to extract a direct proof of $L1 \rightarrow L2$, i.e. one which does not use the notion of partial order.

Formalization of the Lattice Proof. The full **LKDe**-proof of $L1 \rightarrow L2$, formalized in the HandyLK language and compiled to **LKDe** by `hlk`, has 260 rules (214 rules, if structural rules, except cut, are not counted). It is too large to be displayed here. Below we show only a part of it, which is close to the end-sequent and depicts the main structure of the proof, based on the cut-rule with $L3$ as the cut-formula. The full proofs, conveniently viewable with `ProofTool`, are available on the website of `CERES`.

We note here that the proof is formalized in the theory of semi-lattices: it uses (instances of the open versions of) the semi-lattice axioms, and hence the theorem is valid in the theory $\mathcal{T}$ of semi-lattices, but not in general.

$$\dfrac{\dfrac{\dfrac{\dfrac{[p_R]}{\vdash R} \quad \dfrac{\dfrac{[p_{AS}]}{\vdash AS} \quad \dfrac{[p_T]}{\vdash T}}{\vdash AS \wedge T}\wedge:r}{\vdash R \wedge (AS \wedge T)}\wedge:r}{\vdash POSET}\,d:r \qquad \dfrac{\dfrac{[p_{GLB}] \quad [p_{LUB}]}{L1 \vdash GLB \wedge LUB}\wedge:r}{L1 \vdash POSET \wedge (GLB \wedge LUB)}\wedge:r}{L1 \vdash L3}\,d:r \qquad \dfrac{[p_3^2]}{L3 \vdash L2}}{L1 \vdash L2}\,cut$$

- $L1 \equiv \forall x \forall y((x \cap y) = x \rightarrow (x \cup y) = y) \wedge ((x \cup y) = y \rightarrow (x \cap y) = x)$
- $L2 \equiv \forall x \forall y (x \cap y) \cup x = x \wedge \forall x \forall y (x \cup y) \cap x = x$
- $L3 \equiv POSET \wedge (GLB \wedge LUB)$
- p_{AS}, p_T, p_R are proofs of, respectively, anti-symmetry, transitivity and reflexivity of $\leq$, which is defined as $x \leq y \equiv x \cap y = x$.
- p_{GLB} and p_{LUB} are proofs that $\cap$ and $\cup$ are greatest lower bound and greatest upper bound, respectively.
- p_3^2 is a proof that $L3$-lattices are $L2$-lattices.

Cut-Elimination of the Lattice Proof. Prior to cut-elimination, the formalized proof is skolemized by `CERES`, resulting in a proof of the skolemized end-sequent $L1 \vdash (s_1 \cap s_2) \cup s_1 = s_1 \wedge (s_3 \cup s_4) \cap s_3 = s_3$, where s_1, s_2, s_3 and s_4 are skolem constants for the strongly quantified variables of $L2$. Then `CERES` eliminates cuts (using Prover9 for computing the refutation), producing a proof in ACNF (available for visualization with `ProofTool` in the website of `CERES`).

Herbrand Sequent Extraction of the ACNF of the Lattice Proof. As our proof under investigation uses axioms of the theory $\mathcal{T}$, also our ACNF φ is a proof in $\mathcal{T}$, and the Herbrand sequent $H(\varphi)$, extracted from φ according to

the algorithm from [9], is valid in $\mathcal{T}$. After the application of S_{use}, $H(\varphi)$ becomes the sequent H':

$$
\begin{array}{l}
s_1 \cup (s_1 \cup (s_1 \cap s_2)) = s_1 \cup (s_1 \cap s_2) \rightarrow s_1 \cap (s_1 \cup (s_1 \cap s_2)) = s_1, \\
s_1 \cap s_1 = s_1 \rightarrow s_1 \cup s_1 = s_1, \\
(s_1 \cap s_2) \cap s_1 = s_1 \cap s_2 \rightarrow (s_1 \cap s_2) \cup s_1 = s_1, \\
(s_1 \cup (s_1 \cap s_2)) \cup s_1 = s_1 \rightarrow (s_1 \cup (s_1 \cap s_2)) \cap s_1 = s_1 \cup (s_1 \cap s_2), \\
(s_3 \cup (s_3 \cup s_4) = s_3 \cup s_4 \rightarrow s_3 \cap (s_3 \cup s_4) = s_3 \\
\vdash L2, (s_1 \cap s_2) \cup s_1 = s_1 \wedge (s_3 \cup s_4) \cap s_3 = s_3
\end{array}
$$

Observe that S_{use} has pruned some subformulas from $H(\varphi)$: as $H(\varphi)$ is a Herbrand sequent of our theorem, its antecedent only contains instances of $L1$. Observe that the formulas in the antecedent of H' are not instances of $L1$ (some conjuncts where deleted), but still H' is valid in $\mathcal{T}$ and contains the relevant information from the ACNF.

By S_{log}, H' is further pruned and four formulas are deleted, finally resulting in the Herbrand sequent H''

$$
\begin{array}{l}
(s_1 \cap s_2) \cap s_1 = s_1 \cap s_2 \rightarrow (s_1 \cap s_2) \cup s_1 = s_1, \\
s_3 \cup (s_3 \cup s_4) = s_3 \cup s_4 \rightarrow s_3 \cap (s_3 \cup s_4) = s_3 \\
\vdash (s_1 \cap s_2) \cup s_1 = s_1 \wedge (s_3 \cup s_4) \cap s_3 = s_3
\end{array}
$$

H'' is minimal in the sense that if we remove a formula from H'', the resulting sequent is not valid in $\mathcal{T}$ anymore. This is not the case in general; minimality is determined by the resolution refutation computed in S_{log}.

H'' now gives rise to an elementary proof of the theorem $L1 \vdash L2$: Our goal is to prove (1) $(s_1 \cap s_2) \cup s_1 = s_1$ and (2) $(s_3 \cup s_4) \cap s_3 = s_3$. For (1), we prove $(s_1 \cap s_2) \cap s_1 = s_1 \cap s_2$ using idempotency, associativity and commutativity of $\cap$ and conclude with $L1$. For (2), we prove $s_3 \cup (s_3 \cup s_4) = s_3 \cup s_4$ using idempotency and associativity of $\cup$. We conclude with $L1$ and commutativity of $\cap$.

Summarizing, the `CERES` system has taken as input a proof in lattice theory which used the auxiliary notion of partial order. By cut-elimination, a new proof not using any auxiliary notions is computed. From this proof, a Herbrand sequent summarizing the mathematical information (i.e. the instantiations) is extracted. This sequent is further pruned, resulting in a compact presentation of the relevant mathematical ideas of the proof (in this case, an algebraic construction not visible in the input proof). In the present example, the algorithm S_{alg} was not used: we refer to [7] for further examples.

4 Summary of Recent Improvements and Future Work

The simplification of Herbrand sequents is one of the most important features recently added to the `CERES`. Redundancy in the resulting sequent is significantly reduced and the terms are rewritten to a more readable normal-form. However, even if the simplified Herbrand sequent is completely redundancy-free, it can still be large and, consequently, the user can still face difficulties to formulate

the informal mathematical proof that it summarizes. Our experience [9] indicates that enriching the Herbrand sequent with certain kinds of links between its atomic sub-formulas might provide helpful information to the user. These links would resemble the axiom links of proof nets or the connections of the connection method, and they could be obtained by analyzing either the axioms of the ACNF or the resolved literals in the refutation used in the simplifcation of the Herbrand sequent. The theoretical investigation and the implementation of such links remains for future work.

As mentioned in Section 2, `CERES` relies on resolution theorem provers to refute characteristic clause sets or profiles. While only Otter was originally supported, now it is also possible to use `CERES` together with Prover9 and ATP. For the future, we intend to support the TPTP/TSTP format. It is also desirable to interface `CERES` with proof assistants like Isabelle, Coq, PVS and Mizar. On the one hand, `CERES` would benefit from the large libraries of proofs written in the languages of these systems, and on the other hand, these systems would benefit from the proof transformations of `CERES`. The main obstacle has always been the differences in the logical frameworks used by each of these systems and by `CERES`.

Our most ambitious current goal is an extension of `CERES` to higher-order logic [10]. This task encountered a few hard theoretical obstacles, such as the difficulty of skolemizing higher-order proofs [10], as well as practical obstacles, such as the need to change the core data-structures of `CERES` in order to support higher-order formulas. This led to the decision of implementing a new version of `CERES`, currently under development in the more flexible language Scala.

References

1. Andrews, P. B.: Resolution in Type Theory. J. of Symbolic Logic, 36, 414–432 (1971)
2. Baaz, M., Hetzl, S., Leitsch, A., Richter, C., Spohr, H.: Proof Transformation by CERES. Springer Lecture Notes in Computer Science, 4108, 82–93 (2006)
3. Baaz, M., Hetzl, S., Leitsch, A., Richter, C., Spohr, H.: System Description: The Cut-Elimination System CERES. Proc. ESCoR 2006, 159–167 (2006)
4. Baaz, M., Hetzl, S., Leitsch, A., Richter, C., Spohr, H.: CERES: An analysis of Fürstenberg's proof of the infinity of primes. Th. Co. Sci., 403, 160–175 (2008)
5. Baaz, M., Leitsch, A.: Cut-Elimination and Redundancy-Elimination by Resolution. Journal of Symbolic Computation 29, 149–176 (2000)
6. Baaz, M., Leitsch, A.: Towards a Clausal Analysis of Cut-Elimination. Journal of Symbolic Computation, 41, 381–410 (2006)
7. Dunchev, T.: Simplification of Herbrand Sequents. Master Thesis (2009)
8. Gentzen, G.: Untersuchungen über das logische Schliessen. Mathematische Zeitschrift, 39, 176–210, 405–431 (1934-1935)
9. Hetzl, S., Leitsch A., Weller, D., Woltzenlogel Paleo, B.: Herbrand Sequent Extraction. Lecture Notes in Computer Science 5144, 462–477 (2008)
10. Hetzl, S., Leitsch, A., Weller D., Woltzenlogel Paleo, B.: A Clausal Approach to Proof Analysis in Second-Order Logic. Logical Foundations of Computer Sci. (2009)
11. Woltzenlogel Paleo, B.: Herbrand Sequent Extraction. VDM-Verlag (2008)
12. Woltzenlogel Paleo, B.: A General Analysis of Cut-Elimination by CERes. PhD Thesis (2009)