J Autom Reasoning (2019) 62:393–430
https://doi.org/10.1007/s10817-018-9453-9

Extraction of Expansion Trees

Alexander Leitsch[1] · **Anela Lolic**[2]

Received: 14 February 2017 / Accepted: 16 January 2018 / Published online: 27 January 2018

Abstract We define a new method for proof mining by CERES (cut-elimination by resolution) that is concerned with the extraction of expansion trees in first-order logic (see Miller in Stud Log 46(4):347–370, 1987) with equality. In the original CERES method expansion trees can be extracted from proofs in normal form (proofs without quantified cuts) as a post-processing of cut-elimination. More precisely they are extracted from an ACNF, a proof with at most atomic cuts. We define a novel method avoiding proof normalization and show that expansion trees can be extracted from the resolution refutation and the corresponding proof projections. We prove that the new method asymptotically outperforms the standard method (which first computes the ACNF and then extracts an expansion tree). Finally we compare an implementation of the new method with the old one; it turns out that the new method is also more efficient in our experiments.

Keywords Cut-elimination · Proof mining · Herbrand sequent · Expansion tree

1 Introduction

Proof analysis and proof mining are central mathematical activities. Extracting additional mathematical information from existing proofs plays an important role in the process of proof mining. Mathematical proofs in general are based on the structuring of reasoning by intermediate statements (lemmas). The drawback of the use of lemmas is that only their truth but not their proofs are reflected in the derivation of their end-sequents. These proofs, however,

✉ Anela Lolic
anela@logic.at

Alexander Leitsch
leitsch@logic.at

1 Institute of Computer Languages (E185), Vienna University of Technology, Favoritenstrasse 9, 1040 Vienna, Austria

2 Institute of Discrete Mathematics and Geometry, Vienna University of Technology, Wiedner Hauptstrasse 8 - 10, 1040 Vienna, Austria

may contain important mathematical information which can be extracted only from the proofs of these lemmas. One of the most important theorems in mathematical logic is Gentzen's Hauptsatz [15]. It states that lemmas (cuts) can be eliminated from first-order derivations, resulting in a lemma-free proof combining all subproofs of the original derivation.

The result of cut-elimination is a purely combinatorial proof. These combinatorial proofs can be used to extract explicit mathematical information. Proofs can be transformed in a way such that this information becomes visible. Such a transformation for cut-free **LK**-proofs of prenex end-sequents was given by Gentzen [15] in the mid-sequent theorem. It basically states that a proof φ can be transformed into a proof φ' such that φ' contains a so-called mid-sequent, that splits the proof into a propositional part and a part with quantifier inferences. The mid-sequent is propositionally valid and contains the instantiations of the quantifiers needed to prove the end-sequent. These instantiations may contain crucial mathematical information.

Cut-elimination plays a key role in the analysis of mathematical proofs. A prominent example is Girard's analysis (see in [16]) of Fürstenberg and Weiss' topological proof [14] of van der Waerden's theorem [27] on partitions. After cut-elimination was applied to the proof of Fürstenberg and Weiss, the result was van der Waerden's original elementary proof.

Girard's analysis of the proof of Fürstenberg and Weiss was carried out by hand within mathematical meta-language. However, in automated proof analysis, formal proofs are vital. Therefore the first step in automated proof analysis consists in formalizing the mathematical proofs (typically expressed in traditional mathematical language). The next steps are algorithmic cut-elimination and, finally, the interpretation of the resulting formal proof.

For automated proof analysis of mathematical proofs, the cut-elimination method CERES (Cut-Elimination by RESolution) was developed (see [4,5]). CERES substantially differs from the traditional reductive cut-elimination methods a la Gentzen. In the reductive methods cuts are eliminated by stepwise reduction of cut-complexity. These methods always identify the uppermost logical operator in the cut-formula and either eliminate it directly (grade reduction) or indirectly (rank reduction). It is typical for such a method that the cut formulas are "peeled" from the outside till only atomic cuts are left. These methods are local in the sense that only a small part of the whole proof is analyzed, namely the derivation corresponding to the introduction of the uppermost logical operator. As a consequence, many types of redundancy in proofs are left undetected in the reductive methods, leading to an unfortunate computational behavior. In contrast, the method CERES to be presented in Sect. 3 is based on a structural analysis of the whole proof. Here all cut-derivations in an **LK**-proof φ of a sequent S are analyzed simultaneously. The interplay of binary rules, which produce ancestors of cut formulas and those which do not, defines a structure which can be represented as a set of clauses $\mathrm{CL}(\varphi)$. $\mathrm{CL}(\varphi)$ is always unsatisfiable and thus admits resolution refutations. A resolution refutation γ of $\mathrm{CL}(\varphi)$ may serve as a skeleton of an **LK**-proof of S with only atomic cuts. The proof itself (a CERES normal form) is obtained by replacing clauses in γ by so-called proof projections of φ. To handle predicate logic with equality the calculi can be extended by equality rules; instead of resolution refutations we obtain refutations by resolution and paramodulation (for details see [2]). CERES is a semi-semantical method of cut-elimination (see [7]). A detailed description of CERES, a comparison with reductive methods, its extensions and a complexity analysis of the method can be found in the book [6].

CERES has been applied to real mathematical proofs. The most interesting application was the analysis of Fürstenberg's proof of the infinitude of primes [13] where, as a result of cut-elimination by CERES, Euclid's original argument of prime construction was obtained.

The last step in automated proof analysis consists in the interpretation of the result. In this interpretation it is crucial to obtain compact and meaningful information rather than a full (and typically very long) formal proof. The relevant information can be bounds for variables

that are used in the proof or even programs representing its algorithmic content. Actually, it is possible to extract functionals, based on Gödel's dialectica interpretation [17], and construct programs from proofs in Peano arithmetic; see [8,9] for applications to mathematical proofs.

Another structure representing explicit information are mid-sequents (also called Herbrand sequents). Herbrand's theorem, [10,18], provides one of the most fundamental insights of logic and characterizes the validity of a formula in classical first-order logic by the existence of a propositional tautology composed of instances of that formula. Roughly speaking, Herbrand sequents are compact structures encoding the essence of proofs with prenex end-sequents. Hence in mathematical proof analysis it is frequently more important to extract Herbrand sequents than full formal proofs (which may be too large to be interpreted). There are efficient algorithms for extracting Herbrand sequents from cut-free proofs, see e.g. [20]. Though every formula (and any sequent) can be transformed to prenex form such a transformation is unnatural and can have a disastrous impact on proof complexity (see [3]). Thus it is vital to extend the methods to non-prenex formulas and sequents. Miller [23] developed the structure of expansion trees (and expansion proofs) generalizing the derivation of end-sequents from a mid-sequent in the prenex case. The so-called deep function of an expansion proof generalizes the mid-sequent itself. As expansion proofs abstract from propositional reasoning they provide compact and explicit information about the mathematical content of formal cut-free proofs.

The result of the method CERES is a CERES normal form, which is a (typically very long) formal proof with at most atomic cuts. This proof can then be used for further investigation and particularly for the extraction of Herbrand sequents and expansion proofs in order to obtain compact information. In the ordinary CERES-method, expansion proofs can be extracted from an ACNF. In this paper we show that even the construction of an ACNF can be avoided in computing the expansion proofs. In particular, we prove that the expansion proof of the CERES normal form can be constructed from the partial expansion proofs of the projections obtained by CERES, after deleting the clause parts. A ground refutation of the characteristic clause set and the projections suffice for the extraction of expansion proofs making the construction of the CERES normal form itself obsolete. This improvement yields a gain in asymptotic complexity. In particular we show that the new method outperforms the old one (quadratic versus cubic) and that the complexity of the new method can never be higher than that of the old one. Finally we describe an implementation of the new method and the traditional one (both methods are implemented in the Gapt system [12]) and show how they can be used to extract expansion proofs from proofs. We compare the implementations of the two methods and it turns out that even for very small and simple proofs a visible speed-up in computing time can be obtained.

2 Preliminaries

2.1 Sequents and Sequent Calculus

We define an extended version of Gentzen's calculus **LK** in predicate logic with equality and arbitrary function symbols.

Definition 1 Let Γ and Δ be two multi-sets of formulas and $\vdash$ be a symbol not belonging to the logical language. Then $\Gamma \vdash \Delta$ is called a sequent.

If $S_1 : \Gamma \vdash \Delta$ and $S_2 : \Pi \vdash \Lambda$ are sequents we define the *concatenation* of S_1 and S_2 (notation $S_1 \circ S_2$) as $\Gamma, \Pi \vdash \Delta, \Lambda$.

Definition 2 Let $S : A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ be a sequent and $\mathcal{M}$ be an interpretation over the signature of $\{A_1, \ldots, A_n, B_1, \ldots, B_m\}$. Then S is valid in $\mathcal{M}$ if the formula $(A_1 \wedge \ldots \wedge A_n) \rightarrow (B_1 \vee \cdots \vee B_m)$ is valid in $\mathcal{M}$. S is called valid if S is valid in all interpretations.

Definition 3 Let $S : A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ be a sequent. S is called a weakly quantified sequent if there is no $\exists$ quantifier of positive polarity in some formula A_i $(1 \leq i \leq n)$ and there is no $\forall$ quantifier of positive polarity in some formula B_j $(1 \leq j \leq m)$.

Definition 4 (Calculus $\mathbf{LK}_=$) Basically we use Gentzen's version of **LK** [15] but extend it by equality rules as in [2] and call the calculus $\mathbf{LK}_=$. Since we consider multi-sets of formulas, we do not need exchange or permutation rules. There are two groups of rules, the logical and the structural ones. All rules except the cut have left and right versions, denoted by l and r, respectively. The binary rules are of multiplicative type, i.e. no auto-contraction of the context is applied. In the following, A and B denote formulas whereas $\Gamma, \Delta, \Pi, \Lambda$ denote multi-sets of formulas.

The logical rules:

$\wedge$-introduction

$$\frac{A, B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta} \wedge_l \qquad \frac{\Gamma_1 \vdash \Delta_1, A \qquad \Gamma_2 \vdash \Delta_2, B}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A \wedge B} \wedge_r$$

$\vee$-introduction

$$\frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B} \vee_r \qquad \frac{A, \Gamma_1 \vdash \Delta_1 \qquad B, \Gamma_2 \vdash \Delta_2}{A \vee B, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \vee_l$$

$\rightarrow$-introduction

$$\frac{\Gamma_1 \vdash \Delta_1, A \qquad B, \Gamma_2 \vdash \Delta_2}{A \rightarrow B, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \rightarrow_l \qquad \frac{A, \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \rightarrow B} \rightarrow_r$$

$\neg$-introduction

$$\frac{\Gamma \vdash \Delta, A}{\neg A, \Gamma \vdash \Delta} \neg_l \qquad \frac{A, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \neg A} \neg_r$$

$\forall$-introduction

$$\frac{A\{x \leftarrow t\}, \Gamma \vdash \Delta}{(\forall x)A(x), \Gamma \vdash \Delta} \forall_l \qquad \frac{\Gamma \vdash \Delta, A\{x \leftarrow \alpha\}}{\Gamma \vdash \Delta, (\forall x)A(x)} \forall_r$$

where t is an arbitrary term that does not contain any variables which are bound in A and α is a free variable which may not occur in Γ, Δ, A. α is called an eigenvariable.

$\exists$-introduction

$$\frac{A\{x \leftarrow \alpha\}, \Gamma \vdash \Delta}{(\exists x)A(x), \Gamma \vdash \Delta} \exists_l \qquad \frac{\Gamma \vdash \Delta, A\{x \leftarrow t\}}{\Gamma \vdash \Delta, (\exists x)A(x)} \exists_r$$

where the variable conditions for $\exists_l$ are the same as those for $\forall_r$ and similarly for $\exists_r$ and $\forall_l$. The quantifier-rules $\forall_l, \exists_r$ are called *weak*, the rules $\exists_l, \forall_r$ *strong*.

The structural rules:

weakening

$$\frac{\Gamma \vdash \Delta}{A, \Gamma \vdash \Delta} w_l \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A} w_r$$

contraction

$$\frac{A, A, \Gamma \vdash \Delta}{A, \Gamma \vdash \Delta} c_l \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A} c_r$$

cut

$$\frac{\Gamma_1 \vdash \Delta_1, A^m \qquad A^n, \Gamma_2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\, cut(A, m, n)$$

where $m, n \geq 1$.

The equality rules:

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

for inference on the left and

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

on the right, where Λ denotes a position of a subterm 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.

Axioms:
Any set of atomic sequents which is closed under substitution and contains the sequent $\vdash x = x$ (and thus all sequents of the form $\vdash t = t$ for arbitrary terms t) is admitted as an axiom set. We define the set TAUT $= \{A \vdash A \mid A \text{ is an atom}\}$ and the axiom set $Ax =$ TAUT $\cup \{\vdash t = t \mid t \text{ a term}\}$, which is called the standard axiom set.

An $\mathbf{LK}_=$-proof from a set of axioms $\mathcal{A}$ is a tree formed according to the rules of $\mathbf{LK}_=$ such that all leaves are in $\mathcal{A}$. The formulas in Γ, Δ, Π, Λ are called context formulas. The formulas in the upper sequents that are not context formulas are called auxiliary formulas and those in the lower sequents are called main formulas. The auxiliary formulas of a cut-rule are also called cut-formulas. If $\mathcal{S}$ is a set of sequents, then an **LK**-refutation of $\mathcal{S}$ is an **LK**-tree π where the end-sequent of π is the empty sequent and the leaves of π are either axioms of the standard axiom set or sequents in $\mathcal{S}$.

For the proof transformations in this paper we need the concept of ancestors of nodes in a proof tree and formula occurrences within sequents occurring in proofs.

Definition 5 (*Formula ancestor*) Let ν be a formula occurrence in a sequent calculus proof φ. Then ν is an ancestor of itself in φ (the relation is reflexive). If ν is a principal formula occurrence of an inference then the occurrences of the auxiliary formula (formulas) in the premises are formula ancestors of ν. If ν is not a principal occurrence then the corresponding occurrences in contexts of the (premise) premises are formula ancestors of ν. The formula ancestor relation is then defined as the transitive closure.

Definition 6 (*Sequent ancestor*) Let ν be an occurrence of a sequent in a sequent calculus proof φ. Then ν is a sequent ancestor of itself (reflexivity). If ν corresponds to the conclusion of an inference with premises μ_1, μ_2 (μ) then μ_1, μ_2 (μ) are sequent ancestors of ν in φ. The sequent ancestor relation is then defined as the transitive closure.

Definition 7 (*Clause*) A sequent $\Gamma \vdash \Delta$ is called a clause if Γ and Δ are multisets of atoms.

Definition 8 (*PR-calculus*) The PR-calculus works on clauses and consists of the following rules:

1. the resolution rule:

$$\frac{\Gamma \vdash \Delta, A_1, \ldots, A_m \quad \Gamma', A'_1, \ldots, A'_n \vdash \Delta'}{\Gamma\sigma, \Gamma'\sigma \vdash \Delta\sigma, \Delta'\sigma} \; R$$

   Where $n, m \geq 1$ and σ is a most general unifier of $\{A_1, \ldots, A_m, A'_1, \ldots, A'_n\}$. It is also required that $\Gamma \vdash \Delta, A$ and $\Gamma', A' \vdash \Delta'$ are variable disjoint.
2. the paramodulation rules:
   We assume that the two clauses in the premises are always variable disjoint and that σ is a most general unifier of $\{s, s'\}$.

$$\frac{\Gamma_1 \vdash \Delta_1, s = t \quad A[s']_\Lambda, \Gamma_2 \vdash \Delta_2}{A[t]_\Lambda\sigma, \Gamma_1\sigma, \Gamma_2\sigma \vdash \Delta_1\sigma, \Delta_2\sigma} \quad \frac{\Gamma_1 \vdash \Delta_1, t = s \quad A[s']_\Lambda, \Gamma_2 \vdash \Delta_2}{A[t]_\Lambda\sigma, \Gamma_1\sigma, \Gamma_2\sigma \vdash \Delta_1\sigma, \Delta_2\sigma}$$

   for inference on the left side of the clauses and

$$\frac{\Gamma_1 \vdash \Delta_1, s = t \quad \Gamma_2 \vdash \Delta_2, A[s']_\Lambda}{\Gamma_1\sigma, \Gamma_2\sigma \vdash \Delta_1\sigma, \Delta_2\sigma, A[t]_\Lambda\sigma} \quad \frac{\Gamma_1 \vdash \Delta_1, t = s \quad \Gamma_2 \vdash \Delta_2, A[s']_\Lambda}{\Gamma_1\sigma, \Gamma_2\sigma \vdash \Delta_1\sigma, \Delta_2\sigma, A[t]_\Lambda\sigma}$$

   for the right side, where Λ denotes a position of a subterm where s' is replaced by t. We call $s = t$ the *active equation* of the rules.

A *PR-derivation* from a set of clauses $\mathcal{C}$ is a tree derivation based on the rules above where all clauses in the leaves are variants of clauses in $\mathcal{C}$. A *PR-derivation* of $\vdash$ from $\mathcal{C}$ is called a *PR-refutation* of $\mathcal{C}$.

Definition 9 Let φ be a proof and η be some arbitrary inference in φ. We say that η goes into the end-sequent of φ if the principal formula of η is an ancestor of the end-sequent. In this case, η cannot be a cut. η goes into a cut otherwise.

2.2 Expansion Trees

Expansion trees, first introduced in [23], are natural structures representing the instantiated variables for quantified formulas.

These structures record the substitutions for quantifiers in the original formula and the formulas resulting from instantiations. Expansion trees may contain logical connectives as well as the new connective $+^t$, where t is a term. Informally, an expression of the kind $QxA(x) +^{t_1} E_1 +^{t_2} \cdots +^{t_n} E_n$ is an expansion tree, where $Q \in \{\forall, \exists\}$ and $t_1, \ldots, t_n$ are terms such that this expansion tree represents the result when instantiating the quantified expression $QxA(x)$ with the terms $t_1, \ldots, t_n$ to get the structures E_i. E_i is again an expansion tree representing $A(t_i)$ for $i = 1, \ldots, n$.

Our definition is a modified one as our proofs are skolemized and we do not have quantifiers with eigenvariable conditions. The definition below takes care that only trees with weak quantifiers are constructed.

Definition 10 Expansion trees, dual expansion trees and a function Sh (shallow) which maps expansion trees to formulas are defined inductively as follows:

1. If A is a quantifier-free formula then A is an expansion tree (and a dual expansion tree) for A and $Sh(A) = A$.
2. If E is an expansion tree then $\neg E$ is a dual expansion tree and $Sh(\neg E) = \neg Sh(E)$.
3. If E is a dual expansion tree then $\neg E$ is an expansion tree and $Sh(\neg E) = \neg Sh(E)$.
4. If E_1 and E_2 are (dual) expansion trees, then $E_1 \wedge E_2$, $E_1 \vee E_2$ are (dual) expansion trees and $Sh(E_1 \wedge E_2) = Sh(E_1) \wedge Sh(E_2)$, the same for $\vee$.

5. If E_1 is a dual expansion tree and E_2 is an expansion tree then $E_1 \to E_2$ is an expansion tree and $Sh(E_1 \to E_2) = Sh(E_1) \to Sh(E_2)$.
6. If E_1 is an expansion tree and E_2 is a dual expansion tree then $E_1 \to E_2$ is a dual expansion tree and $Sh(E_1 \to E_2) = Sh(E_1) \to Sh(E_2)$.
7. Let $A(x)$ be a formula and $t_1, \ldots, t_n$ $(n \geq 1)$ be a list of terms. Let $E_1, \ldots, E_n$ be expansion trees with $Sh(E_i) = A(t_i)$ for $i = 1, \ldots, n$; then $\exists x A(x) +^{t_1} E_1 +^{t_2} \cdots +^{t_n} E_n$ is an expansion tree with $Sh(\exists x A(x) +^{t_1} E_1 +^{t_2} \cdots +^{t_n} E_n) = \exists x A(x)$.
8. Let $A(x)$ be a formula and $t_1, \ldots, t_n$ $(n \geq 1)$ be a list of terms. Let $E_1, \ldots, E_n$ be dual expansion trees with $Sh(E_i) = A(t_i)$ for $i = 1, \ldots, n$; then $\forall x A(x) +^{t_1} E_1 +^{t_2} \cdots +^{t_n} E_n$ is a dual expansion tree with $Sh(\forall x A(x) +^{t_1} E_1 +^{t_2} \cdots +^{t_n} E_n) = \forall x A(x)$.

Example 1 Let $P(x)$ be an atom. Then $P(a)$ is a dual expansion tree and $\forall x.P(x) +^a P(a)$ is a dual expansion tree. $\exists x.P(x) +^a P(a)$ is an expansion tree. So

$$\forall x.P(x) +^a P(a) \to \exists x.P(x) +^a P(a)$$

is an expansion tree. But note that

$$\forall x.P(x) +^a P(a) \to \forall x.P(x) +^a P(a)$$

is *not* an expansion tree according to Definition 10 as $\forall x.P(x) +^a P(a)$ is a dual expansion tree but not an expansion tree. Indeed, having strong quantifiers with the type of expansion defined above would be unsound.

The function Dp (deep) maps expansion trees (and dual expansion trees) to quantifier-free formulas, their *full expansion*.

Definition 11 Dp maps a (dual) expansion tree to a formula as follows:

$$\begin{array}{ll}
Dp(E) & = E \text{ for an atomic expansion tree E,} \\
Dp(\neg E) & = \neg Dp(E), \\
Dp(E_1 \circ E_2) & = Dp(E_1) \circ Dp(E_2) \text{ for } \circ \in \{\wedge, \vee, \to\}, \\
Dp(\exists x A +^{t_1} E_1 +^{t_2} \cdots +^{t_n} E_n) & = Dp(E_1) \vee \cdots \vee Dp(E_n), \\
Dp(\forall x A +^{t_1} E_1 +^{t_2} \cdots +^{t_n} E_n) & = Dp(E_1) \wedge \cdots \wedge Dp(E_n).
\end{array}$$

In [23] a notion of expansion proof was defined from expansion trees using the conditions acyclicity and tautology. Acyclicity ensures that there are no cycles between the strong quantifier nodes in the expansion tree. Since our formulas are skolemized and hence do not contain strong quantifiers, we do not need this condition.

Definition 12 (*Expansion proof*) Let ET be an expansion tree of a formula A without strong quantifiers. Then ET is called an expansion proof of A from a set of axioms $\mathcal{A}$ if $Sh(ET) = A$ and $\mathcal{A} \models Dp(ET)$ (where $\models$ is the consequence relation in predicate logic with equality).

Expansion proofs encode a proof of validity of the formula they represent. They can be directly translated into sequent calculus, see [23], and the transformation is based on so-called q-sequents, which we refer to as s-expansion trees (sequent of expansion trees) in this paper.

Definition 13 (*S-expansion tree*) The structure $S\colon\ \Gamma \vdash \Delta$ where $\Delta\colon\ Q_1, \ldots, Q_s$ is a multiset of expansion trees, $\Gamma\colon\ P_1, \ldots, P_r$ is a multiset of dual expansion trees is called an *s-expansion tree*. If $\neg\Gamma \vee \Delta$ (which stands for $\neg P_1 \vee \cdots \vee \neg P_r \vee Q_1 \vee \cdots \vee Q_s$) is an expansion proof then S is called an *s-expansion proof*. This expansion proof is the expansion proof associated with S; the sequent

$$Seq(S)\colon\ Sh(P_1), \ldots, Sh(P_r) \to Sh(Q_1), \ldots, Sh(Q_s)$$

is the sequent associated with S.

It is also possible to read off expansion proofs from sequent calculus proofs. Note that the expansion proof of a proof φ is a sequent of expansion trees, which are defined to be the expansion trees of all formulas in the end-sequent of φ. An algorithm for the extraction of expansion proofs from sequent calculus proofs is presented in [23] and modified algorithms (dealing with cuts and equality) are presented in [21,22]. There exist also algorithms for a transformation of resolution-trees into expansion-trees, see [24].

We will use an algorithm that is briefly described in [21]. In order to show how an expansion proof is extracted from a proof in $\mathbf{LK}_=$, we first need to define an operation on expansion trees. The Merge operator on expansion trees is defined in [23]. Intuitively, two expansion trees T_1 and T_2 can be merged, if $Sh(T_1) = Sh(T_2)$. We give a definition adapted to our concept of expansion tree.

Definition 14 (*Merge*) Let E_1, E_2 be (dual) expansion trees such that $Sh(E_1) = Sh(E_2)$. We define the merge inductively on the complexity of E_1.

- If E_1 is an atom then E_2 is an atom too and $E_1 = E_2$; we define $\text{Merge}_t(E_1, E_2) = E_1$.
- If $E_1 = \neg E_1'$. Then $E_2 = \neg E_2'$ for some E_2'. Let $\text{Merge}_t(E_1', E_2') = E_3'$, then $\text{Merge}_t(E_1, E_2) = \neg E_3'$.
- Let $E_1 = E_{11} \circ E_{12}$ for $\circ \in \{\wedge, \vee, \rightarrow\}$. Then $E_2 = E_{21} \circ E_{22}$ for some E_{21}, E_{22}. Let $E_1' = \text{Merge}_t(E_{11}, E_{21})$ and $E_2' = \text{Merge}_t(E_{21}, E_{22})$. Then $\text{Merge}_t(E_1, E_2) = E_1' \circ E_2'$.
- Let $E_1 = Qx.A(x) +^{t_1} E_{11} + \cdots +^{t_n} E_{1n}$. Then E_2 is of the form $Qx.A(x) +^{s_1} E_{21} + \cdots +^{s_m} E_{2m}$. Then

$$\text{Merge}_t(E_1, E_2) = Qx.A(x) +^{t_1} E_{11} + \cdots +^{t_n} E_{1n} +^{s_1} E_{21} + \cdots +^{s_m} E_{2m}.$$

Example 2 Let $T_1 = \forall x\, Px +^a Pa$ and $T_2 = \forall x\, Px +^b Pb$ be two dual expansion trees then

$$\text{Merge}_t(T_1, T_2) = \forall x\, Px +^a Pa +^b Pb.$$

The definition can be easily extended to more than two expansion trees. Let $T_1, \ldots, T_n$ (for $n \geq 2$) be (dual) expansion trees such that $Sh(T_i) = Sh(T_j)$ for all $i, j \in \{1, \ldots, n\}$. Then we define

$$\text{merge}_t(T_1, T_2) = \text{Merge}_t(T_1, T_2),$$
$$\text{merge}_t(T_1, \ldots, T_n) = \text{Merge}_t(\text{merge}_t(T_1, \ldots, T_{n-1}), T_n) \text{ for } n > 2.$$

It is also possible to merge s-expansion trees. As an s-expansion tree is defined via multisets of expansion trees, some expansion trees might occur more than once either on the left or on the right. In such cases merging s-expansion trees might become ambiguous. To avoid this ambiguity we restrict the merge of s-expansion trees to so-called normalized ones, where the shallow forms occur only once.

Definition 15 (*Normalized sequents*) A sequent $\Gamma \vdash \Delta$ is called *normalized* if the multiplicity of all formulas occurring in Γ (Δ) is one, more precisely: if $\Gamma = A_1, \ldots, A_n$ and $\Delta = B_1, \ldots, B_m$ then $A_i \neq A_j$ for $i \neq j$ ($i, j \in \{1, \ldots, n\}$) and $B_l \neq B_k$ for $l \neq k$ ($l, k \in \{1, \ldots, m\}$). Let S be an s-expansion tree then S is called normalized if $Seq(S)$ is normalized.

Remark 1 Note that every **LK**-proof φ of S and every s-expansion tree can be easily transformed into a proof (s-expansion tree) of a normalized sequent: just apply the rules c_l, c_r

to S. In Sect. 4 we will merge s-expansion trees only if they correspond to end-sequents of proofs. Therefore restricting the merge to normalized s-expansion proofs does not affect the generality of our approach.

In normalized sequents the multisets become sets which allows us to define some set-based operations on sequents:

Definition 16 Let S_1, S_2 be two normalized sequents such that $S_1 = \Gamma_1 \vdash \Delta_1$, $S_2 = \Gamma_2 \vdash \Delta_2$. Let $\Gamma = \Gamma_1 \cap \Gamma_2$, $\Delta = \Delta_1 \cap \Delta_2$. We define the following operations on S_1, S_2:

$$S_1 \cap S_2 = \Gamma \vdash \Delta,\ S_1 \setminus S_2 = (\Gamma_1 \setminus \Gamma) \vdash (\Delta_1 \setminus \Delta),\ S_2 \setminus S_1 = (\Gamma_2 \setminus \Gamma) \vdash (\Delta_2 \setminus \Delta).$$

The sequents S_1 and S_2 are called *disjoint* if $S_1 \cap S_2 = \vdash$.Then, obviously, $S_1 \cap S_2$, $S_1 \setminus S_2$ and $S_2 \setminus S_1$ are pairwise disjoint and

$$S_1 = (S_1 \cap S_2) \circ (S_1 \setminus S_2),$$
$$S_2 = (S_1 \cap S_2) \circ (S_2 \setminus S_1).$$

Now we are ready to define the merging of normalized s-expansion trees.

Definition 17 (*Merge of s-expansion trees*) Let S_1 and S_2 be two normalized s-expansion trees and $S_1^* = Seq(S_1)$, $S_2^* = Seq(S_2)$. Then, by definition, S_1^*, S_2^* are normalized sequents. We define $\Gamma^* \vdash \Delta^* = S_1^* \cap S_2^*$, $\Pi_1^* \vdash \Lambda_1^* = S_1^* \setminus S_2^*$, $\Pi_2^* \vdash \Lambda_2^* = S_2^* \setminus S_1^*$. Then

$$S_1^* = (\Gamma^* \vdash \Delta^*) \circ (\Pi_1^* \vdash \Lambda_1^*),$$
$$S_2^* = (\Gamma^* \vdash \Delta^*) \circ (\Pi_2^* \vdash \Lambda_2^*).$$

Then there exist s-expansion trees $\Gamma \vdash \Delta$, $\Gamma' \vdash \Delta'$, $\Pi_1 \vdash \Lambda_1$ and $\Pi_2 \vdash \Lambda_2$ such that

$$S_1 = (\Gamma \vdash \Delta) \circ (\Pi_1 \vdash \Lambda_1),$$
$$S_2 = (\Gamma' \vdash \Delta') \circ (\Pi_2 \vdash \Lambda_2),$$

where $Seq(\Gamma \vdash \Delta) = Seq(\Gamma' \vdash \Delta') = \Gamma^* \vdash \Delta^*$, $Seq(\Pi_1 \vdash \Lambda_1) = \Pi_1^* \vdash \Lambda_1^*$ and $Seq(\Pi_2 \vdash \Lambda_2) = \Pi_2^* \vdash \Lambda_2^*$. Note that the concatenation $\circ$ of sequents can be directly extended to s-expansion trees.

Then there exist bijective mappings $\pi_l \colon \Gamma \to \Gamma'$ and $\pi_r \colon \Delta \to \Delta'$ with $\pi_l(T) = T'$ iff $Sh(T) = Sh(T')$ (the same for π_r). So assume

$$\Gamma \vdash \Delta = T_1, \ldots, T_n \vdash T_{n+1}, \ldots, T_{n+m} \text{ and therefore}$$
$$\Gamma' \vdash \Delta' = \pi_l(T_1), \ldots, \pi_l(T_n) \vdash \pi_r(T_{n+1}), \ldots, \pi_r(T_{n+m}).$$

Now let $T_i^* = \text{merge}_t(T_i, \pi_l(T_i))$ for $i = 1, \ldots, n$ and $T_i^* = \text{merge}_t(T_i, \pi_r(T_i))$ for $i = n+1, \ldots, n+m$. Then we define

$$\text{Merge}_s(S_1, S_2) = (T_1^*, \ldots, T_n^* \vdash T_{n+1}^*, \ldots, T_{n+m}^*) \circ (\Pi_1 \vdash \Lambda_1) \circ (\Pi_2 \vdash \Lambda_2).$$

Note that, by construction, $\text{Merge}_s(S_1, S_2)$ is a normalized s-expansion tree.

We extend the merging of s-sequents to more than two as follows. Let $n \geq 2$ and $S_1, \ldots, S_n$ be normalized s-expansion trees. Then

$$\text{merge}_s(S_1, S_2) = \text{Merge}_s(S_1, S_2) \text{ for } n = 2,$$
$$\text{merge}_s(S_1, \ldots, S_n) = \text{Merge}_s(\text{merge}_s(S_1, \ldots, S_{n-1}), S_n) \text{ for } n > 2.$$

The s-expansion tree $\text{merge}_s(S_1, \ldots, S_n)$ is also normal which can be verified by an obvious inductive argument.

Frequently we will write $\text{merge}_s\{S_i \mid i = 1, \ldots, n\}$ for $\text{merge}_s(S_1, \ldots, S_n)$. If no confusion arises we will frequently write merge instead of merge_t and merge_s.

Example 3 Let $S_1 = \forall x Px +^a Pa \vdash$, $S_2 = \forall x Px +^b Pb \vdash Qa$ and $S_3 = \vdash \exists y Qy$ be s-expansion trees. Then

$$\text{merge}_s(S_1, S_2, S_3) = \forall x Px +^a Pa +^b Pb \vdash Qa, \exists y Qy.$$

The extraction of expansion proofs from **LK**-proofs requires quantifier-free cuts. Due to the structure of the CERES-method (which will be used for a efficient method of extracting expansion proofs) we consider proofs with only atomic cuts.

Definition 18 A proof φ in $\mathbf{LK}_=$ is in the subclass $\mathbf{LK}_0$ if

1. φ does not contain strong quantifier inferences.
2. All cuts in φ are atomic.
3. Equality rules are only applied to atoms.
4. The axiom set contains Ax.

Definition 19 (*Extraction of s-expansion trees from proofs in* $\mathbf{LK}_0$) We define a transformation ET which maps proofs in $\mathbf{LK}_0$ to s-expansion trees. We define the transformation inductively (on the number of inferences in the proof) but the rules for $\neg_l, \neg_r, \vee_l$, $\vee_{r_1}, \vee_{r_2}, =_{l2}, =_{r2}$ are omitted, the transformation of the these rules being obvious.

base case: φ is an axiom. Then φ is of the form $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ for atoms A_i, B_j and so $\text{ET}(\varphi) = \varphi$.
If $\varphi =$

$$\dfrac{\begin{matrix}(\pi)\\ A, B, \Gamma \vdash \Delta\end{matrix}}{A \wedge B, \Gamma \vdash \Delta} \wedge_l$$

and $\text{ET}(\pi) = A^*, B^*, \Gamma^* \vdash \Delta^*$, then $\text{ET}(\varphi) = A^* \wedge B^*, \Gamma^* \vdash \Delta^*$.
If $\varphi =$

$$\dfrac{\begin{matrix}(\pi_1) & (\pi_2)\\ \Gamma_1 \vdash \Delta_1, A & \Gamma_2 \vdash \Delta_2, B\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A \wedge B} \wedge_r$$

and $\text{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, A^*$ and $\text{ET}(\pi_2) = \Gamma_2^* \vdash \Delta_2^*, B^*$, then $\text{ET}(\varphi) = \Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*, A^* \wedge B^*$.
If $\varphi =$

$$\dfrac{\begin{matrix}(\pi_1) & (\pi_1)\\ \Gamma_1 \vdash \Delta_1, A & B, \Gamma_2 \vdash \Delta_2\end{matrix}}{A \to B, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2} \to_l$$

and $\text{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, A^*$ and $\text{ET}(\pi_2) = B^*, \Gamma_2^* \vdash \Delta_2^*$, then $\text{ET}(\varphi) = A^* \to B^*, \Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*$.
If $\varphi =$

$$\dfrac{\begin{matrix}(\pi)\\ A, \Gamma \vdash \Delta, B\end{matrix}}{\Gamma \vdash \Delta, A \to B} \to_r$$

and $\text{ET}(\pi) = A^*, \Gamma^* \vdash \Delta^*, B^*$, then $\text{ET}(\varphi) = \Gamma^* \vdash \Delta^*, A^* \to B^*$.

If $\varphi =$

$$\dfrac{\begin{array}{c}(\pi)\\ A\{x \leftarrow t\}, \Gamma \vdash \Delta\end{array}}{(\forall x)A(x), \Gamma \vdash \Delta}\ \forall_l$$

and $\mathrm{ET}(\pi) = A\{x \leftarrow t\}^*, \Gamma^* \vdash \Delta^*$, then $\mathrm{ET}(\varphi) = (\forall x)A(x) +^t A\{x \leftarrow t\}^*, \Gamma^* \vdash \Delta^*$.
If $\varphi =$

$$\dfrac{\begin{array}{c}(\pi)\\ \Gamma \vdash \Delta, A\{x \leftarrow t\}\end{array}}{\Gamma \vdash \Delta, (\exists x)A(x)}\ \exists_r$$

and $\mathrm{ET}(\pi) = \Gamma^* \vdash \Delta^*, A\{x \leftarrow t\}^*$, then $\mathrm{ET}(\varphi) = \Gamma^* \vdash \Delta^*, (\exists x)A(x) +^t A\{x \leftarrow t\}^*$.
If $\varphi =$

$$\dfrac{\begin{array}{c}(\pi)\\ \Gamma \vdash \Delta\end{array}}{A, \Gamma \vdash \Delta}\ w_l$$

and $\mathrm{ET}(\pi) = \Gamma^* \vdash \Delta^*$, then $\mathrm{ET}(\varphi) = A, \Gamma^* \vdash \Delta^*$. Similarly for w_r.
If $\varphi =$

$$\dfrac{\begin{array}{c}(\pi)\\ A, A, \Gamma \vdash \Delta\end{array}}{A, \Gamma \vdash \Delta}\ c_l$$

and $\mathrm{ET}(\pi) = A_1^*, A_2^*, \Gamma^* \vdash \Delta^*$ then $\mathrm{ET}(\varphi) = \mathrm{merge}(A_1^*, A_2^*), \Gamma^* \vdash \Delta^*$. Similarly for c_r.
Note that, for the rules below, the auxiliary formulas of the rules are atomic.
If $\varphi =$

$$\dfrac{\begin{array}{cc}(\pi_1) & (\pi_2)\\ \Gamma_1 \vdash \Delta_1, A^m & A^n, \Gamma_2 \vdash \Delta_2\end{array}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ cut(A, m, n)$$

where $\mathrm{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, A^m$ and $\mathrm{ET}(\pi_1) = A^n, \Gamma_2^* \vdash \Delta_2^*$; then $\mathrm{ET}(\varphi) = \Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*$.
If $\varphi =$

$$\dfrac{\begin{array}{cc}(\pi_1) & (\pi_2)\\ \Gamma_1 \vdash \Delta_1, s = t & A[s]_\Lambda, \Gamma_2 \vdash \Delta_2\end{array}}{A[t]_\Lambda, \Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ =_{l1}$$

and $\mathrm{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, s = t$ and $\mathrm{ET}(\pi_2) = A[s]_\Lambda, \Gamma_2^* \vdash \Delta_2^*$, then $\mathrm{ET}(\varphi) = \Gamma_1^*, \Gamma_2^*, A[t]_\Lambda \vdash \Delta_1^*, \Delta_2^*$.
If $\varphi =$

$$\dfrac{\begin{array}{cc}(\pi_1) & (\pi_2)\\ \Gamma_1 \vdash \Delta_1, s = t & \Gamma_2 \vdash \Delta_2, A[s]_\Lambda\end{array}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A[t]_\Lambda}\ =_{r1}$$

and $\mathrm{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, s = t$ and $\mathrm{ET}(\pi_1) = \Gamma_2^* \vdash \Delta_2^*, A[s]_\Lambda$, then $\mathrm{ET}(\varphi) = \Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*, A[t]_\Lambda$.

Proposition 1 *The transformation* ET *is sound: if φ is a proof in* $\mathbf{LK}_0$ *then* $\mathrm{ET}(\varphi)$ *is an s-expansion proof.*

Proof We proceed by induction on the number of inferences in φ. We consider the cases of axioms (represented by an axiom set $\mathcal{A}$), $\wedge_r$, *cut* and $=_{r_1}$; the other cases are analogous.

- (axiom) Let S be an axiom sequent in $\mathcal{A}$. Then $S = A_1, \ldots, A_n \vdash B_1, \ldots, B_m$ for atoms A_i, B_j. Therefore for F_S: $\neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots B_m$ we have $Sh(F_S) = Dp(F_S) = F_S$ and $\mathcal{A} \models Dp(F_S)$.
- $(\wedge_r)$ $\varphi =$

$$\dfrac{\begin{matrix}(\pi_1) & (\pi_2)\\ \Gamma_1 \vdash \Delta_1, A & \Gamma_2 \vdash \Delta_2, B\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A \wedge B}\ \wedge_r$$

  and $\mathrm{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, A^*$ and $\mathrm{ET}(\pi_2) = \Gamma_2^* \vdash \Delta_2^*, B^*$ are s-expansion-proofs. Therefore, $\neg\Gamma_1^* \vee \Delta_1^* \vee A^*$ and $\neg\Gamma_2^* \vee \Delta_2^* \vee B^*$ are expansion proofs and $\mathcal{A} \models Dp(\neg\Gamma_1^* \vee \Delta_1^* \vee A^*)$ and $\mathcal{A} \models Dp(\neg\Gamma_2^* \vee \Delta_2^* \vee B^*)$. But then $\mathcal{A} \models Dp(\neg\Gamma_1^* \vee \Delta_1^* \vee \neg\Gamma_2^* \vee \Delta_2^* \vee (A^* \wedge B^*))$ and $\neg\Gamma_1^* \vee \Delta_1^* \vee \neg\Gamma_2^* \vee \Delta_2^* \vee (A^* \wedge B^*)$ is an expansion proof. Therefore $\Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*, A^* \wedge B^*$ $(= \mathrm{ET}(\varphi))$ is an s-expansion-proof.
- (*cut*) $\varphi =$

$$\dfrac{\begin{matrix}(\pi_1) & (\pi_2)\\ \Gamma_1 \vdash \Delta_1, A^m & A^n, \Gamma_2 \vdash \Delta_2\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2}\ cut(A, m, n)$$

  where $\mathrm{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, A^m$ and $\mathrm{ET}(\pi_1) = A^n, \Gamma_2^* \vdash \Delta_2^*$ are s-expansion-proofs. Therefore, $\neg\Gamma_1^* \vee \Delta_1^* \vee A^m$ and $\neg A^n \vee \neg\Gamma_2^* \vee \Delta_2^*$ are expansion proofs and $\mathcal{A} \models Dp(\neg\Gamma_1^* \vee \Delta_1^* \vee A^m)$ and $\mathcal{A} \models Dp(\neg A^n \vee \neg\Gamma_2^* \vee \Delta_2^*)$. But then $\mathcal{A} \models Dp(\neg\Gamma_1^* \vee \Delta_1^* \vee \neg\Gamma_2^* \vee \Delta_2^*)$ and $\neg\Gamma_1^* \vee \Delta_1^* \vee \neg\Gamma_2^* \vee \Delta_2^*$ is an expansion proof. Therefore $\Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*$ $(= \mathrm{ET}(\varphi))$ is an s-expansion-proof.
- $(=_{r_1})$ $\varphi =$

$$\dfrac{\begin{matrix}(\pi_1) & (\pi_2)\\ \Gamma_1 \vdash \Delta_1, s = t & \Gamma_2 \vdash \Delta_2, A[s]_\Lambda\end{matrix}}{\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2, A[t]_\Lambda}\ =_{r1}$$

  and $\mathrm{ET}(\pi_1) = \Gamma_1^* \vdash \Delta_1^*, s = t$ and $\mathrm{ET}(\pi_1) = \Gamma_2^* \vdash \Delta_2^*, A[s]_\Lambda$ are s-expansion proofs. Therefore $\neg\Gamma_1^* \vee \Delta_1^* \vee s = t$ and $\neg\Gamma_2^* \vee \Delta_2^* \vee A[s]_\Lambda$ are expansion proofs and hence $\mathcal{A} \models Dp(\neg\Gamma_1^* \vee \Delta_1^* \vee s = t)$ and $\mathcal{A} \models Dp(\neg\Gamma_2^* \vee \Delta_2^* \vee A[s]_\Lambda)$. But then $\mathcal{A} \models Dp(\neg\Gamma_1^* \vee \Delta_1^* \vee \neg\Gamma_2^* \vee \Delta_2^* \vee A[t]_\Lambda)$ and $\neg\Gamma_1^* \vee \Delta_1^* \vee \neg\Gamma_2^* \vee \Delta_2^* \vee A[t]_\Lambda)$ is an expansion proof. Therefore, $\Gamma_1^*, \Gamma_2^* \vdash \Delta_1^*, \Delta_2^*, A[t]_\Lambda$ $(= \mathrm{ET}(\varphi))$ is an s-expansion-proof. □

In case of a prenex end-sequent S an expansion proof corresponds to the derivation of S from the mid-sequent (Herbrand sequent). The essence of Herbrand's theorem [18] consists of the replacement of quantified formulas by instances of these formulas. This results in a quantifier-free formula which is validity-equivalent to the original formula. Of course, the function *Dp* of an expansion proof corresponds to a Herbrand sequent. But if we consider proofs of prenex end-sequents we can extract Herbrand sequents directly, instead of extracting expansion proofs and computing their deep functions. The method for Herbrand sequent extraction in the prenex case is based on collecting instances, and is described in [6,20].

To illustrate construction of an s-expansion proof from an $\mathbf{LK}_0$-proof , consider the following simple example.

Example 4 We work with $Ax \cup \{\vdash a = b\}$. Let φ be the proof of $S = P(a) \wedge Q(a, f(a)) \vdash \exists x(P(x) \wedge \exists y.Q(x, y))$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash a = b \quad Q(a, f(a)) \vdash Q(a, f(a))}{Q(a, f(a)) \vdash Q(a, f(b))}\,=_{r_1}}{Q(a, f(a)) \vdash \exists y.Q(a, y)}\,\exists_r \qquad P(a) \vdash P(a)}{P(a), Q(a, f(a)) \vdash P(a) \wedge \exists y.Q(a, y)}\,\wedge_r}{P(a) \wedge Q(a, f(a)) \vdash P(a) \wedge \exists y.Q(a, y)}\,\wedge_l}{P(a) \wedge Q(a, f(a)) \vdash \exists x(P(x) \wedge \exists y.Q(x, y))}\,\exists_r$$

Note that S is not in prenex form. Therefore, extracting the Herbrand sequent by collecting instances is not possible. Instead we compute the expansion proof $\mathrm{ET}(\varphi)$.

Below we compute the s-expansion proof corresponding to φ. First we compute expansion trees for all formulas F in S and call them $\mathrm{ET}(F)$.

$$\begin{aligned}
\mathrm{ET}(P(a) \wedge Q(a, f(a))) &= P(a) \wedge Q(a, f(a)) \\
\mathrm{ET}(\exists x(P(x) \wedge \exists y.Q(x, y))) &= \exists x(P(x) \wedge \exists y.Q(x, y)) \\
&\quad +^{a} (P(a) \wedge (\exists y Q(a, y) +^{f(b)} Q(a, f(b))))
\end{aligned}$$

The s-expansion proof $\mathrm{ET}(\varphi)$ associated with the end-sequent S is:

$$\mathrm{ET}(P(a) \wedge Q(a, f(a))) \vdash \mathrm{ET}(\exists x(P(x) \wedge \exists y.Q(x, y)))$$

The corresponding expansion proof associated with $\mathrm{ET}(\varphi)$ is:

$$\neg\mathrm{ET}(P(a) \wedge Q(a, f(a))) \vee \mathrm{ET}(\exists x(P(x) \wedge \exists y.Q(x, y))).$$

To obtain the tautologous formula (corresponding to the Herbrand sequent) we construct the deep function for the expansion proof; we compute $Dp(\mathrm{ET}(S_i))$:

$$Dp(\mathrm{ET}(P(a) \wedge Q(a, f(a)))) = Dp(P(a)) \wedge Dp(Q(a, f(a))) = P(a) \wedge Q(a, f(a))$$

$$\begin{aligned}
&Dp(\mathrm{ET}(\exists x(P(x) \wedge \exists y.Q(x, y)))) &&= \\
&Dp(\exists x(P(x) \wedge \exists y.Q(x, y)) +^{a} (P(a) \wedge (\exists y Q(a, y) +^{f(b)} Q(a, f(b))))) &&= \\
&Dp(P(a) \wedge (\exists y Q(a, y) +^{f(b)} Q(a, f(b)))) &&= \\
&Dp(P(a)) \wedge Dp(\exists y Q(a, y) +^{f(b)} Q(a, f(b))) &&= \\
&P(a) \wedge Q(a, f(b))
\end{aligned}$$

Hence, combining those deep functions we get $P(a) \wedge Q(a, f(a)) \vdash P(a) \wedge Q(a, f(b))$. Note that this sequent is valid in $Ax \cup \{\vdash a = b\}$ (though it is not tautological).

3 The Method CERES

The method CERES [4,5], is a cut-elimination method that is based on resolution. It differs from the reductive stepwise methods a la Gentzen [15] by analyzing the whole proof in a preprocessing step and extracting a formula in clausal form which forms the kernel of the cut-elimination method.

CERES in predicate logic with equality roughly works as follows: The structure of a proof φ containing cuts is encoded in an unsatisfiable set of clauses $\mathrm{CL}(\varphi)$ (the *characteristic clause set* of φ). A refutation of $\mathrm{CL}(\varphi)$ by resolution and paramodulation (abbreviated as PR-refutation) then serves as a skeleton for an *atomic cut normal form*, a new proof which contains at most atomic cuts. The corresponding proof theoretic transformation uses so-called proof projections $\varphi[C]$ for $C \in \mathrm{CL}(\varphi)$, which are simple cut-free proofs extracted

from φ (proving the end-sequent S extended by the atomic sequent C). In [5] it was shown that CERES outperforms reductive methods of cut-elimination (a la Gentzen or Tait) in computational complexity: there are infinite sequences of proofs where the computing time of CERES is nonelementarily faster than that of the reductive methods; on the other hand a nonelementary speed-up of CERES via reductive methods is shown impossible.

In this section we describe the original CERES method which was designed for classical logic. Given an $\mathbf{LK}_=$-proof φ of a skolemized sequent $\Gamma \vdash \Delta$, the main steps of (classical) CERES are:

1. Extraction of the characteristic clause set $\mathrm{CL}(\varphi)$.
2. Construction of a PR-refutation (see Definition 8) of $\mathrm{CL}(\varphi)$.
3. Extraction of a set of projections $\pi(C)$ for every $C \in \mathrm{CL}(\varphi)$.
4. Merging of refutation and projections into a proof φ^* (a CERES-normal form) with only atomic cuts.

We will use the following proof as our running example to clarify the definitions below.

Example 5 The set of axioms Ax_s is defined as $Ax_s = Ax \cup \{\vdash f^2z = gz\}$.

Let φ be a proof of the sequent $Pa, \forall x(Px \to Pfx) \vdash \exists z Pf^4z$.

$$\dfrac{\dfrac{}{\forall x(Px \to Pfx) \vdash \forall \mathbf{x}(\mathbf{Px} \to \mathbf{Pgx})}(\varphi_1) \qquad \dfrac{}{Pc, \forall \mathbf{x}(\mathbf{Px} \to \mathbf{Pgx}) \vdash Pg^2c}(\varphi_2)}{Pc, \forall x(Px \to Pfx) \vdash Pg^2c}\, cut$$

φ_1 is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{Pz} \vdash Pz} \quad \dfrac{\dfrac{}{Pfz \vdash Pfz} \quad \dfrac{}{Pf^2z \vdash \mathbf{Pf^2z}} \quad \dfrac{}{\vdash \mathbf{f^2z = gz}} =_{r_1}}{Pf^2z \vdash \mathbf{Pgz}} \to_l}{Pfz, Pfz \to Pf^2z \vdash \mathbf{Pgz}} \to_l}{\mathbf{Pz}, Pz \to Pfz, Pfz \to Pf^2z \vdash \mathbf{Pgz}}\, 2 \times \forall_l + c_l}{\mathbf{Pz}, \forall x(Px \to Pfx) \vdash \mathbf{Pgz}} \to_r}{\forall x(Px \to Pfx) \vdash \mathbf{Pz} \to \mathbf{Pgz}} \to_r}{\forall x(Px \to Pfx) \vdash \forall \mathbf{x}(\mathbf{Px} \to \mathbf{Pgx})} \forall_r$$

φ_2 is

$$\dfrac{\dfrac{\dfrac{}{Pc \vdash \mathbf{Pc}} \quad \dfrac{\dfrac{}{\mathbf{Pgc} \vdash \mathbf{Pgc}} \quad \dfrac{}{\mathbf{Pg^2c} \vdash Pg^2c} \to_l}{\mathbf{Pgc}, \mathbf{Pgc} \to \mathbf{Pg^2c} \vdash Pg^2c} \to_l}{Pc, \mathbf{Pc} \to \mathbf{Pgc}, \mathbf{Pgc} \to \mathbf{Pg^2c} \vdash Pg^2c}\, 2 \times \forall_l + c_l}{Pc, \forall \mathbf{x}(\mathbf{Px} \to \mathbf{Pgx}) \vdash Pg^2c}$$

The ancestors of the cut formulas are indicated in bold face.

Intuitively, the clause set extraction consists in collecting all atomic ancestors of the cuts which occur in the axioms of the proof. The clauses are formed depending on how these atoms are related via binary inferences in the proof.

Definition 20 (*Characteristic clause-set*) Let φ be a proof of a skolemized sequent. The characteristic clause set is built recursively from the leaves of the proof to the end-sequent. Let ν be the occurrence of a sequent in this proof. Then:

- If ν is an axiom, then $\mathrm{CL}(\nu)$ contains the sub-sequent of ν composed only of cut-ancestors.
- If ν is the result of the application of a unary rule on a sequent μ, then $\mathrm{CL}(\nu) = \mathrm{CL}(\mu)$
- If ν is the result of the application of a binary rule on sequents μ_1 and μ_2, then we distinguish two cases:
  - If the rule is applied to ancestors of the cut formula, then $\mathrm{CL}(\nu) = \mathrm{CL}(\mu_1) \cup \mathrm{CL}(\mu_2)$
  - If the rule is applied to ancestors of the end-sequent, then $\mathrm{CL}(\nu) = \mathrm{CL}(\mu_1) \times \mathrm{CL}(\mu_2)$

  where

$$\mathrm{CL}(\mu_1) \times \mathrm{CL}(\mu_2) = \{C \circ D \mid C \in \mathrm{CL}(\mu_1), D \in \mathrm{CL}(\mu_2)\}.$$

  Note that $\circ$ represents the merging of sequents.

If ν_0 is the root node $\mathrm{CL}(\nu_0)$ is called the characteristic clause set of φ.

Example 6 The characteristic clause set of our proof φ from Example 5 is constructed as follows:

We consider the following cut-ancestors (in axioms) in φ_1

$$\{Pz \vdash\}; \{\vdash Pf^2z\}; \{\vdash f^2z = gz\}$$

$=_{r_1}$ operates on cut-ancestors, therefore we get

$$S_1 = \{\vdash Pf^2z\} \cup \{\vdash f^2z = gz\}$$

$\rightarrow_l$ operates on end-sequent ancestors, hence

$$S = \{Pz \vdash\} \times S_1 = \{Pz \vdash Pf^2z; Pz \vdash f^2z = gz\}$$

We proceed analogously for the cut-ancestors in φ_2 and obtain

$$S' = \{\vdash Pc; Pgc \vdash Pgc; Pg^2c \vdash\}$$

The characteristic clause set is $S \cup S'$

$$CL(\varphi) = \{Pz \vdash Pf^2z;\ Pz \vdash f^2z = gz;\ \vdash Pc;\ Pgc \vdash Pgc;\ Pg^2c \vdash\}.$$

The next step is to obtain a resolution refutation of $CL(\varphi)$. It is thus important to show that this set is always refutable.

Theorem 1 *Let φ be a proof of a skolemized end-sequent. Then the characteristic clause set* $\mathrm{CL}(\varphi)$ *is refutable by resolution and paramodulation.*

Proof In [2,6]. □

Example 7 We give a PR-refutation γ of $\mathrm{CL}(\varphi)$ for φ in Example 5:

$$\dfrac{\dfrac{\dfrac{\dfrac{(\pi)}{\vdash Pgc} \quad Pz \vdash f^2z = gz}{\vdash f^2gc = g^2c}\,R \quad \dfrac{\dfrac{(\pi)}{\vdash Pgc} \quad Pz \vdash Pf^2z}{\vdash Pf^2gc}\,R}{\vdash Pg^2c}\,para \quad Pg^2c \vdash}{\vdash}\,R$$

where π is

$$\dfrac{\dfrac{\vdash Pc \quad Pz \vdash Pf^2z}{\vdash Pf^2c}\,R \quad \dfrac{\vdash Pc \quad Pz \vdash f^2z = gz}{\vdash f^2c = gc}\,R}{\vdash Pgc}\,para$$

Definition 21 For a characteristic clause set $\mathrm{CL}(\varphi)$ and a ground refutation R of of a PR refutation ρ of $\mathrm{CL}(\varphi)$ we define

$$\mathrm{CL}(\varphi, R) = \{C_i \mid C_i \in \mathrm{CL}(\varphi) \text{ and } C_i \text{ occurs in R}\}.$$

Each clause in the clause set will have a projection associated with it. A projection of a clause C is a derivation built from φ by taking the axioms in which the atoms of C occur and all the inferences that operate on end-sequent ancestors. As a result, the end-sequent of a projection will be the end-sequent of φ extended by the atoms of C.

Definition 22 (*Projections*) Let φ be a proof of a skolemized end-sequent $\Gamma \vdash \Delta$ in $\mathbf{LK}_{=}$. For nodes ν in φ we define inductively the set of cut-free proofs $p(\nu)$. If ν_0 is the root node and $\psi \in p(\nu_0)$ we call ψ a *projection*. Let ν be a node in φ such that $S(\nu) = \Gamma \vdash \Delta$; then $\Gamma \vdash \Delta = \Gamma_c, \Gamma_e \vdash \Delta_e, \Delta_c$ where $\Gamma_c \vdash \Delta_c$ consists of cut-ancestors and $\Gamma_e \vdash \Delta_e$ of ancestors of the end-sequent.

(a) ν is a leaf in φ. Then the sequent at ν is an axiom and we define $p(\nu) = \{\nu\}$. The clause part of ν is the subsequent $\mathrm{CL}(\nu)$.

(b) ν is the conclusion of a unary rule ξ with premise μ.

(b1) The principal formula of ξ is an ancestor of a cut. Then $\varphi.\nu$ is of the form

$$\dfrac{\overset{(\varphi.\mu)}{\Gamma_c, \Gamma_e \vdash \Delta_c, \Delta_e}}{\Gamma'_c, \Gamma_e \vdash \Delta'_c, \Delta_e}\ \xi$$

We define $p(\nu) = p(\mu)$.

(b2) The principal formula of ξ is an ancestor of the end-sequent. Then $\varphi.\nu$ is of the form

$$\dfrac{\overset{(\varphi.\mu)}{\Gamma_c, \Gamma_e \vdash \Delta_c, \Delta_e}}{\Gamma_c, \Gamma'_e \vdash \Delta_c, \Delta'_e}\ \xi$$

Let $\psi \in p(\mu)$ be a proof of $C, \Gamma_e \vdash \Delta_e, D$ where $C \vdash D$ is the clause part of ψ. Then $\psi' \in p(\nu)$ for $\psi' =$

$$\dfrac{\overset{(\psi)}{C, \Gamma_e \vdash \Delta_e, D}}{C, \Gamma'_e \vdash \Delta'_e, D}\ \xi$$

and $C \vdash D$ is the clause part of ψ'.

(c) ν is the conclusion of a binary rule ξ with premises μ_1, μ_2.

(c1) The auxiliary formulas of ξ are ancestors of a cut. Then $\varphi.\nu$ is of the form

$$\dfrac{\overset{(\varphi.\mu_1)}{\Gamma_c, \Gamma_e \vdash \Delta_c, \Delta_e} \quad \overset{(\varphi.\mu_2)}{\Pi_c, \Pi_e \vdash \Lambda_c, \Lambda_e}}{\Gamma'_c, \Pi'_c, \Gamma_e, \Pi_e \vdash \Delta'_c, \Lambda'_c, \Delta_e, \Lambda_e}\ \xi$$

Let $\psi \in p(\mu_1)$ such that ψ is a proof of $C, \Gamma_e \vdash \Delta_e, D$ where $C \vdash D$ is the clause part of ψ. Then $\psi' \in p(\nu)$ for $\psi' =$

$$\dfrac{\overset{(\psi)}{C, \Gamma_e \vdash \Delta_e, D}}{C, \Gamma_e, \Pi_e \vdash \Delta_e, \Lambda_e, D}\ w^*$$

and $C \vdash D$ is the clause part of ψ'.
Let $\psi \in p(\mu_2)$ such that ψ is a proof of $E, \Pi_e \vdash \Lambda_e, F$ where $E \vdash F$ is the clause part of ψ. Then $\psi' \in p(\nu)$ for $\psi' =$

$$\dfrac{\begin{matrix}(\psi)\\ E, \Pi_e \vdash \Lambda_e, F\end{matrix}}{E, \Gamma_e, \Pi_e \vdash \Delta_e, \Lambda_e, F}\ w^*$$

and $E \vdash F$ is the clause part of ψ'.

(c2) The auxiliary formulas of ξ are ancestors of the end-sequent. Then $\varphi.\nu$ is of the form

$$\dfrac{\begin{matrix}(\varphi.\mu_1) & (\varphi.\mu_2)\\ \Gamma_c, \Gamma_e \vdash \Delta_c, \Delta_e & \Pi_c, \Pi_e \vdash \Lambda_c, \Lambda_e\end{matrix}}{\Gamma_c, \Pi_c, \Gamma'_e, \Pi'_e \vdash \Delta_c, \Lambda_c, \Delta'_e, \Lambda'_e}\ \xi$$

Let $\psi_1 \in p(\mu_1)$ such that ψ_1 is a proof of $C, \Gamma_e \vdash \Delta_e, D$ and $C \vdash D$ is the clause part of ψ_1; likewise let $\psi_2 \in p(\mu_2)$ such that ψ_2 is a proof of $E, \Pi_e \vdash \Lambda_e, F$ and $E \vdash F$ is the clause part of ψ_2. Then $\psi \in p(\nu)$ for $\psi =$

$$\dfrac{\begin{matrix}(\psi_1) & (\psi_2)\\ C, \Gamma_e \vdash \Delta_e, D & E, \Pi_e \vdash \Lambda_e, F\end{matrix}}{C, E, \Gamma'_e, \Pi'_e \vdash \Delta'_e, \Lambda'_e, D, F}\ \xi$$

and the clause part of ψ is $C, E \vdash D, F$.

Example 8 We define the projections of φ from Example 5 to the clauses $Pz \vdash Pf^2z$, $Pz \vdash f^2z = gz$, $\vdash Pc$ and $Pg^2c \vdash$. $\varphi[Pz \vdash Pf^2z]$:

$$\dfrac{\dfrac{\dfrac{\dfrac{Pfz \vdash Pfz \qquad Pf^2z \vdash \mathbf{Pf^2z}}{Pfz, Pfz \to Pf^2z \vdash \mathbf{Pf^2z}}\to_l}{\mathbf{Pz} \vdash Pz \qquad Pfz, Pfz \to Pf^2z \vdash \mathbf{Pf^2z}}\to_l}{\mathbf{Pz}, Pz \to Pfz, Pfz \to Pf^2z \vdash \mathbf{Pf^2z}}\ 2 \times \forall_l + c_l}{\mathbf{Pz}, \forall x(Px \to Pfx) \vdash \mathbf{Pf^2z}}\ w : l + w : r}{\mathbf{Pz}, Pc, \forall x(Px \to Pfx) \vdash Pg^2c, \mathbf{Pf^2z}}$$

$\varphi[Pz \vdash f^2z = gz]$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\vdash \mathbf{f^2z = gz}}{Pfz \vdash Pfz \qquad Pf^2z \vdash \mathbf{f^2z = gz}}\ w : l}{\mathbf{Pz} \vdash Pz \qquad Pfz, Pfz \to Pf^2z \vdash \mathbf{f^2z = gz}}\to_l}{\mathbf{Pz}, Pz \to Pfz, Pfz \to Pf^2z \vdash \mathbf{f^2z = gz}}\to_l}{\mathbf{Pz}, \forall x(Px \to Pfx) \vdash \mathbf{f^2z = gz}}\ 2 \times \forall_l + c_l}{\mathbf{Pz}, Pc, \forall x(Px \to Pfx) \vdash Pg^2c, \mathbf{f^2z = gz}}\ w : l + w : r$$

$\varphi[\vdash Pc]$:

$$\dfrac{Pc \vdash \mathbf{Pc}}{Pc, \forall x(Px \to Pfx) \vdash Pg^2c, \mathbf{Pc}}\ w_l + w : r$$

$\varphi[Pg^2c \vdash]$:

$$\dfrac{\mathbf{Pg^2c} \vdash Pg^2c}{\mathbf{Pg^2c}, Pc, \forall x(Px \to Pfx) \vdash Pg^2c}\, w_l + w_r$$

Given the projections and a grounded PR refutation, it is possible to build a proof $\hat{\varphi}$ of $\Gamma \vdash \Delta$ with only atomic cuts.

If we apply all most general unifiers in the PR proof γ we obtain a proof in $\mathbf{LK}_=$ (in fact only contractions, cut and paramodulation remain). If $\gamma\sigma$ is such a proof and we apply a substitution replacing all variables by a constant symbol we obtain a ground PR refutation. Note that after applying the most general unifiers to γ we obtain a derivation in $\mathbf{LK}_=$ where the resolution rule becomes a cut rule. For a formal definition see [6].

Example 9 The ground PR-refutation γ' is

$$\dfrac{\dfrac{\dfrac{\overset{(\pi)}{\vdash Pgc} \qquad Pgc \vdash f^2gc = g^2c}{\vdash f^2gc = g^2c}\, cut \qquad \dfrac{\overset{(\pi)}{\vdash Pgc} \qquad Pgc \vdash Pf^2gc}{\vdash Pf^2gc}\, cut}{\vdash Pg^2c}\, para \qquad Pg^2c \vdash}{\vdash}\, cut$$

where π is

$$\dfrac{\dfrac{\vdash Pc \qquad Pc \vdash Pf^2c}{\vdash Pf^2c}\, cut \qquad \dfrac{\vdash Pc \qquad Pc \vdash f^2c = gc}{\vdash f^2c = gc}\, cut}{\vdash Pgc}\, para$$

Note that γ' is an $\mathbf{LK}_=$-refutation of ground instances of clauses in $\mathrm{CL}(\varphi)$.

γ' can be used as a skeleton of a proof φ^* with only atomic cuts of the original end-sequent S. φ^* is called a CERES-*normal form* of the original proof φ. Below we give a formal definition. First we define a type of top normal form defined by a PR-deduction.

Definition 23 (*Top normal form*) Let $\mathcal{C}$: $\{C_1 \vdash D_1, \ldots, C_n \vdash D_n\}$ be a set of clauses, $\Gamma \vdash \Delta$ be a skolemized sequent and φ_i cut-free proofs of $C_i, \Gamma \vdash \Delta, D_i$ in $\mathbf{LK}_=$ for $i = 1, \ldots, n$. Let $\Phi = \{\varphi_1, \ldots, \varphi_n\}$. Given a PR-deduction ϱ of a clause $C \vdash D$ from $\mathcal{C}$ we define an $\mathbf{LK}_=$-proof $\Theta(\varrho, \mathcal{C}, \Phi)$ of $C, \Gamma \vdash \Delta, D$ inductively on the length of ϱ.

- $\varrho = C_i \vdash D_i$: then $\Theta(\varrho, \mathcal{C}, \Phi) = \varphi_i$ and $top(\Theta(\varrho, \mathcal{C}, \Phi)) = \{\varphi_i\}$.
- The last inference in ϱ is R. Then ϱ is of the form

$$\dfrac{\overset{(\varrho_1)}{E_1 \vdash F_1, A^m} \qquad \overset{(\varrho_2)}{A^k, E_2 \vdash F_2}}{E_1, E_2 \vdash F_1, F_2}\, R$$

Let us assume that

$$\Theta(\varrho_1, \mathcal{C}, \Phi) = \overset{(\psi_1)}{E_1, \Gamma \vdash \Delta, F_1, A^m} \quad \Theta(\varrho_2, \mathcal{C}, \Phi) = \overset{(\psi_2)}{A^k, E_2, \Gamma \vdash \Delta, F_2}$$

Then we define $\Theta(\varrho, \mathcal{C}, \Phi) =$

$$\dfrac{\dfrac{\overset{(\psi_1)}{E_1, \Gamma \vdash \Delta, F_1, A^m} \qquad \overset{(\psi_2)}{A^k, E_2, \Gamma \vdash \Delta, F_2}}{E_1, E_2, \Gamma, \Gamma \vdash \Delta, \Delta, F_1, F_2}\, \text{cut}}{E_1, E_2, \Gamma \vdash \Delta, F_1, F_2}\, c^*$$

and $top(\Theta(\varrho, \mathcal{C}, \Phi)) = top(\Theta(\varrho_1, \mathcal{C}, \Phi)) \cup top(\Theta(\varrho_1, \mathcal{C}, \Phi))$.

– The last inference in ϱ is a paramodulation rule. We consider only the case $=_{r1}$; for the other rules the construction is analogous. Then ϱ is of the form

$$\dfrac{\begin{matrix}(\varrho_1) & (\varrho_2)\\ E_1 \vdash F_1, s = t & E_2 \vdash F_2, A[s]_\Lambda\end{matrix}}{E_1, E_2 \vdash F_1, F_2, A[t]_\Lambda}\ =_{r1}$$

Let us assume that

$$\begin{matrix}(\psi_1)\\ \Theta(\varrho_1, \mathcal{C}, \Phi) = E_1, \Gamma \vdash \Delta, F_1, s = t\end{matrix}\quad \begin{matrix}(\psi_2)\\ \Theta(\varrho_2, \mathcal{C}, \Phi) = E_2, \Gamma \vdash \Delta, F_2, A[s]_\Lambda\end{matrix}$$

Then we define $\Theta(\varrho, \mathcal{C}, \Phi) =$

$$\dfrac{\dfrac{\begin{matrix}(\psi_1) & (\psi_2)\\ E_1, \Gamma \vdash \Delta, F_1, s = t & E_2, \Gamma \vdash \Delta, F_2, A[s]_\Lambda\end{matrix}}{E_1, E_2, \Gamma, \Gamma \vdash \Delta, \Delta, F_1, F_2, A[t]_\Lambda}\ =_{r1}}{E_1, E_2, \Gamma \vdash \Delta, F_1, F_2, A[t]_\Lambda}\ c^*$$

and $top(\Theta(\varrho, \mathcal{C}, \Phi)) = top(\Theta(\varrho_1, \mathcal{C}, \Phi)) \cup top(\Theta(\varrho_1, \mathcal{C}, \Phi))$.

A proof ψ is called in *top normal form* if there are $\mathcal{C}$, Φ and ρ (defined as above) such that $\psi = \Theta(\rho, \mathcal{C}, \Phi)$.

Remark 2 The function *top* collects all cut-free subproofs in a top normal form which occur at the top and thus belong to Φ.

Definition 24 (CERES-*normal form*) Let φ be an $\mathbf{LK}_=$ proof of a skolemized sequent S. Let ϱ be a grounded PR-refutation of $\mathrm{CL}(\varphi)$, $\mathcal{C}$ be the set of all ground instances of clauses in $\mathrm{CL}(\varphi)$ appearing at the leaves of ϱ and Φ be the set of all grounded projections. Then the proof $\Theta(\varrho, \mathcal{C}, \Phi)$ is called a CERES *normal form* of ϱ. As ϱ is a refutation $\Theta(\varrho, \mathcal{C}, \Phi)$ is a proof of S with only atomic cuts.

Remark 3 Note that not all top normal forms are CERES normal forms as the set of cut-free proofs Φ need not be projections.

Example 10 We define a CERES normal form for the proof from Example 5 with respect to the grounded resolution refutation γ' of $\mathrm{CL}(\varphi)$ (in the following example $F = \forall x(Px \to Pfx)$):

$$\dfrac{\dfrac{\dfrac{\begin{matrix}(\varphi_1) & (\varphi_2)\\ Pc, F \vdash Pg^2c, \mathbf{f^2gc = g^2c} & Pc, F \vdash Pg^2c, \mathbf{Pf^2gc}\end{matrix}}{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, \mathbf{Pg^2c}}\ =_{r1}}{Pc, F \vdash \mathbf{Pg^2c}}\ c_l + c_r \qquad \dfrac{\varphi[Pg^2c \vdash]}{\mathbf{Pg^2c}, Pc, F \vdash Pg^2c}}{\dfrac{Pc, Pc, F, F \vdash Pg^2c, Pg^2c}{Pc, F \vdash Pg^2c}\ c_l + c_r}\ cut$$

where φ_1 is

$$\dfrac{\dfrac{\begin{matrix}\varphi_{11} & \varphi[Pz \vdash f^2z = gz]\{z \leftarrow gc\}\\ Pc, F \vdash Pg^2c, \mathbf{Pgc} & \mathbf{Pgc}, Pc, F \vdash Pg^2c, \mathbf{f^2gc = g^2c}\end{matrix}}{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, \mathbf{f^2gc = g^2c}}\ cut}{Pc, F \vdash Pg^2c, \mathbf{f^2gc = g^2c}}\ c_l + c_r$$

φ_{11} is

$$\dfrac{\dfrac{\dfrac{\pi_1}{Pc, F \vdash Pg^2c, \mathbf{Pf^2c}} \quad \dfrac{\pi_2}{Pc, F \vdash Pg^2c, \mathbf{f^2c = gc}}\ =_{r1}}{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, \mathbf{Pgc}}\ c_l + c_r}{Pc, F \vdash Pg^2c, \mathbf{Pgc}}$$

φ_2 is

$$\dfrac{\dfrac{\dfrac{\dfrac{\pi_1}{Pc, F \vdash Pg^2c, \mathbf{Pf^2c}} \quad \dfrac{\pi_2}{Pc, F \vdash Pg^2c, \mathbf{f^2c = gc}}\ =_{r1}}{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, \mathbf{Pgc}}\ c_l + c_r}{Pc, F \vdash Pg^2c, \mathbf{Pgc}} \quad \dfrac{\varphi[Pz \vdash Pf^2z]\{z \leftarrow gc\}}{\mathbf{Pgc}, Pc, F \vdash Pg^2c, \mathbf{Pf^2gc}}\ cut}{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, \mathbf{Pf^2gc}}\ c_l + c_r}{Pc, F \vdash Pg^2c, \mathbf{Pf^2gc}}$$

π_1 is

$$\dfrac{\dfrac{\varphi[\vdash Pc]}{Pc, F \vdash Pg^2c, \mathbf{Pc}} \quad \dfrac{\varphi[Pz \vdash Pf^2z]\{z \leftarrow c\}}{\mathbf{Pc}, Pc, F \vdash Pg^2c, \mathbf{Pf^2c}}\ cut}{Pc, F \vdash Pg^2c, \mathbf{Pf^2c}}$$

and π_2 is

$$\dfrac{\dfrac{\varphi[\vdash Pc]}{Pc, F \vdash Pg^2c, \mathbf{Pc}} \quad \dfrac{\varphi[Pz \vdash f^2z = gz]\{z \leftarrow c\}}{\mathbf{Pc}, Pc, F \vdash Pg^2c, \mathbf{f^2c = gc}}\ cut}{Pc, F \vdash Pg^2c, \mathbf{f^2c = gc}}$$

4 Extraction of Expansion Trees from Projections

The extraction of expansion proofs is usually performed after the construction of a proof in top normal form. However, only the *logical* parts of the proof play a role in the construction of expansion trees. These logical parts can be identified as the cut-free subproofs after removal of all cut-ancestors. Note that no cut-ancestor in such a subproof is principal formula of an inference; we identify such subsequents as *passive subsequent*.

Definition 25 (*Passive subsequent*) Let φ be a cut-free proof of $S : C, \Gamma \vdash \Delta, D$ such that $C \vdash D$ is a clause. The subsequent $C \vdash D$ of S is called *passive* in φ if no ancestor of $C \vdash D$ in φ contains a formula which is principal formula of an inference.

Note that the passive subsequents are just the clauses used to define a top normal form. Examples of proofs with passive clause parts are proof projections in CERES:

Proposition 2 *Let ψ be a cut-free proof of $C', \Gamma \vdash \Delta, D'$ which is an instance of a proof projection $\varphi[C \vdash D]$ in* CERES. *Then $C' \vdash D'$ is passive in ψ.*

Proof Immediate by induction on the length of ψ and by Definition 22. Note that the only case in Definition 22 where the clause part changes is (c2). Here the projection ψ is defined as

$$\dfrac{\dfrac{(\psi_1)}{C, \Gamma_e \vdash \Delta_e, D} \quad \dfrac{(\psi_2)}{E, \Pi_e \vdash \Lambda_e, F}\ \xi}{C, E, \Gamma'_e, \Pi'_e \vdash \Delta'_e, \Lambda'_e, D, F}$$

By induction hypothesis $C \vdash D$ is passive in ψ_1 and $E \vdash F$ is passive in ψ_2. Therefore $C, E \vdash D, F$ is passive in ψ. □

Definition 26 Let φ be a cut-free proof of $C, \Gamma \vdash \Delta, D$ where $C \vdash D$ is passive in φ. We define $\varphi\backslash(C \vdash D)$ by induction on the number of nodes in φ.

- If φ is an axiom then $\varphi = C, C' \vdash D, D'$ (note that the whole sequent is passive in φ). We define $\varphi\backslash(C \vdash D) = C' \vdash D'$.
- Let $\varphi =$

$$\dfrac{\varphi'}{C, \Gamma \vdash \Delta, D}\, x$$

where $C \vdash D$ is passive in φ. Then, by definition of passive subclauses, φ' is a proof of $C, \Gamma' \vdash \Delta', D$ for some Γ' and Δ'. Indeed, the subclause $C \vdash D$ does not contain a formula which is principal formula of an inference. By induction we have a proof $\varphi'\backslash(C \vdash D)$ of $\Gamma' \vdash \Delta'$ (note that $C \vdash D$ is also passive in φ') and we define $\varphi\backslash(C \vdash D) =$

$$\dfrac{\varphi'\backslash(C \vdash D)}{\Gamma' \vdash \Delta'}\, x$$

- Let $\varphi =$

$$\dfrac{\begin{array}{cc}(\varphi_1) & (\varphi_2) \\ S_1 & S_2\end{array}}{C, \Gamma \vdash \Delta, D}\, x$$

where $C \vdash D$ is passive in φ. As C, D are not principal formulas of an inference we get that $S_1 = C_1, \Gamma_1 \vdash \Delta_1, D_1$, $S_2 = C_2, \Gamma_2 \vdash \Delta_2, D_2$, s.t. $C_1, C_2 \vdash D_1, D_2 = C \vdash D$ and $C_1 \vdash D_1$ is passive in φ_1, $C_2 \vdash D_2$ is passive in φ_2.
By induction we have a proof $\varphi_1\backslash(C_1 \vdash D_1)$ of $\Gamma_1 \vdash \Delta_1$ and a proof $\varphi_2\backslash(C_2 \vdash D_2)$ of $\Gamma_2 \vdash \Delta_2$. Then we obtain $\varphi\backslash(C \vdash D) =$

$$\dfrac{\begin{array}{cc}(\varphi_1\backslash(C_1 \vdash D_1)) & (\varphi_2\backslash(C_2 \vdash D_2)) \\ \Gamma_1 \vdash \Delta_1 & \Gamma_2 \vdash \Delta_2\end{array}}{\Gamma \vdash \Delta}\, x$$

The function logical(φ) for a proof in top normal form takes the cut-free proofs on top and "subtracts" from them all ancestors of passive clauses.

Definition 27 (logical(φ)) Let $\varphi\colon \Theta(\rho, \mathcal{C}, \Phi)$ be a proof in top normal form s.t. $\mathcal{C} = \{C_1 \vdash D_1, \ldots, C_n \vdash D_n\}$ and $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ such that φ_i is a cut-free proof of $C_i, \Gamma \vdash \Delta, D_i$. Assume that for all $i = 1, \ldots, n$ $C_i \vdash D_i$ is passive in φ_i. For every $\psi \in top(\varphi)$ and $\psi = \varphi_i$ we define $\psi' = \varphi_i\backslash(C_i \vdash D_i)$ and logical(φ) $= \{\psi' \mid \psi \in top(\varphi)\}$.

Below we define an expansion tree $\hat{E}(\varphi)$ which is defined by merging the expansion trees of logical(φ). This structure will be the key for the development of an efficient algorithm for extracting expansion trees from CERES normal forms.

Definition 28 Let φ be a proof in top normal form of a skolemized and normalized end-sequent. We define

$$\hat{E}(\varphi) = \text{merge}\{\text{ET}(\psi) \mid \psi \in \text{logical}(\varphi)\}.$$

Theorem 2 *Let* $\varphi\colon\ \Theta(\varrho, \mathcal{C}, \Phi)$ *be a proof of a skolemized and normalized sequent* $C, \Gamma \vdash \Delta, D$ *in top normal form such that* $\mathcal{C} = \{C_1 \vdash D_1, \ldots, C_n \vdash D_n\}$ *and* $\Phi = \{\varphi_1, \ldots, \varphi_n\}$, *where* φ_i *is a cut-free proof of* $C_i, \Gamma \vdash \Delta, D_i$. *Assume that for all* $i = 1, \ldots, n$ $C_i \vdash D_i$ *is passive in* φ_i. *Then* $\mathrm{ET}(\varphi) = \hat{E}(\varphi) \circ (C \vdash D)$.

Proof By induction on the number of nodes in ϱ.

Case 1: if ρ consists of just one node then $\varphi = \varphi_i$ for some $i \in \{1, \ldots, n\}$. We have to show that $\mathrm{ET}(\varphi_i) = \hat{E}(\varphi_i) \circ (C_i \vdash D_i)$. But $\hat{E}(\varphi_i) = \mathrm{ET}(\varphi_i \backslash (C_i \vdash D_i))$ and thus it remains to show that

$$(\star)\ \mathrm{ET}(\varphi_i) = \mathrm{ET}(\varphi_i \backslash (C_i \vdash D_i)) \circ (C_i \vdash D_i).$$

$(\star)$ is obtained via an easy induction on the number of inferences in φ_i using Definition 26.
Case 2: The last inference in ϱ is R. Then ϱ is of the form

$$\frac{\begin{matrix}(\varrho_1) & (\varrho_2)\\ C_1 \vdash D_1, A^m & A^k, C_2 \vdash D_2\end{matrix}}{C_1, C_2 \vdash D_1, D_2}\ R$$

Then (by definition of φ as $\Theta(\varrho, \mathcal{C}, \Phi)$) φ=

$$\frac{\dfrac{\begin{matrix}(\varphi_1) & (\varphi_2)\\ C_1, \Gamma \vdash \Delta, D_1, A^m & A^k, C_2, \Gamma \vdash \Delta, D_2\end{matrix}}{C_1, C_2, \Gamma, \Gamma \vdash \Delta, \Delta, D_1, D_2}\ \mathrm{cut}}{C_1, C_2, \Gamma \vdash \Delta, D_1, D_2}\ c^*$$

Assume that

$$\mathrm{ET}(\varphi_1) = C_1, \Gamma^* \vdash \Delta^*, D_1, A^m,$$
$$\mathrm{ET}(\varphi_2) = A^k, C_2, \Gamma^+ \vdash \Delta^+, D_2,$$

where $Seq(\Gamma^* \vdash \Delta^*) = Seq(\Gamma^+ \vdash \Delta^+)$. By Definition 19 we obtain

$$(1)\ \mathrm{ET}(\varphi) = (C_1, C_2 \vdash D_1, D_2) \circ \mathrm{merge}(\Gamma^* \vdash \Delta^*, \Gamma^+ \vdash \Delta^+)$$

Note that Γ^*, Γ^+ and Δ^*, Δ^+ are normalized. By induction hypothesis we have

$$\hat{E}(\varphi_1) \circ (C_1 \vdash D_1, A^m) = \mathrm{ET}(\varphi_1),$$
$$\hat{E}(\varphi_2) \circ (A^k, C_2 \vdash D_2) = \mathrm{ET}(\varphi_2).$$

and therefore

$$(2)\ \hat{E}(\varphi_1) = \Gamma^* \vdash \Delta^*,\ \ \hat{E}(\varphi_2) = \Gamma^+ \vdash \Delta^+.$$

By definition of the merge operator we get from (1) and (2)

$$(3)\ \mathrm{ET}(\varphi) = (C_1, C_2 \vdash D_1, D_2) \circ \mathrm{merge}(\hat{E}(\varphi_1), \hat{E}(\varphi_2)).$$

By Definition 28 we obtain

$$\hat{E}(\varphi_1) = \mathrm{merge}\{\mathrm{ET}(\psi) \mid \psi \in \mathrm{logical}(\varphi_1)\},$$
$$\hat{E}(\varphi_2) = \mathrm{merge}\{\mathrm{ET}(\psi) \mid \psi \in \mathrm{logical}(\varphi_2)\}.$$

Hence

$$\begin{array}{l}\text{merge}(\hat{E}(\varphi_1), \hat{E}(\varphi_2)) = \\ \text{merge}(\text{merge}\{\text{ET}(\psi) \mid \psi \in \text{logical}(\varphi_1)\}, \text{merge}\{\text{ET}(\psi) \mid \psi \in \text{logical}(\varphi_2)\}) = \\ \text{merge}\{\text{ET}(\psi) \mid \psi \in \text{logical}(\varphi)\}.\end{array}$$

But by Definition 28 $\hat{E}(\varphi) = \text{merge}\{\text{ET}(\psi) \mid \psi \in \text{logical}(\varphi)\}$. Combing this with (3) we obtain

$$\text{ET}(\varphi) = (C_1, C_2 \vdash D_1, D_2) \circ \hat{E}(\varphi).$$

Case 3: The last inference in ϱ is a paramodulation. We only consider the case $=_{r_1}$, the others are analogous. Then ϱ is of the form

$$\dfrac{\begin{array}{cc}(\varrho_1) & (\varrho_2) \\ C_1 \vdash D_1, s = t & C_2 \vdash D_2, A[s]\end{array}}{C_1, C_2 \vdash D_1, D_2, A[t]} =_{r_1}$$

Then, by definition of φ, we obtain $\varphi=$

$$\dfrac{\dfrac{\begin{array}{cc}(\varphi_1) & (\varphi_2) \\ C_1, \Gamma \vdash \Delta, D_1, s = t & C_2, \Gamma \vdash \Delta, D_2, A[s]\end{array}}{C_1, C_2, \Gamma, \Gamma \vdash \Delta, \Delta, D_1, D_2, A[t]}\, c^*}{C_1, C_2, \Gamma \vdash \Delta, D_1, D_2, A[t]} =_{r_1}$$

Assume that

$$\begin{array}{l}\text{ET}(\varphi_1) = C_1, \Gamma^* \vdash \Delta^*, D_1, s = t, \\ \text{ET}(\varphi_2) = C_2, \Gamma^+ \vdash \Delta^+, D_2, A[s]\end{array}$$

where $Seq(\Gamma^* \vdash \Delta^*) = Seq(\Gamma^+ \vdash \Delta^+)$. By Definition 19 we obtain

$$(4)\ \text{ET}(\varphi) = (C_1, C_2 \vdash D_1, D_2, A[t]) \circ \text{merge}(\Gamma^* \vdash \Delta^*, \Gamma^+ \vdash \Delta^+).$$

By induction hypothesis we have

$$\begin{array}{l}\hat{E}(\varphi_1) \circ (C_1 \vdash D_1, s = t) = \text{ET}(\varphi_1), \\ \hat{E}(\varphi_2) \circ (C_2 \vdash D_2, A[s]) = \text{ET}(\varphi_2).\end{array}$$

and therefore

$$(5)\ \hat{E}(\varphi_1) = \Gamma^* \vdash \Delta^*, \quad \hat{E}(\varphi_2) = \Gamma^+ \vdash \Delta^+.$$

By definition of the merge operator we get from (4) and (5)

$$(6)\ \text{ET}(\varphi) = (C_1, C_2 \vdash D_1, D_2, A[t]) \circ \text{merge}(\hat{E}(\varphi_1), \hat{E}(\varphi_2)).$$

By Definition 28 we obtain

$$\begin{array}{l}\hat{E}(\varphi_1) = \text{merge}\{\text{ET}(\psi) \mid \psi \in \text{logical}(\varphi_1)\}, \\ \hat{E}(\varphi_2) = \text{merge}\{\text{ET}(\psi) \mid \psi \in \text{logical}(\varphi_2)\}.\end{array}$$

Hence, like in Case 2, we get

$$\text{merge}(\hat{E}(\varphi_1), \hat{E}(\varphi_2)) = \text{merge}\{\text{ET}(\psi) \mid \psi \in \text{logical}(\varphi)\}.$$

But by Definition 28 $\hat{E}(\varphi) = \text{merge}\{\text{ET}(\psi) \mid \psi \in \text{logical}(\varphi)\}$. Combing this with (6) we obtain

$$\text{ET}(\varphi) = (C_1, C_2 \vdash D_1, D_2, A[t]) \circ \hat{E}(\varphi).$$

□

Corollary 1 *Let $\varphi\colon \Theta(\varrho, \mathcal{C}, \Phi)$ be a proof of a skolemized and normalized sequent $\Gamma \vdash \Delta$ in top normal form s.t. $\mathcal{C} = \{C_1 \vdash D_1, \ldots, C_n \vdash D_n\}$ and $\Phi = \{\varphi_1, \ldots, \varphi_n\}$ such that φ_i is a cut-free proof of $C_i, \Gamma \vdash \Delta, D_i$. Assume that for all $i = 1, \ldots, n$ $C_i \vdash D_i$ is passive in φ_i. Then* $\text{ET}(\varphi) = \hat{E}(\varphi)$.

Proof Immediate by Theorem 2: just define $C \vdash D$ as the empty sequent. □

Corollary 2 *Let φ be an* $\mathbf{LK}_=$ *proof of a skolemized and normalized sequent S. Let $\varphi^*\colon \Theta(\varrho, \mathcal{C}, \Phi)$ be a* CERES *normal form of φ such that ϱ is a ground PR-refutation of $\mathcal{C}$, the set of all ground instances of clauses in* $\text{CL}(\varphi)$*, and Φ is the set of all grounded projections. Then* $\text{ET}(\varphi) = \hat{E}(\varphi)$.

Proof Let ψ be a cut-free proof of $C, \Gamma \vdash \Delta, D$ which is an instance of a projection of φ. By Proposition 2 $C \vdash D$ is passive in ψ. As CERES normal forms are top normal forms all conditions of Corollary 1 are fulfilled. □

The last corollary describes a method to compute an expansion tree from any proof in top normal form of a skolemized sequent S. Note that in case of a prenex sequent S we extract Herbrand sequents. The computation of an expansion tree is based on top normal forms. CERES normal forms φ^* of proofs φ are also in top normal form, therefore we can compute expansion trees in the same way. For CERES it means that φ^* has to be constructed first. The usual algorithm for the extraction of expansion trees from the CERES normal form is:

begin % algorithm EXP
1. compute $\text{CL}(\varphi)$;
2. find a PR refutation ρ of $\text{CL}(\varphi)$;
3. compute a ground refutation R from ρ;
4. compute the projections $\varphi[C]$ for $C \in \text{CL}(\varphi, R)$;
5. construct the CERES normal form φ^* from the projections and R;
6. extract an expansion proof $\text{ET}(\varphi)$ from φ^*.

end.

Instead of using algorithm EXP, we make use of Theorem 2 and define a new method that extracts expansion proofs more efficiently by extracting partial expansion trees from the projections. The idea is the following: we do not compute $\text{logical}(\varphi^*)$ which would be the set of all instantiated projections. Note that the size of $\text{logical}(\varphi^*)$ is roughly the size of φ^* itself. Instead we use from a ground resolution refutation R of $\text{CL}(\varphi)$ the general projections $\varphi[C]$ for $C \in \text{CL}(\varphi, R)$ and the set of substitutions $\Sigma(C)$ for $C \in \text{CL}(\varphi, R)$ which are the set of ground substitutions for the clause C in the refutation R.

Definition 29 For every projection $\varphi[C]\colon \mathbf{A}, \Gamma \vdash \Delta, \mathbf{B}$, where $\mathbf{A} \vdash \mathbf{B}$ is the clause part of $\varphi[C]$, we define $\varphi^-[C] = \varphi(C)\backslash(\mathbf{A} \vdash \mathbf{B})$ (note that $\mathbf{A} \vdash \mathbf{B}$ is a passive subsequent of $\mathbf{A}, \Gamma \vdash \Delta, \mathbf{B}$), where the $\backslash$-operator is defined as in Definition 26. Note that $\text{ET}(\varphi^-[C])$ is a proof relative to the axioms in $\varphi^-[C]$, which may differ from the axioms of φ (axioms need not be tautological anyway). We define

$$T(\varphi, R) = \text{merge}_{C \in \text{CL}(\varphi, R)} \text{merge}_{\sigma \in \Sigma(C)} \text{ET}(\varphi^-[C])\sigma.$$

Then the computation of an expansion tree via CERES for a proof φ (of a closed skolemized end-sequent S) goes as follows:

begin % algorithm EXP_{new}
1. compute $\mathrm{CL}(\varphi)$;
2. find a PR refutation ρ of $\mathrm{CL}(\varphi)$;
3. compute a ground refutation R from ρ
   and for every $C \in \mathrm{CL}(\varphi, R)$ the set $\Sigma(C)$;
4. compute the projections $\varphi[C]$ and $\varphi^-[C]$ for $C \in \mathrm{CL}(\varphi, R)$;
5. for every $C \in \mathrm{CL}(\varphi, R)$ compute $T[C]$: $\mathrm{merge}_{\sigma\in\Sigma(C)}\mathrm{ET}(\varphi^-[C])\sigma$;
6. compute $\mathrm{merge}_{C\in\mathrm{CL}(\varphi,R)}T[C]$ which is $T(\varphi, R)$.

end.

Note that the computation of the *Dp* function of an expansion proof via CERES for a proof φ (of a closed skolemized end-sequent S) can be easily obtained by computing the *Dp* function of $T(\varphi, R)$.

Theorem 3 *Let φ be a proof of a skolemized, closed and normalized end-sequent and φ^* the* CERES *normal form based on a ground refutation R of* $\mathrm{CL}(\varphi)$. *Then* $\mathrm{ET}(\varphi^*) = T(\varphi, R)$.

Proof Let $\mathrm{CL}(\varphi, R) = \{C_1, \ldots, C_n\}$. Now $\mathrm{logical}(\varphi^*) = \Phi_1 \cup \cdots \cup \Phi_n$ where $\Phi_i = \{\varphi^-[C_i]\sigma \mid \sigma \in \Sigma(C_i)\}$. Let $\psi_i = \varphi^-[C_i]$. Then by Theorem 2 we know that

$$\mathrm{ET}(\varphi^*) = \mathrm{merge}(\mathrm{merge}_{\sigma\in\Sigma(C_1)}\mathrm{ET}(\psi_1\sigma), \cdots, \mathrm{merge}_{\sigma\in\Sigma(C_n)}\mathrm{ET}(\psi_n\sigma))$$

which is equal to

$$\mathrm{ET}(\varphi^*) = \mathrm{merge}(\mathrm{merge}_{\sigma\in\Sigma(C_1)}\mathrm{ET}(\psi_1)\sigma, \cdots, \mathrm{merge}_{\sigma\in\Sigma(C_n)}\mathrm{ET}(\psi_n)\sigma)$$

which, by Definition 29, is exactly $T(\varphi, R)$. So $\mathrm{ET}(\varphi^*) = T(\varphi, R)$. □

Note that Theorem 3 also holds for the *Dp* function of expansion proofs, i.e. $Dp(\mathrm{ET}(\varphi^*)) = Dp(T(\varphi, R))$.

Corollary 3 *Let φ be a proof of a skolemized,closed and normalized sequent S and R be a refutation of* $\mathrm{CL}(\varphi)$. *Then $T(\varphi, R)$ is an expansion proof of S.*

Proof By Theorems 2 and 3. □

Instead of computing all $\varphi[C_j]\sigma_i^j$ (obtained from the ACNF φ^*) the algorithm EXP_{new} computes the $\varphi[C_j]$ and extracts $\mathrm{ET}(\varphi^-[C_j]) \equiv T_j$, which is a partial expansion proof, then constructs $\times_{\sigma\in\Sigma(C_j)}T_j\sigma$ for all j and merges them. Example 11 illustrates the main features of the method.

Example 11 Consider the proof φ as in Example 5 (where $F = \forall x(Px \to Pfx)$). The ACNF φ is:

$$\dfrac{\dfrac{\dfrac{(\varphi_1) \qquad (\varphi_2)}{Pc, F \vdash Pg^2c, \mathbf{f^2gc} = \mathbf{g^2}c \qquad Pc, F \vdash Pg^2c, \mathbf{Pf^2gc}}\ para}{\dfrac{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, \mathbf{Pg^2c}}{Pc, F \vdash \mathbf{Pg^2c}}\ c_l + c_r} \qquad \dfrac{\varphi[Pg^2c \vdash]}{\mathbf{Pg^2c}, Pc, F \vdash Pg^2c}}{\dfrac{Pc, Pc, F, F \vdash Pg^2c, Pg^2c}{Pc, F \vdash Pg^2c}\ c_l + c_r}\ cut$$

where φ_1 is

$$\dfrac{\dfrac{\dfrac{\varphi_{1_1} \qquad \varphi[Pz \vdash f^2z = gz]\{z \leftarrow gc\}}{Pc, F \vdash Pg^2c, \mathbf{Pgc} \qquad \mathbf{Pgc}, Pc, F \vdash Pg^2c, \mathbf{f^2gc = g^2c}}\, cut}{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, \mathbf{f^2gc = g^2c}}\, c_l + c_r}{Pc, F \vdash Pg^2c, \mathbf{f^2gc = g^2c}}$$

φ_{1_1} is

$$\dfrac{\dfrac{\dfrac{\pi_1 \qquad \pi_2}{Pc, F \vdash Pg^2c, \mathbf{Pf^2c} \qquad Pc, F \vdash Pg^2c, \mathbf{f^2c = gc}}\, para}{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, \mathbf{Pgc}}\, c_l + c_r}{Pc, F \vdash Pg^2c, \mathbf{Pgc}}$$

φ_2 is

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\pi_1 \qquad \pi_2}{Pc, F \vdash Pg^2c, \mathbf{Pf^2c} \qquad Pc, F \vdash Pg^2c, \mathbf{f^2c = gc}}\, para}{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, \mathbf{Pgc}}\, c_l + c_r \qquad \varphi[Pz \vdash Pf^2z]\{z \leftarrow gc\}}{Pc, F \vdash Pg^2c, \mathbf{Pgc} \qquad \mathbf{Pgc}, Pc, F \vdash Pg^2c, \mathbf{Pf^2gc}}\, cut}{Pc, Pc, F, F \vdash Pg^2c, Pg^2c, \mathbf{Pf^2gc}}\, c_l + c_r}{Pc, F \vdash Pg^2c, \mathbf{Pf^2gc}}$$

π_1 is

$$\dfrac{\varphi[\vdash Pc] \qquad \varphi[Pz \vdash Pf^2z]\{z \leftarrow c\}}{Pc, F \vdash Pg^2c, \mathbf{Pc} \qquad \mathbf{Pc}, Pc, F \vdash Pg^2c, \mathbf{Pf^2c}}\, cut$$
$$Pc, F \vdash Pg^2c, \mathbf{Pf^2c}$$

and π_2 is

$$\dfrac{\varphi[\vdash Pc] \qquad \varphi[Pz \vdash f^2z = gz]\{z \leftarrow c\}}{Pc, F \vdash Pg^2c, \mathbf{Pc} \qquad \mathbf{Pc}, Pc, F \vdash Pg^2c, \mathbf{f^2c = gc}}\, cut$$
$$Pc, F \vdash Pg^2c, \mathbf{f^2c = gc}$$

Now compute the Herbrand sequent of φ^* (with the old method):

$$H(\varphi^*) = Pc, Pc \to Pfc, Pfc \to Pf^2c, Pgc \to Pfgc, Pfgc \to Pf^2gc \vdash Pg^2c$$

Note that $H(\varphi^*)$ is a valid sequent in our axiom set ($\vdash f^2z = gz$ is an axiom).

With our new method we first compute $T_i = \mathrm{ET}(\varphi^-[C_i])$ and define the substitutions σ_i^j:

$$\begin{aligned}
T_1 &= Pc, \forall x(Px \to Pfx) +^z Pz \to Pfz +^{fz} Pfz \to Pf^2z \vdash Pg^2c\\
T_2 &= Pc, \forall x(Px \to Pfx) +^z Pz \to Pfz +^{fz} Pfz \to Pf^2 \vdash Pg^2c\\
T_3 &= Pc, \forall x(Px \to Pfx) \vdash Pg^2c\\
T_4 &= Pc, \forall x(Px \to Pfx) \vdash Pg^2c\\
\sigma_1^1 &= (z \leftarrow c) \quad \sigma_2^1 = (z \leftarrow gc)\\
\sigma_1^2 &= (z \leftarrow c) \quad \sigma_2^2 = (z \leftarrow gc)
\end{aligned}$$

Note that $T_1 = T_2$. Now we compute $T(\varphi, R) = \text{merge}(T_1\,\sigma_1^1, T_1\,\sigma_2^1, T_2\,\sigma_1^2, T_2\,\sigma_2^2, T_3, T_4)$

$$\begin{aligned} T(\varphi, R) = \text{merge}(&Pc, \forall x(Px \to Pfx) +^c Pc \to Pfc +^{fc} Pfc \to Pf^2c \\ &\vdash Pg^2c, \\ &(Pc, \forall x(Px \to Pfx) +^{gc} Pgc \to Pfgc +^{fgc} Pfgc \to Pf^2gc \\ &\vdash Pg^2c, \\ &Pc, \forall x(Px \to Pfx) \vdash Pg^2c, \\ &Pc, \forall x(Px \to Pfx) \vdash Pg^2c) \end{aligned}$$

$$\begin{aligned} T(\varphi, R) = Pc, \forall x(Px \to Pfx) &+^c Pc \to Pfc, \\ &+^{fc} Pfc \to Pf^2c, \\ &+^{gc} Pgc \to Pfgc, \\ &+^{fgc} Pfgc \to Pf^2gc \vdash Pg^2c \end{aligned}$$

The *Dp* function is

$$Dp(T(\varphi, R))) = Pc, Pc \to Pfc, Pfc \to Pf^2c, Pgc \to Pfgc, Pfgc \to Pf^2gc \vdash Pg^2c$$

5 Complexity

In this section we prove that the algorithm EXP_{new} outperforms the old algorithm EXP. In particular we prove that the complexity of EXP_{new} is always better or equal to that of EXP. Then we define an infinite sequence of **LK**-proofs φ_n where the complexity of EXP is cubic in n while that of EXP_{new} is only quadratic. This implies that the computational complexity of EXP cannot be linearly bounded by that of EXP_{new}. Our complexity measure will be the maximal logical complexity of objects constructed by the algorithms.

Definition 30 (*Size of a formula*) Let A be a formula, then the size of A ($\|A\|_f$) is inductively defined as follows

$$\begin{aligned} \|A\|_f &= 1 \text{ if } A \text{ is an atomic formula ,} \\ \|\neg A\|_f &= 1 + \|A\|_f, \\ \|A_1 \circ A_2\|_f &= 1 + \|A_1\|_f + \|A_2\|_f, \circ \in \{\wedge, \vee, \to\}, \\ \|Qx.A\|_f &= 1 + \|A\|_f, Q \in \{\forall, \exists\}. \end{aligned}$$

To improve legibility we write $\|F\|$ instead of $\|F\|_f$ (with the exception of cases where the precise notation is essential) and use the measure $\|\ \|$ also for sequents, proofs and clause sets.

Definition 31 (*Size of a sequent*) Let $S : A_1, \ldots, A_n \vdash A_{n+1}, \ldots, A_m$ be a sequent, then the size of S is

$$\|A_1, \ldots, A_n \vdash A_{n+1}, \ldots, A_m\| = \Sigma_{i=1}^m \|A_i\|$$

Definition 32 (*Size of an* $\mathbf{LK}_=$*-proof*) Let φ be an $\mathbf{LK}_=$-proof. If φ is an axiom then φ consists of just one node labelled by a sequent S; here we define $\|\varphi\| = \|S\|$.

If φ is not an axiom then the end-sequent is a conclusion of a unary or of a binary rule. So we distinguish two cases:

(a) $\varphi =$

$$\frac{\varphi'}{S}\ x$$

Then $\|\varphi\| = \|\varphi'\| + \|S\|$.

(b) $\varphi =$

$$\frac{\varphi_1 \quad \varphi_2}{S}\ y$$

Then $\|\varphi\| = \|\varphi_1\| + \|\varphi_2\| + \|S\|$.

Definition 33 (*Size of an expansion tree*) Let E be an expansion tree, then the size of E ($\|E\|$) is inductively defined as follows

$$\begin{array}{ll}
\|E\| & = \|E\|_f \text{ if } E \text{ is a quantifier-free formula ,} \\
\|\neg E\| & = 1 + \|E\|, \\
\|E_1 \circ E_2\| & = 1 + \|E_1\| + \|E_2\|, \circ \in \{\wedge, \vee, \rightarrow\}, \\
\|Qx.E +^{t_1} E_1 +^{t_2} \cdots +^{t_n} E_n\| & = 1 + \|E\| + \|E_1\| + \cdots + \|E_n\|, Q \in \{\forall, \exists\}.
\end{array}$$

Definition 34 (*Size of an s-expansion proof*) Let $E : E_1, \ldots, E_n \vdash E_{n+1}, \ldots, E_m$ be an s-expansion proof, then the size of E ($\|E\|$) is

$$\|E_1, \ldots, E_n \vdash E_{n+1}, \ldots, E_m\| = \Sigma_{i=1}^{m} \|E_i\|.$$

In our algorithms EXP and EXP_{new} we do not only construct sequents, formulas and proofs, but also sets of clauses (which are finite sets of atomic sequents). If $\mathcal{C} = \{C_1, \ldots, C_n\}$ we define $\|\mathcal{C}\| = \|C_1\| + \ldots + \|C_n\|$. We call the objects produced by a proof transformation expressions.

Definition 35 (*Expression*) An *expression* is a formula, a sequent, a proof or a set of clauses.

Now we consider computations as sequences of expressions which are generated by an algorithmic proof transformation. So let A be an algorithm and φ be a proof serving as input to A. Then $E_A(\varphi)$ is the sequence of all expressions generated by A on input φ. Below we define a complexity function induced by A given by the maximal expression generated by A:

Definition 36 Let A be an algorithm on proofs. Then we define

$$C_A(\varphi) = \max\{\|x\| \mid x \in E_A(\varphi)\}.$$

Theorem 4 $C_{\text{EXP}_{new}}(\varphi) \leq C_{\text{EXP}}(\varphi)$ *for all proofs* φ *in* $\mathbf{LK}_{=}$.

Proof sketch: the first 4 steps of EXP and EXP_{new} are identical. The sum of the sizes of the expansion trees generated by EXP_{new} is smaller or equal to the size of the CERES normal form generated by EXP. □

We show now that EXP_{new} can be asymptotically better that EXP. To this aim we define the following sequence of **LK**-proofs φ_n:

$$\dfrac{\dfrac{}{\forall x(Px \rightarrow Pfx) \vdash \forall x(Px \rightarrow Pf^n x)}(\omega_n) \qquad \dfrac{}{\forall x(Px \rightarrow Pf^n x), Pa \vdash Pf^{n^2} a}(\pi_n)}{Pa, \forall x(Px \rightarrow Pfx) \vdash Pf^{n^2} a}\ cut$$

where ω_n is:

$$\dfrac{\dfrac{\dfrac{(\psi_n)}{Py, Py \to Pfy, \ldots, Pf^{n-1}y \to Pf^n y \vdash Pf^n y}\to_r}{Py \to Pfy, \ldots, Pf^{n-1}y \to Pf^n y \vdash Py \to Pf^n y}\forall_l(n\times)}{\forall x(Px \to Pfx) \vdash Py \to Pf^n y}\forall_r$$

$$\forall x(Px \to Pfx) \vdash \forall x(Px \to Pf^n x)$$

and π_n is:

$$\dfrac{\dfrac{(\chi_n)}{Pa \to Pf^n a, Pf^n a \to Pf^{2n}a, \ldots, Pf^{(n-1)n}a \to Pf^{n^2}a, Pa \vdash Pf^{n^2}a}\forall_l(n\times)}{\forall x(Px \to Pf^n x), Pa \vdash Pf^{n^2}a}$$

recursive definition of ψ_n: $\psi_0 = Py \vdash Py$. And for $n > 0$ $\psi_n =$

$$\dfrac{Py \vdash Py \qquad \overset{\psi_{n-1}\{y \leftarrow fy\}}{Pfy, Pfy \to Pf^2 y, \ldots, Pf^{n-1}y \to Pf^n y \vdash Pf^n y}}{Py, Py \to Pfy, \ldots, Pf^{n-1}y \to Pf^n y \vdash Pf^n y}\to_l$$

For our complexity measure we obtain

$$\|\psi_n\| = 2 + 2(n+1) + \|\psi_{n-1}\|$$

note that $\|\psi_{n-1}\| = \|\psi_{n-1}\{y \leftarrow fy\}\|$.

$$C_P(\psi_0) = 2$$

Obviously, there are constants a_1, a_2, b_1, b_2 (all > 0) such that

$$a_1 * n^2 \leq \|\psi_n\| \leq a_2 * (n+1)^2 \text{ and } b_1 * n^2 \leq \|\chi_n\| \leq b_2 * (n+1)^2.$$

Putting things together there are constants $c_1, c_2 > 0$ with

$$c_1 * n^2 \leq \|\varphi_n\| \leq c_2 * (n+1)^2.$$

Now we compute the characteristic clause sets of the φ_n. After elimination of tautologies we get

$$CL(\varphi_n) = \{C_{1,n} : Py \vdash Pf^n y;\ C_2 : \vdash Pa;\ C_{3,n} : Pf^{n^2}a \vdash\}.$$

Now we compute the resolution refutation. The recursive definition is the following:

γ_1:

$$\dfrac{\vdash Pa \qquad Pa \vdash Pf^n a}{\vdash Pf^n a}\,R$$

γ_n:

$$\dfrac{\overset{(\gamma_{n-1})}{\vdash Pf^{(n-1)n}a} \qquad Pf^{(n-1)n}a \vdash Pf^{n^2}a}{\vdash Pf^{n^2}a}\,R$$

We obtain $\|\gamma_n\| = \|\gamma_{n-1}\| + 3$.

Thus, the resolution schema is δ_n:

$$\dfrac{\overset{(\gamma_n)}{\vdash Pf^{n^2}a} \qquad Pf^{n^2}a \vdash}{\vdash}\,R$$

with substitutions $\{y \leftarrow a\}, \ldots, \{y \leftarrow f^{(n-1)n}a\}$. We then get

$$\|\delta_n\| = 3n + 2.$$

Concerning the projections we adapt an improved version, *minimal projections*, which is also used in the implementation of CERES. In this form of projection it is sufficient to derive just a subsequent of the end-sequent which reduces the number of weakenings and contractions in the ACNF.

$\varphi_n[C_{1,n}]$:

$$\dfrac{\dfrac{(\psi_n)}{Py, Py \to Pfy, \ldots, Pf^{n-1}y \to Pf^n y \vdash Pf^n y}}{Py, \forall x(Px \to Pfx) \vdash Pf^n y}\, \forall_l(nx) + c_l(nx)$$

$\varphi_n[C_2]$ (minimal projection)

$$Pa \vdash Pa$$

$\varphi_n[C_3]$ (minimal projection):

$$Pf^{n^2}a \vdash Pf^{n^2}a$$

Now we can construct the ACNF. φ_n^* is the ACNF-schema:

φ_1^*:

$$\dfrac{\dfrac{Pa \vdash Pa \qquad \overset{(\varphi_n[C_{1,n}\{y \leftarrow a\}])}{Pa, F \vdash Pf^n a}}{Pa, F \vdash Pf^n a}\, cut \qquad \overset{(\varphi_n[C_{1,n}\{y \leftarrow f^n a\}])}{Pf^n a, F \vdash Pf^{2n} a}}{Pa, F \vdash Pf^{2n} a}\, cut + c^*$$

φ_n^*:

$$\dfrac{\dfrac{\overset{(\varphi_{n-1}^*)}{\cdots}\quad Pa, F \vdash Pf^{(n-1)n}a \qquad \overset{(\varphi_n[C_{1,n}\{y \leftarrow f^{(n-1)n}a\}])}{Pf^{(n-1)n}a \vdash Pf^{n^2}a}}{Pa, F \vdash Pf^{n^2}a}\, cut + c^* \qquad \overset{(\varphi_n[C_{3,n}])}{Pf^{n^2}a \vdash Pf^{n^2}a}}{Pa, F \vdash Pf^{n^2}a}\, cut$$

where $F = \forall x(Px \to Pfx)$. In φ_n^* there are n substitution instances of the proof ψ_n and therefore the size of the ACNF-schema φ_n^* is

$$\|\varphi_n^*\| \geq a_1 * n^3.$$

As $C_{\mathrm{EXP}}(\varphi_n) \geq \|\varphi_n^*\|$ (the algorithm EXP contains the construction of φ_n^*) we finally obtain

$$C_{\mathrm{EXP}}(\varphi_n^*) \geq a_1 * n^3.$$

Therefore every expansion tree extraction from φ_n^* via EXP is at least cubic in n. Now we consider the complexity of our improved method for extraction of expansion trees. We construct the projections first, here we have the complexity $O(n^2)$ (just for $\varphi_n[C_{1,n}]$, otherwise

constant). The construction of the refutation δ_n is in $O(n)$. For the construction of the partial expansion proof:

$$\begin{aligned} T(y) = \forall x(Px \to Pfx) \; &+^{y} Py \to Pfy, \\ &+^{fy} Pfy \to Pf^2y, \\ &\cdots \\ &+^{f^{n-1}y} Pf^{n-1}y \to Pf^n y \vdash \end{aligned}$$

we obtain

$$\|T(y)\| = 3(n-1) + 4 = 3n + 1.$$

The last step is the computation of

$$\text{merge}(Pa \vdash, T(y)\{y \leftarrow a\}, T(y)\{y \leftarrow f^n a\}, \ldots, T(y)\{y \leftarrow f^{(n-1)n} a\}, \vdash Pf^{n^2} a)$$

$O(n^2)$: concatenate sequents of complexity $3n + 1$ n-times.

The last sequent is an expansion proof of φ_n^*. The total expense of EXP_{new} is therefore (k being a constant)

$$C_{\text{EXP}_{new}}(\varphi_n) \leq k * (n+1)^2.$$

Now we put things together and obtain that EXP_{new} is never more expensive than EXP, but EXP cannot be linearly bounded in EXP_{new}:

Theorem 5 EXP_{new} *outperforms* EXP*, i.e.*

(a) $C_{\text{EXP}_{new}}(\varphi) \leq C_{\text{EXP}}(\varphi)$ *for all proofs* φ *in* $\mathbf{LK}_{=}$.
(b) There exists no constant d *such that for all proofs* φ *in* $\mathbf{LK}_{=}$*:* $C_{\text{EXP}}(\varphi) \leq d * C_{\text{EXP}_{new}}(\varphi)$.

Proof (a) By Theorem 4. (b): Assume that such a constant d exists. By the construction of the proofs φ_n above we would obtain

$$\begin{aligned} C_{\text{EXP}}(\varphi_n) &\leq d * C_{\text{EXP}_{new}}(\varphi_n) \text{ for all } n \text{ and thus} \\ a_1 * n^3 &\leq d * k * (n+1)^2 \text{ for all } n. \end{aligned}$$

But $a_1 * n^3 > d * k * (n+1)^2$ almost everywhere and we obtain a contradiction. □

Remark 4 We have shown that, for all proofs φ in $\mathbf{LK}_{=}$, $C_{\text{EXP}_{new}}(\varphi) \leq C_{\text{EXP}}(\varphi)$ and that a asymptotic speed-up of C_{EXP} via $C_{\text{EXP}_{new}}$ is possible. The problem to define a sharp bound on C_{EXP} in terms of $C_{\text{EXP}_{new}}$ remains open. Our conjecture is that C_{EXP} cannot be exponential in $C_{\text{EXP}_{new}}$, i.e. that there exists a polynomial p such that

$$C_{\text{EXP}}(\varphi) \leq p(C_{\text{EXP}_{new}}(\varphi)) \text{ for all } \varphi \text{ in } \mathbf{LK}_{=}.$$

In addition to the gain in complexity, another gain is a compact symbolic representation of the expansion tree:

$$\text{merge}(T_1 \Sigma_1, T_2 \Sigma_2, \ldots, T_n \Sigma_n).$$

6 Implementation and Experiments in Gapt

The algorithm EXP_{new} is implemented in the Gapt-system[1] [12], which is a framework for implementing proof transformations written in the programming language Scala. Initially it was developed for the method CERES, but has been extended to other proof transformation algorithms (note that the methods described in this paper are available from version 2.5 on). This section explains how to run the described algorithm, followed by a discussion of results obtained by experiments with formal proofs.

For information on how to install and use the system Gapt we refer to the Gapt User Manual.[2] Gapt opens in a Scala interactive shell (scala>) which can be used to run all the commands provided by the system. Under "Proof Examples" there is a set of functions that generate proofs of some end-sequent. In the following demonstration we will use the proof from Example 5, which is referred to as CERESExpansionExampleProof.proof in Gapt. The sequence of commands

```
scala> val p = CERESExpansionExampleProof.proof
scala> val p1 = CERES( p )
scala> prooftool( p1 )
```

instantiates the proof from Example 5 and stores it in the variable p, which is used as input for the method CERES. The variable p1 stores the generated CERES normal form. Note that the outputs stored in variables p and p1 are strings representing the proofs. To obtain a proof in a tree-like structure prooftool can be used, which is a viewer for proofs and other elements also implemented in Gapt [11]. The algorithm EXP, which extracts expansion proofs from CERES normal forms is implemented in Gapt as the method LKToExpansionProof. Note that this method extracts an expansion proof from any **LK**-proof and not only from CERES normal forms. More information on expansion trees in Gapt can be found in [21,25]. The following demonstration shows how to obtain expansion proofs and Herbrand sequents (corresponding to the deep function of an expansion proof) from the CERES normal form p1 (defined in the demonstration above):

```
scala> val exp = LKToExpansionProof( p1 )
scala> prooftool( exp )
scala> val dp = exp.deep
scala> prooftool( dp )
```

The output stored in exp is a string representing the expansion proof of p1; using prooftool a better representation can be obtained. Note that also the deep function can be displayed in prooftool. The next demonstration shows how to use the algorithm EXP_{new}, which is implemented as the method CERESExpansionProof (within the CERES implementation)

```
scala> val p = CERESExpansionExampleProof.proof
scala> val exp = CERES.CERESExpansionProof( p, Escargot )
scala> val dp = exp.deep
```

The output stored in variable dp is a string representing the deep function of the expansion proof extracted from the proof projections and the corresponding ground PR refutation. Note that we use a simple built-in prover called Escargot. Instead of using Escargot, any other

[1] http://www.logic.at/gapt/.

[2] http://www.logic.at/gapt/downloads/gapt-user-manual.pdf.

resolution prover supported by Gapt may be used (Gapt includes interfaces to several first-order theorem provers, such as Prover9, E Prover and LeanCoP, for more details we refer to the Gapt User Manual).

To measure the complexity of algorithms we use the command time, provided by the Gapt system. This command measures the time in ms that is needed on the system in use to perform a method. We performed several experiments with proofs containing cuts and measured the speed-up in time via

```
scala> time{ LKToExpansionProof( CERES( p ) ) }
scala> time{ CERES.CERESExpansionProof( p ) }
```

Our experiments have shown that there is a speed-up in the computation of expansion proofs with the algorithm EXP_{new} compared to the algorithm EXP already for small and simple proofs like in Example 5: our best result for the algorithm EXP is $47ms$, on the other hand, with the algorithm EXP_{new} we obtained $18ms$. Since even for a small and simple proof like the proof in Example 5 a speed-up is obtained, it is clear that we can increase the speed-up when we consider more complex and longer proofs. Therefore we analyzed more complicated proofs provided by Gapt, as `lattice.proof`, a proof of (one direction of) the equivalence of different definitions of the concept of a lattice. For background information about this proof and an informal version, we refer to Section 5 of [20]. Another interesting proof for analyzing and comparing the implemented methods is `tape.proof`, which is a proof of the statement "Given an infinite tape labelled by zeros and ones there are two cells with the same value.". The proof proceeds by a lemma stating that on such a tape there are either infinitely many zeros or infinitely many ones. This is a subcase of the proof of the unbounded pigeonhole principle, for more information we refer to Section 4.2 of [26] and Section 3 of [1]. Note that the formalization of the tape proof as described in [26] is realized in Gapt as `tabeUrban.proof`. Figure 1 shows our results on experiments with the proofs `lattice.proof`, `tape.proof` and `tapeUrban.proof`; in all three cases, the method EXP_{new} outperforms the method EXP.

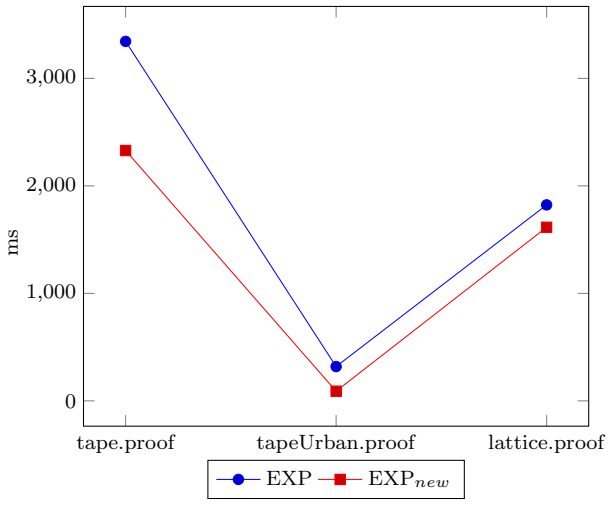

Fig. 1 Comparison of the methods EXP and EXP_{new} for the proofs: `tape.proof`, `tapeUrban.proof`, `lattice.proof`

Table 1 Comparison of the three different methods for the extraction of expansion proofs: based on Gentzen's reductive method, methods EXP and EXP_{new}

Proof	Reductive (ms)	EXP (ms)	EXP_{new} (ms)
fol1.proof	68	27	17
fol2.proof	53	30	21
Pi2Pigeonhole.proof	1350	680	170
Pi3Pigeonhole.proof	842	552	181
poset.proof.cycleImpliesEqual3	1621	360	147
poset.proof.cycleImpliesEqual4	6877	850	250
CutIntroduction(LinearExampleProof(4))	138	70	26
CutIntroduction(LinearExampleProof(8))	294	42	31
CutIntroduction(LinearExampleProof(10))	577	32	28
CutIntroduction(LinearExampleProof(15))	890	36	26
CutIntroduction(LinearExampleProof(18))	1329	54	25
CutIntroduction(LinearExampleProof(19))	1205	85	51
CutIntroduction(LinearEqExampleProof(2))	190	72	32
CutIntroduction(LinearEqExampleProof(5))	924	98	34
CutIntroduction(LinearEqExampleProof(10))	2640	113	43
CutIntroduction(LinearEqExampleProof(15))	6510	134	46
CutIntroduction(LinearEqExampleProof(16))	10,875	163	53
CutIntroduction(LinearEqExampleProof(18))	12,423	455	97
CutIntroduction(FactorialFunctionEqualityExampleProof(3))	3525	500	360
CutIntroduction(FactorialFunctionEqualityExampleProof(4))	8473	795	590
CutIntroduction(FactorialFunctionEqualityExampleProof(5))	20,006	1430	930

Furthermore, we analyzed the two methods based on CERES in comparison to reductive cut-elimination methods. The Gapt system contains an implementation of Gentzen-style reductive cut-elimination, which can be used by calling the method ReductiveCutElimination. For this analysis we used several proofs provided by Gapt, as for instance simple proofs containing cuts (fol1.proof, fol2.proof), formalizations of the so-called poset proof (poset.proof) and of the pigeonhole principle (Pi2Pigeonhole.proof, Pi3Pigeonhole.proof) as well as some "artificial" proofs in the sense that we introduced cuts to originally cut-free proofs. Indeed, Gapt provides a cut-introduction procedure called CutIntroduction(p), which in some cases compresses the cut-free proof p by adding cuts. We use this cut-introduction method on sequences of cut-free proofs, as for instance the sequence of proofs LinearExampleProof(n), constructing cut-free proofs of sequents $P(0), \forall x(P(x) \rightarrow P(s(x))) \vdash P(s^n(0))$, where $n \geq 0$, FactorialFunctionEquality ExampleProof(n), constructing proofs of $f(n) = g(n, 1)$, where f is the head recursive and g the tail recursive formulation of the factorial function and LinearEqExampleProof(n), constructing cut-free proofs of sequents Refl, Trans, $\forall x(f(x) = x) \vdash f^n(a) = a$. Introducing cuts to these three sequences of proofs results in sequences of shorter proofs. Table 1 summarizes our results and shows that not only EXP_{new} is faster that EXP, but both CERES based methods clearly outperform the method based on Gentzen-style cut-elimination.

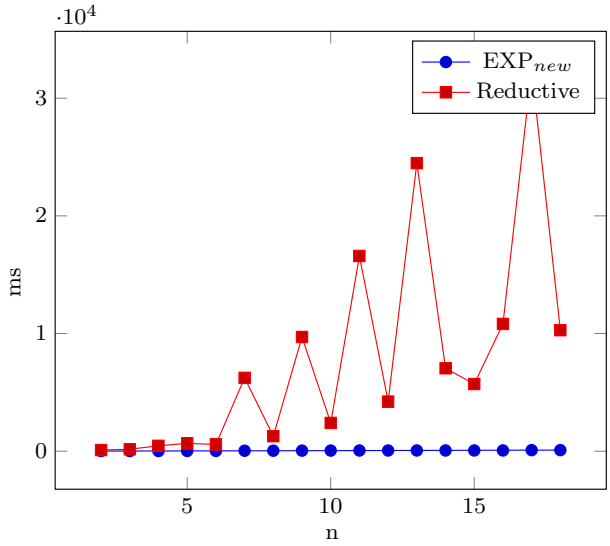

Fig. 2 Comparison of the method EXP_{new} and the method Reductive for the proof CutIntroduction(SquareDiagonalExampleProof(n)) for $2 \leq n \leq 18$

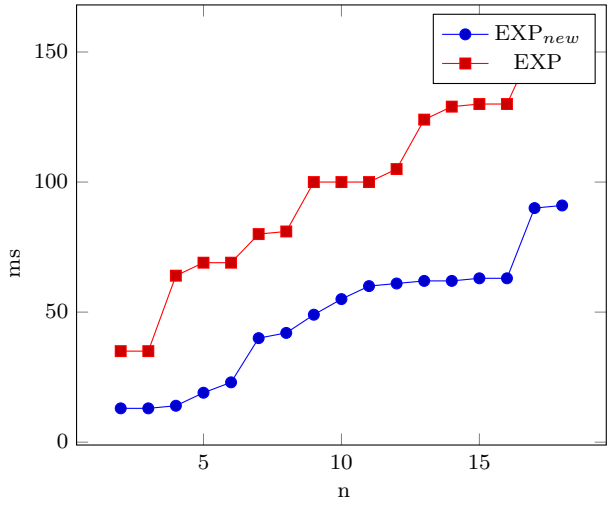

Fig. 3 Comparison of the methods EXP and EXP_{new} for the proof (containing cuts) CutIntroduction(SquareDiagonalExampleProof(n)) for $2 \leq n \leq 18$

Another interesting sequence of cut-free proofs provided by Gapt is formalized in SquareDiagonalExampleProof(n), which constructs a sequence of cut-free proofs of the sequents $P(0,0), \forall x \forall y(P(x,y) \rightarrow P(s(x),y)), \forall x \forall y(P(x,y) \rightarrow P(x,s(y))) \vdash P(s^n(0), s^n(0))$, where $n \geq 0$. For every n the constructed proof goes along the diagonal of P, i.e. one x-step, then one y-step, etc. Cuts can be introduced to this sequence of proofs, however the proof containing cuts obtained by the method CutIntroduction(SquareDiagonal ExampleProof(n)) is not necessarily longer the higher the value for n is. More precisely,

the proof SquareDiagonalExampleProof(n) might not get as much compressed as the proof SquareDiagonalExampleProof(n+1) by introducing cuts, leading to a shorter proof for $n+1$. Therefore, the method based on reductive cut-elimination is not always slower for higher values of n. In fact, as can be observed in Fig. 2, the computing time for the method based on reductive cut-elimination on the proof CutIntroduction(SquareDiagonalExampleProof(7)) is $6235ms$, while on the proof CutIntroduction(SquareDiagonalExampleProof(8)) it is $1278ms$. The analysis in Fig. 2 is performed for values $2 \leq n \leq 18$, as for higher values the constructed proof is too long to be analyzed on the operating system in use. Note that also for SquareDiagonalExampleProof(n) the method EXP_{new} outperforms the method EXP, as can be seen in Fig. 3.

We want to remark that the obtained results depend to a great degree on the operating system in use. Therefore, it is possible that the obtained results fluctuate. Nevertheless, the speed-up of the method EXP_{new} compared to EXP is given and can be clearly recognized.

7 Conclusion

In the analysis of mathematical proofs it is usually more important to gain essential mathematical information from proofs (which typically lies in the terms), than to traverse complicated and long propositional proofs. This was our motivation for avoiding the construction of an atomic cut normal form and focus on the quantifier inferences in the final proof (represented by a Herbrand sequent or by an expansion proof). In the original CERES method, the extraction of expansion proofs is performed after the final result of CERES (a proof containing at most atomic cuts, the CERES normal form) is obtained. We first analysed ACNFs and defined a new version of ACNF where the cut-inferences are the last inferences in the proof and only contractions and weakenings occur between them. Proofs of that structure are said to be in top normal form. We have shown that the logical parts of proofs in top normal form suffice to compute expansion proofs. Since proofs in CERES normal form are in top normal form, its logical parts (the proof-projections) suffice to extract expansion proofs. First we refute the characteristic clause set $\text{CL}(\varphi)$ and, from a corresponding ground refutation R, obtain a set of ground substitutions $\Sigma(C)$ for all clauses $C \in \text{CL}(\varphi)$ that occur in R. We construct the general CERES projections $\varphi[C]$, corresponding to the clauses $C \in \text{CL}(\varphi)$ that occur in R, and extract partial expansion proofs $T[C]$. Finally we instantiate the partial expansion proofs $T[C]$ by substitutions in $\Sigma(C)$ and combine the resulting expansion proofs by merging; the result is an expansion proof of the end-sequent (which coincides with the expansion proof of the CERES normal form). Using this method we avoided the construction of an atomic cut normal form.

We also obtain an improvement in asymptotic complexity. Note that we do not compute n instances of the projection $\varphi[C_i]$ with the corresponding substitution σ_j^i like it is needed for the construction of the CERES normal form, instead we compute the projection once, remove clause parts and then instantiate the corresponding expansion proof (Herbrand sequent) with the corresponding substitutions. The described algorithm is implemented in the Gapt system and we describe how to use it. We show that even for small and not complicated proofs a speed-up in time is obtained by the new algorithm.

Since we investigated this method for first-order logic only, further work has to deal with a generalization of this method to higher-order logic. In higher-order logic we have to deal with a different CERES-method (CERES^{ω} [19]) and a resolution calculus for higher-order logic. The extraction of expansion proofs from CERES-projections in the higher-order case is

a non-trivial task, since CERES$^{\omega}$ as well as the resolution calculus for higher-order logic are much more complicated than in the first-order case. If and how the method can be generalized to the higher-order case, future work will tell.

Acknowledgements Open access funding provided by Austrian Science Fund (FWF).

Funding Funding was provided by Austrian Science Fund (Grant No. I-2671-N35).

References

1. Baaz, M., Hetzl, S., Leitsch, A., Richter, C., Spohr, H.: Cut-elimination: experiments with ceres. In: International Conference on Logic for Programming Artificial Intelligence and Reasoning, pp. 481–495. Springer (2005)
2. Baaz, M., Hetzl, S., Leitsch, A., Richter, C., Spohr, H.: Proof transformation by CERES. In: Proceedingsof 5th International Conference on Mathematical Knowledge Management, MKM 2006, Wokingham, UK, August 11–12, 2006, pp. 82–93 (2006)
3. Baaz, M., Leitsch, A.: On skolemization and proof complexity. Fundam. Inf. **20**(4), 353–379 (1994)
4. Baaz, M., Leitsch, A.: Cut-elimination and redundancy-elimination by resolution. J. Symb. Comput. **29**(2), 149–177 (2000)
5. Baaz, M., Leitsch, A.: Towards a clausal analysis of cut-elimination. J. Symb. Comput. **41**(3), 381–410 (2006)
6. Baaz, M., Leitsch, A.: Methods of Cut-Elimination, vol. 34. Springer, Berlin (2011)
7. Baaz, M., Leitsch, A.: Cut-elimination: syntax and semantics. Stud. Log. **102**(6), 1217–1244 (2014)
8. Berger, U., Berghofer, S., Letouzey, P., Schwichtenberg, H.: Program extraction from normalization proofs. Stud. Log. **82**(1), 25–49 (2006)
9. Berger, U., Buchholz, W., Schwichtenberg, H.: Refined program extraction from classical proofs. Ann. Pure Appl. Log. **114**(1), 3–25 (2002)
10. Buss, S.R.: On Herbrand's theorem. In: Leivant, D. (ed.) Logical and Computational Complexity. Selected Papers. Logic and Computational Complexity, International Workshop LCC '94, Indianapolis, Indiana, USA, 13–16 October 1994. Lecture Notes in Computer Science, vol. 960, pp. 195–209. Springer (1994)
11. Dunchev, C., Leitsch, A., Libal, T., Riener, M., Rukhaia, M., Weller, D., Woltzenlogel-Paleo, B.: Prooftool: a gui for the gapt framework. arXiv preprint arXiv:1307.1942 (2013)
12. Ebner, G., Hetzl, S., Reis, G., Riener, M., Wolfsteiner, S., Zivota, S.: System description: Gapt 2.0. In: International Joint Conference on Automated Reasoning, pp. 293–301. Springer (2016)
13. Fürstenberg, H.: On the infinitude of the primes. Am. Math. Mon. **62**, 353 (1955)
14. Fürstenberg, H.: Topological dynamics and combinatorial number theory. J. Anal. Math. **34**(1), 61–85 (1978)
15. Gentzen, G.: Untersuchungen über das logische Schließen. Math. Z. **39**, 176–210, 405–431 (1934–35)
16. Girard, J.Y.: Proof Theory and Logical Complexity, vol. 1. Biblopolis, Napoli (1987)
17. Gödel, K.: Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. Dialectica **12**(3–4), 280–287 (1958)
18. Herbrand, J.: Recherches sur la théorie de la démonstration. PhD thesis, Université de Paris (1930)
19. Hetzl, S., Leitsch, A., Weller, D.: Ceres in higher-order logic. Ann. Pure Appl. Log. **162**(12), 1001–1034 (2011)
20. Hetzl, S., Leitsch, A., Weller, D., Paleo, B.W.: Herbrand sequent extraction. In: Autexier, S., Campbell, J.A., Rubio, J., Sorge, V., Suzuki, M., Wiedijk, F. (eds.) Proceedings of Intelligent Computer Mathematics, 9th International Conference, AISC 2008, 15th Symposium, Calculemus 2008, 7th International Conference, MKM 2008, Birmingham, UK, 28 July-1 August 2008. Lecture Notes in Computer Science, vol. 5144, pp. 462–477. Springer (2008)
21. Hetzl, S., Libal, T., Riener, M., Rukhaia, M.: Understanding resolution proofs through Herbrands theorem. In: Galmiche, D., Larchey-Wendling, D. (eds.) Proceedings of Automated Reasoning with Analytic

Tableaux and Related Methods - 22nd International Conference, TABLEAUX 2013, Nancy, France, 16–19 September 2013. Lecture Notes in Computer Science, vol. 8123, pp. 157–171. Springer (2013)
22. Hetzl, S., Weller, D.: Expansion trees with cut. arXiv preprint arXiv:1308.0428 (2013)
23. Miller, D.A.: A compact representation of proofs. Stud. Log. **46**(4), 347–370 (1987)
24. Pfenning, F.: Analytic and non-analytic proofs. In: 7th International Conference on Automated Deduction, pp. 394–413. Springer (1984)
25. Reis, G.: Importing SMT and connection proofs as expansion trees. arXiv preprint arXiv:1507.08715 (2015)
26. Urban, C.: Classical logic and computation. PhD thesis, University of Cambridge (2000)
27. Van der Waerden, B.L.: Beweis einer Baudetschen Vermutung. Nieuw Arch. Wiskd. **15**(2), 212–216 (1927)