# Extracting expansion trees from resolution proofs with splitting and definitions[⋆]

Gabriel Ebner

TU Wien

**Abstract.** We present a new and efficient algorithm to translate proofs generated by resolution-based automated theorem provers into expansion proofs—a formalism for Herbrand disjunctions for non-prenex formulas. In contrast to previous approaches, this algorithm supports definition-introducing structural clausification and Avatar-style splitting inferences.

## 1 Introduction

Herbrand's theorem [7,3] captures the fundamental insight in logic that the validity of a quantified formula is characterized by the existence of a tautological finite set of quantifier-free instances. In its simplest case, the validity of a purely existential formula is characterized by the existence of a tautological disjunction of instances, a Herbrand disjunction.

Expansion trees were introduced in [14] to generalize this result to higher-order logic in the form of simple type theory. In first-order logic, they provide a technically convenient formalism to store these tautological instances of non-prenex formulas.

The fundamental importance of Herbrand's theorem is underlined by the variety of its applications. Herbrand disjunctions directly contain the answer substitutions of logic programs. Luckhardt [13] used them to give a polynomial bound for Roth's theorem. Grammar-based compression of expansion proofs has been used to introduce non-analytic quantified lemmas [10,9,8,5], and prove theorems with induction in [4].

Particularly for the last two applications it is vitally important to efficiently generate expansion proofs using automated theorem provers. Algorithms to translate automatically generated resolution proofs to expansion proofs have already been proposed in the context of the TPS [16] and GAPT [11] systems.

The approach in [11] transforms a resolution refutation first into a sequent calculus proof with atomic cuts, and then extracts an expansion proof in a second step; Skolemization is not handled in the algorithm and is instead treated as a preprocessing step of the formula.

More closely related to the approach in this paper is the algorithm in [16]: it directly transforms a resolution refutation into an expansion proof. Skolemization is handled explicitly during the translation, and Skolem terms are automatically converted to variables. However this treatment has two unfortunate

⋆ Supported by the Vienna Science and Technology Fund (WWTF) project VRG12-004

consequences: first, only outer Skolemization can be used in this way—and this conflicts with the goal to reduce the number of Skolem functions and the number of the arguments, which is necessary for efficient proof search. Additionally, the deskolemization is incompatible with built-in equational inferences which implicitly make use of congruences for Skolem functions.

Both of the approaches require a naive distributivity-based transformation of the input formula into clause normal form (CNF), which can increase the size of the problem exponentially. Moreover, neither approach can handle splitting inferences as used by SPASS [21] or Vampire [20].

The solution we present in this paper includes clausification and splitting steps as inferences of the resolution proof in a similar way as higher-order resolution calculi [1], conveniently keeping track of introduced subformula definitions for structural clausification and Skolem functions in the same data structure as the resolution refutation. Concretely, we present the following:

- We describe an algorithm that directly transforms resolution proofs with integrated clausification inferences into expansion proofs, supporting inner Skolemization and also Skolemization in the presence of equality rules.
- We show how to support Avatar-style splitting inferences.
- We show how to handle subformula definitions in this transformation.

In Sections 3 and 4 we describe the resolution and expansion proof calculi used, respectively. Basic operations on expansion proofs, merge- and cut-reduction, are introduced in Section 4 as well. Section 5 then explains the extraction of expansion proofs from resolution proofs. The resulting expansion proofs may still contain definitions if the resolution proof contains subformula definitions. These definitions are then eliminated in Section 6. Finally in Section 8 we evaluate the performance and clausification quality of an implementation of this algorithm.

## 2 Preliminaries

We work in first-order logic with equality. A quantifier-free formula is called quasi-tautological iff it is valid in first-order logic with equality—that is, if it becomes a tautology by adding congruence and reflexivity axioms for the $=$ relation. In the terminology of SMT solvers, quasi-tautologies are the `QF_UF`-valid formulas.

A *sequent* $\Gamma \vdash \Delta$ is a pair of two multisets, the antecedent $\Gamma$ and the succedent $\Delta$. Unless stated otherwise sequents contain formulas, but we will also consider sequents containing other objects. Clauses are sequents of atoms. We usually use the names $a, b, c$ for constants, $x, y, z$ for variables, $t, s, r$ for terms, $\varphi, \psi$ for formulas, $A, B$ for atoms, $E$ for expansion trees, and $\mathcal{S}$ for sequents. We define composition of sequents as $(\Gamma \vdash \Delta)\,(\Pi \vdash \Lambda) = (\Gamma \cup \Pi \vdash \Delta \cup \Lambda)$, prepending an element to the antecedent as $\varphi +\!: (\Gamma \vdash \Delta) = (\Gamma \cup \{\varphi\} \vdash \Delta)$, appending an element to the succedent as $(\Gamma \vdash \Delta) :\!+\, \varphi = (\Gamma \vdash \Delta \cup \{\varphi\})$, and concatenating sequents as $(\Gamma \vdash \Delta) +\!+\, (\Pi \vdash \Lambda) = (\Gamma \cup \Pi) \vdash (\Delta \cup \Lambda)$.

Given a formula or term $e$, the set $\mathrm{FV}(e)$ consists of the free variables of $e$. For sequences of terms $e_1, \ldots, e_n$ we use the vector notation $\overline{e}$, this notation is also used for blocks of quantifiers, i.e. $\forall \overline{x}\, \varphi$ means $\forall x_1 \cdots \forall x_n\, \varphi$. Substitutions are written as $[x \backslash t]$ or $[\overline{x} \backslash \overline{t}]$, meaning a capture-avoiding parallel substitution of the variables $\overline{x}$ by the terms $\overline{t}$.

## 3 Resolution proofs

As a calculus for resolution proofs we consider a first-order variant of Andrew's system $\mathcal{R}$ [1], extended with inferences to support subformula definitions and Avatar splitting. The inferences in this calculus operate on sequents. Let $\varphi$ be the closed formula that is to be proven, a resolution proof then starts with the sequent $\varphi \vdash$ (using the Input inference) and ends with the empty sequent $\vdash$.

Resolution proofs can be roughly divided into an upper part starting with $\varphi \vdash$ that performs clausification inferences, and a lower part that ends with $\vdash$ and represents the actual proof search with resolution, rewriting, factoring, and other inferences. There is no implicit unification in the inferences, substitution is handled explicitly using an extra inference.

Both Skolemization and subformula definition inferences make use of global dictionaries that store the interpretation of the defined atoms and Skolem functions, respectively. Defined atoms and Skolem functions are fresh symbols that do not occur in the problem. For definitions, the entry $D(\overline{x}) \overset{\mathrm{def}}{\mapsto} \psi$ means that the atom $D(\overline{x})$ is defined to by $\psi$, that is, $\forall \overline{x}\, (D(\overline{x}) \leftrightarrow \psi)$ is true. For Skolem functions, the entry $s(\overline{x}) \overset{\mathrm{sk}}{\mapsto} \forall y\, \varphi[y]$ means that $s(\overline{x})$ is the Skolem term used to instantiate $\forall y\, \varphi[y]$, that is, $\forall \overline{x}\, (\varphi[s(\overline{x})] \to \forall y\, \varphi[y])$ is true. We use both relations also for substitution instances of the definitions, i.e. we write $D(\overline{t}) \overset{\mathrm{def}}{\mapsto} \psi[\overline{x} \backslash \overline{t}]$ as well as $s(\overline{t}) \overset{\mathrm{sk}}{\mapsto} (\forall y\, \varphi[y])[\overline{x} \backslash \overline{t}]$. Furthermore, we require these definitions to be acyclic.

For reasons of space, we only show inferences for the connectives $\top, \neg, \wedge, \forall$ here. This is not a restriction since the other connectives can be defined in terms of these. We will use the analogous rules for the other connectives in examples.

*Clausification inferences:*

$$\frac{}{\varphi \vdash}\,\mathrm{Input} \qquad \frac{\top, \Gamma \vdash \Delta}{\Gamma \vdash \Delta}\,\mathrm{TopL} \qquad \frac{\neg\varphi, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, \varphi}\,\mathrm{NegL} \qquad \frac{\Gamma \vdash \Delta, \neg\varphi}{\varphi, \Gamma \vdash \Delta}\,\mathrm{NegR}$$

$$\frac{\varphi \wedge \psi, \Gamma \vdash \Delta}{\varphi, \psi, \Gamma \vdash \Delta}\,\mathrm{AndL} \qquad \frac{\Gamma \vdash \Delta, \varphi \wedge \psi}{\Gamma \vdash \Delta, \varphi}\,\mathrm{AndR1} \qquad \frac{\Gamma \vdash \Delta, \varphi \wedge \psi}{\Gamma \vdash \Delta, \psi}\,\mathrm{AndR2}$$

$$\frac{\Gamma \vdash \Delta, \forall x\, \varphi}{\Gamma \vdash \Delta, \varphi}\,\mathrm{AllR} \qquad \frac{\forall x\, \varphi, \Gamma \vdash \Delta}{\varphi[x \backslash t], \Gamma \vdash \Delta}\,\mathrm{AllL}\ (\text{where } t \overset{\mathrm{sk}}{\mapsto} \forall x\, \varphi)$$

*Subformula definition inferences:*

$$\frac{\Gamma \vdash \Delta, \varphi}{\Gamma \vdash \Delta, D(\overline{t})}\,\mathrm{AbbrL} \qquad \frac{}{D(\overline{t}) \vdash \varphi}\,\mathrm{DefR} \qquad (\text{where } D(\overline{t}) \overset{\mathrm{def}}{\mapsto} \varphi)$$

*Logical inferences:*

$$\frac{\Gamma \vdash \Delta, A \qquad A, \Pi \vdash \Lambda}{\Gamma, \Pi \vdash \Delta, \Lambda}\ \text{Res} \qquad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}\ \text{Factor} \qquad \frac{\Gamma \vdash \Delta}{\Gamma\sigma \vdash \Delta\sigma}\ \text{Subst}$$

$$\frac{}{A \vdash A}\ \text{Taut} \qquad \frac{\Gamma \vdash \Delta, t = s \qquad A[t], \Pi \vdash \Lambda}{A[s], \Gamma, \Pi \vdash \Delta, \Lambda}\ \text{Rw} \qquad \frac{}{\vdash t = t}\ \text{Refl}$$

*Avatar splitting inference:*

$$\frac{\mathcal{S}_1 +\!\!+ \mathcal{S}_2}{\mathcal{S}_1 :\!+ D}\ \text{AvSplit} \quad (\text{where } D \overset{\text{def}}{\mapsto} \forall\overline{x}\, \mathcal{S}_2 \text{ and } \mathrm{FV}(\mathcal{S}_1) \cap \mathrm{FV}(\mathcal{S}_2) = \emptyset)$$

The AvSplit inference here is the minimal version necessary to represent Avatar splitting. The calculus in [20] uses A-clauses, where an A-clause $\neg a_1 \vee \cdots \vee \neg a_m \vee b_1 \vee \cdots \vee b_n \leftarrow C_1, \ldots, C_k$ is a pair of a clause and an assertion, which is a finite set of clauses called "components". Each clause $C$ that occurs in an assertion has an associated propositional atom $[C]$. In the calculus here, we represent A-clauses by combining the assertion and the clause part into a single clause $[C_1], \ldots, [C_k], a_1, \ldots, a_n \vdash b_1, \ldots b_n$. The distinction between the assertion and non-assertion part of the clauses is important for proof search, but is not relevant to the transformation here.

During proof search, splitting takes a clause $\mathcal{S}_1 +\!\!+ \mathcal{S}_2$ that can be partitioned into two (or more) clauses $\mathcal{S}_1$ and $\mathcal{S}_2$ with pairwise disjoint free variables, and splits it into three clauses: $D_1 +\!: \mathcal{S}_1$ and $D_2 +\!: \mathcal{S}_2$ plus an additional clause $\vdash D_1, D_2$ that gets sent to the SAT solver. In the calculus here we represent this step as follows:

$$\frac{\dfrac{\mathcal{S}_1 +\!\!+ \mathcal{S}_2}{\mathcal{S}_1 :\!+ D_2}\ \text{AvSplit}}{\vdash D_1, D_2}\ \text{AvSplit} \qquad\qquad \frac{\dfrac{}{D_i \vdash \forall\overline{x}\, \mathcal{S}_i}\ \text{DefR}}{D_i +\!: \mathcal{S}_i}\ \text{AllR, OrR, NegR}$$

Other splitting schemes such as the one implemented in SPASS [21] can be simulated using these inferences as well. The splitting dependencies a clause implicitly depends on are simply translated to explicit splitting atoms. For each split we then take the resulting proofs of the two branches and resolve on the opposing splitting atoms.

We assume a few minor technical restrictions on resolution proofs: there are no AbbrL inferences below Rw, Refl, and Taut, and DefR with the same defined atom. It is also important to note that we only allow subformula definitions in a single polarity here—if we want to abbreviate a formula $\varphi$ that occurs on both sequent sides, then we need to introduce a different $D_i$ atom for each side.

*Example 1 (Running example).* The following is natural proof of a variant of the Drinker's formula. For simplicity, this proof does not make use of Avatar or definition inferences. We label the subproofs on the left-hand side for later reference.

$(\pi_0)$ $\exists x \forall y (P(x) \to P(y)) \vdash$ (Input)
$(\pi_1)$ $\forall y (P(x) \to P(y)) \vdash$ (ExL($\pi_0$))
$(\pi_2)$ $P(x) \to P(s(x)) \vdash$ (AllL($\pi_1$))
$(\pi_3)$ $\vdash P(x)$ (ImpL($\pi_2$))
$(\pi_4)$ $P(s(x)) \vdash$ (ImpR($\pi_2$))
$(\pi_5)$ $\vdash P(s(x))$ (Subst($\pi_3$))
$(\pi_6)$ $\vdash$ (Res($\pi_4$, $\pi_5$))

## 4 Expansion proofs

The proof formalism of expansion trees was introduced in [14] to describe Herbrand disjunctions in higher-order logic. We use them in first-order logic as well, since they are an elegant and convenient data structure. The central idea is that each expansion tree $E$ comes with a *shallow formula* $\mathrm{sh}(E)$ and a quantifier-free *deep formula* $\mathrm{dp}(E)$. The deep formula corresponds to the Herbrand disjunction, the shallow formula is the quantified formula. If the deep formula is a quasi-tautology (a tautology modulo equality), then the shallow formula is valid.

Expansion trees have two polarities: consider the formula $(\forall x\, \varphi) \land \neg(\forall x\, \varphi)$. In this example, the first quantifier has positive polarity, and the second one negative polarity. This distinction is important since we must instantiate the first one with a Skolem term, while we can instantiate the second one with whatever terms we want.

We base the development on [12], and will only review the necessary results without repeating the proofs. We include several extensions here: there are subformula definition nodes to represent the AbbrL inferences in the resolution calculus. We also add cut nodes as in [12], these cuts are similar to cuts in a sequent calculus and will be used to represent Avatar splitting inferences. We also add Skolem nodes to represent the Skolemization steps.

**Definition 1.** *We inductively define the set* $\mathrm{ET}^p(\varphi)$ *of* expansion trees *with polarity* $p \in \{+,-\}$ *and* shallow formula $\varphi$*:*

$$\frac{\varphi \text{ formula}}{\mathrm{wk}^p(\varphi) \in \mathrm{ET}^p(\varphi)} \qquad \frac{E_1 \in \mathrm{ET}^p(\varphi) \qquad E_2 \in \mathrm{ET}^p(\varphi)}{E_1 \sqcup E_2 \in \mathrm{ET}^p(\varphi)}$$

$$\frac{A \text{ atom}/\top}{A^p \in \mathrm{ET}^p(A)} \qquad \frac{E \in \mathrm{ET}^p(\varphi)}{\neg E \in \mathrm{ET}^{-p}(\neg\varphi)} \qquad \frac{E_1 \in \mathrm{ET}^p(\varphi) \qquad E_2 \in \mathrm{ET}^p(\psi)}{E_1 \land E_2 \in \mathrm{ET}^p(\varphi \land \psi)}$$

$$\frac{D(\bar{t}) \overset{\mathrm{def}}{\mapsto} \varphi}{\varphi +_{\mathrm{def}} D^p(\bar{t}) \;\in\; \mathrm{ET}^p(\varphi)} \qquad \frac{E_1 \in \mathrm{ET}^+(\varphi) \qquad E_2 \in \mathrm{ET}^-(\varphi)}{\mathrm{Cut}(E_1, E_2) \in \mathrm{ET}^-(\top)}$$

$$\frac{E \in \mathrm{ET}^+(\varphi[x\backslash y])}{\forall x\, \varphi +^y_{\mathrm{ev}} E \;\in\; \mathrm{ET}^+(\forall x\, \varphi)} \qquad \frac{E \in \mathrm{ET}^+(\varphi[x\backslash t]) \qquad t \overset{\mathrm{sk}}{\mapsto} \forall x\, \varphi}{\forall x\, \varphi +^t_{\mathrm{sk}} E \;\in\; \mathrm{ET}^+(\forall x\, \varphi)}$$

$$\frac{E_1 \in \mathrm{ET}^-(\varphi[x\backslash t_1]) \qquad \cdots \qquad E_n \in \mathrm{ET}^-(\varphi[x\backslash t_n])}{\forall x\, \varphi +^{t_1} E_1 \cdots +^{t_n} E_n \;\in\; \mathrm{ET}^-(\forall x\, \varphi)}$$

The tree $\mathrm{wk}^p(\varphi)$ is called "weakening", $E_1 \sqcup E_2$ is called merge, $\forall x\, \varphi +^y_{\mathrm{ev}} E \in \mathrm{ET}^+(\forall x\, \varphi)$ is an eigenvariable node, and $\forall x\, \varphi +^t_{\mathrm{sk}} E \;\in\; \mathrm{ET}^+(\forall x\, \varphi)$ is a Skolem node. Each expansion tree $E$ has a uniquely determined shallow formula and polarity, we write $\mathrm{sh}(E)$ for its shallow formula, and $\mathrm{pol}(E)$ for its polarity. The *size* $|E|$ of an expansion tree is the number of its leaves counted as in a tree. Given an expansion tree $E = \forall x \varphi +^{y}_{\mathrm{ev}} E'$, we say that $y$ is the eigenvariable of $E$. The set $\mathrm{EV}(E)$ contains all eigenvariables of subtrees in $E$, including $E$. We also use the notation for blocks of quantifiers with expansion trees, that is, we write $\forall \overline{x}\, \varphi +^{\overline{x}}_{\mathrm{ev}} E$ as an abbreviation for $\forall \overline{x}\, \varphi +^{x_1}_{\mathrm{ev}} \cdots +^{x_n}_{\mathrm{ev}} E$.

*Example 2.* The following expansion tree $E \in \mathrm{ET}^{+}(\exists x \forall y\, (P(x) \to P(y)))$ has our variant of the Drinker's formula as shallow formula:

$$\begin{aligned}
&\exists x \forall y\, (P(x) \to P(y))\\
&+^{x} \forall y\, (P(x) \to P(y)) +^{s(x)}_{\mathrm{sk}} (\mathrm{wk}^{-}(P(x)) \to P(s(x))^{+})\\
&+^{s(x)} \forall y\, (P(s(x)) \to P(y)) +^{s(s(x))}_{\mathrm{sk}} (P(s(x))^{-} \to \mathrm{wk}^{+}(P(s(s(x)))))
\end{aligned}$$

While the shallow formula describes the quantified formula to be proven, the deep formula is a quantifier-free formula corresponding to the Herbrand disjunction:

**Definition 2.** *Let $E$ be an expansion tree, we define the* deep formula $\mathrm{dp}(E)$ *recursively as follows:*

$$\mathrm{dp}(\mathrm{wk}^{+}(\varphi)) = \bot, \qquad \mathrm{dp}(\mathrm{wk}^{-}(\varphi)) = \top$$

$$\mathrm{dp}(E_1 \sqcup E_2) = \mathrm{dp}(E_1) \vee \mathrm{dp}(E_2) \quad \textit{if } \mathrm{pol}(E_1 \sqcup E_2) = +$$

$$\mathrm{dp}(E_1 \sqcup E_2) = \mathrm{dp}(E_1) \wedge \mathrm{dp}(E_2) \quad \textit{if } \mathrm{pol}(E_1 \sqcup E_2) = -$$

$$\mathrm{dp}(A^{p}) = A, \quad \mathrm{dp}(\neg E) = \neg \mathrm{dp}(E), \quad \mathrm{dp}(E_1 \wedge E_2) = \mathrm{dp}(E_1) \wedge \mathrm{dp}(E_2)$$

$$\mathrm{dp}(\varphi +_{\mathrm{def}} D^{p}(\bar{t})) = D^{p}(\bar{t}), \quad \mathrm{dp}(\forall x\, \varphi +^{y}_{\mathrm{ev}} E) = \mathrm{dp}(E), \quad \mathrm{dp}(\forall x\, \varphi +^{t}_{\mathrm{sk}} E) = \mathrm{dp}(E)$$

$$\mathrm{dp}(\forall x\, \varphi +^{t_1} E_1 \cdots +^{t_n} E_n) = \mathrm{dp}(E_1) \wedge \cdots \wedge \mathrm{dp}(E_n)$$

*Example 3.* The expansion tree $E$ in Example 2 has the following tautological deep formula: $\mathrm{dp}(E) = (\top \to P(s(x))) \vee (P(s(x)) \to \bot)$

Eigenvariable nodes add a small technical complication to the definition of an expansion proof. (But we need them since they arise during the extraction of splitting inferences.) Not only is it necessary that the deep formula is quasi-tautological, but the eigenvariables also need to be acyclic in a certain sense (this is a similar restriction to the eigenvariable condition in sequent calculi). We formalize this acyclicity using a dependency relation, which we will require to be acyclic:

**Definition 3.** *Let $E$ be an expansion tree. The dependency relation $<_E$ is a binary relation on variables where $x <_E y$ iff $E$ contains a subtree $E'$ such that $x \in \mathrm{FV}(\mathrm{sh}(E'))$ and $y \in \mathrm{EV}(E')$.*

**Definition 4.** *An* expansion sequent $\mathcal{E}$ *is a sequent of expansion trees. Its dependency relation* $<_\mathcal{E} = \bigcup_{E\in\mathcal{E}} <_E$ *is the union of the dependency relations of its trees. Its shallow sequent and deep sequent consists of the shallow and deep formulas of its expansion trees, respectively.*

*An expansion sequent is* positive(negative) *iff the trees in the succedent have positive(negative) polarity and the trees in the antecedent have negative(positive) polarity. The* size $|\mathcal{E}|$ *of an expansion sequent is the sum of the sizes of its expansion trees.*

**Definition 5.** *An* expansion proof $\mathcal{E}$ *is a positive expansion sequent such that:*

1. $<_\mathcal{E}$ *is acyclic (i.e., can be extended to a linear order) and there are no duplicate eigenvariables,*
2. $\mathrm{dp}(\mathcal{E})$ *is a quasi-tautology*

*Example 4.* Let $E$ be as in Example 2. The sequent $\mathrm{dp}(\vdash E)$ is a tautology, and the dependency relation of $\vdash E$ is empty and hence acyclic. So $\vdash E$ is an expansion proof.

Expansion proofs are sound and complete for first-order logic. That is, if $\mathcal{S}$ is a sequent, then there exists an expansion proof $\mathcal{E}$ with $\mathcal{S}$ as the shallow sequent if and only if $\mathcal{S}$ is valid. Given a substitution $\sigma$ that maps eigenvariables to variables, we can apply it to expansion trees and expansion sequents, written $E\sigma$ and $\mathcal{E}\sigma$, respectively.

The calculus for expansion proofs we presented is redundant: it is still complete even without the $\sqcup$, $+_{\mathrm{def}}$, and Cut nodes. We also only need either the eigenvariable or Skolem nodes. We will now review two reductions from [12] that will eliminate the $\sqcup$ and Cut nodes. The elimination of $+_{\mathrm{def}}$ will be discussed in Section 6.

The relation $\overset{\sqcup}{\rightsquigarrow}$ pushes merge nodes to the leaves of an expansion tree until they disappear, for example $(E_1 \land E_2) \sqcup (E_3 \land E_4) \overset{\sqcup}{\rightsquigarrow} (E_1 \sqcup E_3) \land (E_2 \sqcup E_4)$ reduces merges on conjunctions.

**Lemma 1.** *The relation* $\overset{\sqcup}{\rightsquigarrow}$ *on expansion sequents has the following properties:*

1. $\overset{\sqcup}{\rightsquigarrow}$ *is terminating.*
2. *Whenever* $\mathcal{E} \overset{\sqcup}{\rightsquigarrow} \mathcal{E}'$*, then* $\mathrm{dp}(\mathcal{E}) \to \mathrm{dp}(\mathcal{E}')$ *is a tautology.*
3. $\overset{\sqcup}{\rightsquigarrow}$ *preserves acyclicity of the dependency relation.*
4. $\overset{\sqcup}{\rightsquigarrow}$ *preserves polarity and the shallow formula.*
5. *If* $\mathcal{E}$ *does not contain Skolem or definition nodes, then its* $\overset{\sqcup}{\rightsquigarrow}$*-normal forms do not contain merge nodes.*

*In particular the* $\overset{\sqcup}{\rightsquigarrow}$*-normal forms of expansion proofs without Skolem or definition nodes are merge-free expansion proofs.*

*Proof.* Analogous to Lemmas 12 and 13 in [12]. Note that merge reduction can get stuck on merges of two Skolem nodes with different Skolem terms, or two definition nodes with different definitions, hence the condition in point 5.

Cuts can be reduced and eventually eliminated as well. The cut-reduction relation $\overset{\text{cut}}{\rightsquigarrow}$ (written as $\mapsto$ in [12]) extends merge-reduction and reduces quantified cuts via substitution.

**Lemma 2.** *$\overset{\text{cut}}{\rightsquigarrow}$ preserves quasi-tautology of the deep formula, and is weakly normalizing. If no definition or Skolem nodes appear in cuts, then the $\overset{\text{cut}}{\rightsquigarrow}$-normal forms are cut-free.*

*Proof.* See Lemma 16 and Theorem 33 in [12]. Just as in Lemma 1, cut-reduction can get stuck on cuts with Skolem nodes such as $\mathrm{Cut}(\forall x\, \varphi +^{x}_{\mathrm{sk}} \dots, \forall x\, \varphi +^{t} \dots)$.

The following lemma will bound the complexity introduced by Avatar splitting inferences, since they will be translated to cuts on formulas of the form $\forall \overline{x}\, C(\overline{x})$ where $C(\overline{x})$ is a clause.

**Lemma 3.** *Let $|\mathcal{E}|$ be an expansion proof with $n$ cuts such that all cut formulas are universally quantified closed prenex formulas, and let $\mathcal{E} \overset{\text{cut}^*}{\rightsquigarrow} \mathcal{E}^*$ such that $\mathcal{E}^*$ is cut-free. Then $|\mathcal{E}^*| \leq |\mathcal{E}|^{2^n}$.*

*Proof.* Merge reduction and propositional reduction steps reduce the size of the expansion proof and keep the number of cuts constants. Each quantifier reduction step decreases the number of cuts by one and increases the size of the proof at most quadratically: essentially we duplicate the proof $m$ times, where $m$ is the number of weak quantifier term blocks in the cut—and $m$ is less than the size of the expansion proof.

## 5 Extraction

In this section we convert a resolution proof of a formula $\varphi$ into an expansion proof with a shallow sequent of the following form:

$$\forall \overline{x}\, (\varphi_1(\overline{x}) \to D_1(\overline{x})), \cdots, \forall \overline{x}\, (\varphi_n(\overline{x}) \to D_n(\overline{x})) \vdash \varphi$$

That is, the expansions in the antecedent describe subformula definitions and have definition axioms $\forall \overline{x}\, (\varphi(\overline{x}) \to D(\overline{x}))$ as shallow formulas where $D(\overline{x}) \overset{\text{def}}{\mapsto} \varphi(\overline{x})$. Furthermore the resulting expansion proof will not have any eigenvariable nodes. We will eliminate the additional expansion trees for definitions in Section 6.

The extraction proceeds bottom-up, starting from the proof ending in the empty clause, propagating expansions trees upward until they are at the Input-rules. At every point, each subproof is assigned a finite set of expansion sequents. Formally, we describe the extraction as a binary relation on these assignments of expansion sequents, called extraction states:

**Definition 6.** *An extraction state $\mathcal{P}; \mathcal{S}$ is a pair consisting of a set $\mathcal{P}$ and an expansion sequent $\mathcal{S}$. Each element of $\mathcal{P}$ is a pair $(\pi, \mathcal{E})$, where $\pi$ is a subproof* *ending in $\mathcal{T}$ and $\mathcal{E}$ is a negative expansion sequent such that there exists a substitution $\sigma$ with $\mathcal{T}\sigma = \mathrm{sh}(\mathcal{E})$. The* deep formula of the extraction state *is defined as:*

$$\mathrm{dp}(\mathcal{P};\mathcal{S}) = \left(\bigwedge_{(\pi,\mathcal{E})\in\mathcal{P}} \mathrm{dp}(\mathcal{E})\right) \to \mathrm{dp}(\mathcal{S})$$

Please note that the trees of the *negative* expansion sequents here have the opposite polarity as in the expansion sequents in Section 4. This is due to the fact that resolution proofs are proofs by refutation. The resolution proof starts with $\varphi \vdash$ while the expansion proof has $\vdash \varphi$ as the shallow sequent—observe that $\varphi$ occurs in opposite polarities here.

Given a resolution proof $\pi$, the *initial extraction state* is $\mathcal{C}_\pi = (\{(\pi,\vdash)\};\vdash)$. We can now define a relation $\rightsquigarrow$ on extraction states, that transforms the initial extraction state into extraction state of the form $\emptyset;\mathcal{E}$, where $\mathcal{E}$ is the desired expansion proof. This relation preserves the quasi-tautologyhood of the deep formula—we will prove this crucial property in Lemma 5. The relation $\rightsquigarrow$ is the smallest relation containing the following cases:

$$\begin{aligned}
\mathcal{P},(\pi,\mathcal{E}_1),(\pi,\mathcal{E}_2);\mathcal{S} &\rightsquigarrow \mathcal{P},(\pi,\mathcal{E}_1 \sqcup \mathcal{E}_2);\mathcal{S} \quad \text{if } \mathrm{sh}(\mathcal{E}_1) = \mathrm{sh}(\mathcal{E}_2)\\
\mathcal{P}; E_1 +\!: E_2 +\!: \mathcal{S} &\rightsquigarrow \mathcal{P}; E_1 \sqcup E_2 +\!: \mathcal{S} \quad \text{if } \mathrm{sh}(E_1) = \mathrm{sh}(E_2)\\
\mathcal{P};\mathcal{S} :\!+ E_1 :\!+ E_2 &\rightsquigarrow \mathcal{P};\mathcal{S} :\!+ E_1 \sqcup E_2 \quad \text{if } \mathrm{sh}(E_1) = \mathrm{sh}(E_2)
\end{aligned}$$

*Logical connectives:*

$$\begin{aligned}
\mathcal{P},(\mathrm{Input}, E\vdash);\mathcal{S} &\rightsquigarrow \mathcal{P};\mathcal{S} :\!+ E\\
\mathcal{P},(\mathrm{TopL}(\pi),\mathcal{E});\mathcal{S} &\rightsquigarrow \mathcal{P},(\pi,\top^+ +\!: \mathcal{E});\mathcal{S}\\
\mathcal{P},(\mathrm{NegL}(\pi),\mathcal{E} :\!+ E_1);\mathcal{S} &\rightsquigarrow \mathcal{P},(\pi,\neg E_1 +\!: \mathcal{E});\mathcal{S}\\
\mathcal{P},(\mathrm{NegR}(\pi),E_1 +\!: \mathcal{E});\mathcal{S} &\rightsquigarrow \mathcal{P},(\pi,\mathcal{E} :\!+ \neg E_1);\mathcal{S}\\
\mathcal{P},(\mathrm{AndL}(\pi),E_1 +\!: E_2 +\!: \mathcal{E});\mathcal{S} &\rightsquigarrow \mathcal{P},(\pi,(E_1 \wedge E_2) :\!+ \mathcal{E});\mathcal{S}\\
\mathcal{P},(\mathrm{AndR1}(\pi),\mathcal{E} :\!+ E_1);\mathcal{S} &\rightsquigarrow \mathcal{P},(\pi,\mathcal{E} :\!+ (E_1 \wedge \mathrm{wk}^-(\dots)));\mathcal{S}\\
\mathcal{P},(\mathrm{AndR2}(\pi),\mathcal{E} :\!+ E_2);\mathcal{S} &\rightsquigarrow \mathcal{P},(\pi,\mathcal{E} :\!+ (\mathrm{wk}^-(\dots) \wedge E_2));\mathcal{S}\\
\mathcal{P},(\mathrm{AllR}(\pi),\mathcal{E} :\!+ E_1);\mathcal{S} &\rightsquigarrow \mathcal{P},(\pi,\mathcal{E} :\!+ (\forall x\,\varphi +^{x\sigma} E_1));\mathcal{S} \quad (\varphi\sigma = \mathrm{sh}(E_1))\\
\mathcal{P},(\mathrm{AllL}(\pi),E_1 +\!: \mathcal{E});\mathcal{S} &\rightsquigarrow \mathcal{P},(\pi,(\forall x\,\varphi +^{t}_{\mathrm{sk}} E_1) +\!: \mathcal{E});\mathcal{S}
\end{aligned}$$

*Subformula definitions:*

$$\begin{aligned}
\mathcal{P},(\mathrm{AbbrL}(\pi),D^+(\bar t) +\!: \mathcal{E});\mathcal{S} &\rightsquigarrow \mathcal{P},(\pi,(\varphi +_{\mathrm{def}} D^+(\bar t)) +\!: \mathcal{E}));\mathcal{S}\\
\mathcal{P},(\mathrm{DefR}(D(\bar t)),E_D \vdash E_\varphi);\mathcal{S} &\rightsquigarrow \mathcal{P};\forall\bar x(D(\bar x) \to \varphi) +^{\bar t} (E_D \to E_\varphi) +\!: \mathcal{S}
\end{aligned}$$

*Avatar splitting:*

$$\mathcal{P},(\mathrm{AvSplit}(\pi),\mathcal{E}_1 :\!+ D^-);\mathcal{S} \rightsquigarrow \mathcal{P},(\pi,\mathcal{E}_1 +\!+ \mathcal{S}_2^-[\bar x\backslash\bar y]);(\forall\bar x\,\mathcal{S}_2 +^{\bar y} \mathcal{S}_2^+[\bar x\backslash\bar y] \to D^-) +\!: \mathcal{S}$$

where $\mathcal{S}_2$ is as in the inference rule, and $\bar y$ are fresh variables

$$\emptyset; E_1 \to D^-, D^+ \to E_2, \mathcal{S} \rightsquigarrow \emptyset; \mathrm{Cut}(E_1,E_2) +\!: \mathcal{S} \quad \text{if } D \text{ does not occur in } \mathcal{S}$$

*Logical inferences:*

$$\begin{aligned}
\mathcal{P}, (\mathrm{Res}(\pi_1, \pi_2), \mathcal{E}_1\ \mathcal{E}_2); \mathcal{S} &\rightsquigarrow \mathcal{P}, (\pi_1, \mathcal{E}_1 :+ \varphi^-), (\pi_2, \varphi^+ +: \mathcal{E}_2); \mathcal{S} \\
\mathcal{P}, (\mathrm{Factor}(\pi), \mathcal{E} :+ E_1); \mathcal{S} &\rightsquigarrow \mathcal{P}, (\pi, \mathcal{E} :+ E_1 :+ E_1); \mathcal{S} \\
\mathcal{P}, (\mathrm{Subst}(\pi), \mathcal{E}); \mathcal{S} &\rightsquigarrow \mathcal{P}, (\pi, \mathcal{E}); \mathcal{S} \\
\mathcal{P}, (\mathrm{Taut}(\varphi), E_1 \vdash E_2); \mathcal{S} &\rightsquigarrow \mathcal{P}; \mathcal{S} \\
\mathcal{P}, (\mathrm{Rw}(\pi_1, \pi_2), \mathcal{E}_1 ++ \mathcal{E}_2 :+ E_3[s]); &\rightsquigarrow \mathcal{P}, (\pi_1, \mathcal{E}_1 :+ (t = s)^-), (\pi_2, \mathcal{E}_2 :+ E_3[t]); \mathcal{S} \\
\mathcal{P}, (\mathrm{Refl}, \vdash (t = t)^-) &\rightsquigarrow \mathcal{P}; \mathcal{S}
\end{aligned}$$

*Example 5.* Let $\pi_6$ be the resolution proof in Example 1. Then the relation $\rightsquigarrow$ extracts the expansion proof as follows, where $\varphi_x = \forall y (P(x) \to P(y))$:

$$\begin{aligned}
&(\pi_6, \vdash); \vdash \ \rightsquigarrow\ (\pi_4, \underbrace{P(s(x))}_{=E_1} \vdash), (\pi_5, \vdash E_1); \vdash \ \rightsquigarrow\ (\pi_4, E_1 \vdash), (\pi_3, \vdash E_1); \vdash \\
&\rightsquigarrow^2 (\pi_2, \underbrace{\mathrm{wk}^-(P(x)) \to E_1}_{=E_2} \vdash), (\pi_2, \underbrace{E_1 \to \mathrm{wk}^+(P(s(s(x))))}_{=E_3} \vdash); \vdash \\
&\rightsquigarrow^2 (\pi_1, \varphi_x +^x_{\mathrm{sk}} E_2 \vdash), (\pi_1, \varphi_{s(x)} +^{s(x)}_{\mathrm{sk}} E_3 \vdash); \vdash \\
&\rightsquigarrow^2 (\pi_1, \underbrace{\forall x \varphi_x +^x \varphi_x +^x_{\mathrm{sk}} E_2}_{=E_4} \vdash), (\pi_1, \underbrace{\forall x \varphi_x +^{s(x)} \varphi_{s(x)} +^{s(x)}_{\mathrm{sk}} E_3}_{=E_5} \vdash); \vdash \\
&\rightsquigarrow (\pi_1, E_4 \sqcup E_5 \vdash); \vdash \ \rightsquigarrow\ \emptyset; \vdash E_4 \sqcup E_5
\end{aligned}$$

**Definition 7.** *Let $\pi$ be a resolution proof. Then $|\pi|_t$ and $|\pi|_d$ denote the number of its inferences when counted as a tree or a DAG, respectively.*

**Lemma 4.** *$\rightsquigarrow$ is terminating.*

*Proof.* Let $\mathcal{P}; \mathcal{S}$ be a extraction state, we define its termination measure $|\mathcal{P}; \mathcal{S}|_n = |\mathcal{S}|_c + 2 \sum_{(\pi, \mathcal{E}) \in \mathcal{P}} |\pi|_t$ where $|\mathcal{S}|_c$ is the number of expansion trees in $\mathcal{S}$. Each case of the relation $\rightsquigarrow$ decreases this termination measure.

**Lemma 5.** *Let $\mathcal{C}_1 \rightsquigarrow \mathcal{C}_2$, then $\mathrm{dp}(\mathcal{C}_1) \to \mathrm{dp}(\mathcal{C}_2)$ is a quasi-tautology. If $C_1$ is acyclic, then so is $\mathcal{C}_2$.*

*Proof.* Straightforward induction on $\rightsquigarrow$.

**Lemma 6.** *Let $\pi$ be a resolution proof where all Input inferences have the formula $\varphi$, and let $\mathcal{C}_\pi \rightsquigarrow^* (\mathcal{P}, \mathcal{S}_1 \vdash \mathcal{S}_2)$. Then $\mathcal{P} = \emptyset$, and:*

1. *for every $E \in \mathcal{S}_1$, $\mathrm{sh}(E)$ is either $\top$ or $\forall \overline{x}\, (D(\overline{x}) \to \varphi)$ where $D(\overline{t}) \overset{\mathrm{def}}{\mapsto} \varphi$, and*
2. *for every $E \in \mathcal{S}_2$, $\mathrm{sh}(E) = \varphi$.*

*Proof.* If $\mathcal{P} \neq \emptyset$, then we can apply one of the cases of $\rightsquigarrow$. Properties 1. and 2. are preserved in every case.

Using the relation $\rightsquigarrow$ we obtain an expansion proof with cuts. Since there are no Skolem or definition nodes in these cuts, we can eliminate them (see Lemma 2) to obtain a cut-free expansion proof $\mathcal{E}^*$ with definitions: $\mathcal{C}_\pi \rightsquigarrow^* (\emptyset, \mathcal{E}) \overset{\mathrm{cut}^*}{\rightsquigarrow} (\emptyset, \mathcal{E}^*)$

## 6 Definition elimination

Consider now a cut-free expansion proof $\mathcal{E}$ without eigenvariable nodes where the shallow sequent contains only definition axioms in the antecedent (this is the form of expansion proofs that are produced by the extraction in Section 5):

$$\mathcal{E} = (E_1^D, \cdots, E_n^D \vdash E_\varphi), \quad \mathrm{sh}(E_i^D) = \forall \overline{x}\, (\varphi_i(\overline{x}) \to D_i(\overline{x})), \quad \mathrm{sh}(E_\varphi) = \varphi \quad (1)$$

The expansions of $\varphi_i(\bar{t})$ may contain definition nodes as well. Due to the acyclicity of the AbbrL-inferences, we can assume that each expansion of $\varphi_i(\bar{t})$ only contains definition nodes $+_{\mathrm{def}}\, D_j(\dots)$ where $j < i$. We will now successively eliminate each of the definition axioms, starting with the one for $n$. The expansion tree for $\forall \overline{x}\, (\varphi_n(\overline{x}) \to D_n(\overline{x}))$ has the following form:

$$\forall \overline{x}\, (\varphi_n(\overline{x}) \to D_n(\overline{x})) +^{\overline{t_1}} \left(E_1 \to D_n(\overline{t_1})^-\right) \cdots +^{\overline{t_k}} \left(E_k \to D_n(\overline{t_k})^-\right)$$

For performance reasons, we consider two cases here: in the first case the deep sequent is tautological, in the second case the deep sequent is only quasi-tautological. In both cases, we replace definition nodes $+_{\mathrm{def}}\, D_n(\dots)$ in the expansion proof by subtrees of $E_n^D$. However in the second case we perform many duplications, and we want to avoid this if possible.

*Tautological deep sequent.* If $\mathrm{dp}(\mathcal{E})$ is not just quasi-tautological but tautological (this is the case if the resolution proof did not use the built-in equational inference Rw and Refl), then we merely need to replace each occurrence of a definition node $+_{\mathrm{def}}\, D_n(\overline{t_i})^+$ with the corresponding expansion tree $E_i$. We define a function $R[\cdot]$ that performs this replacement on definition nodes for $D_n$, and recursively maps over the other possible nodes:

$$R[\varphi_n(\bar{t}) +_{\mathrm{def}} D_n(\bar{t})^+] = \begin{cases} E_i & \text{if there exists } i \text{ such that } \bar{t} = \overline{t_i} \\ \mathrm{wk}^+(\varphi_n(\overline{t_i})) & \text{otherwise} \end{cases}$$

**Lemma 7.** *Let $\mathcal{E}$ be an expansion sequent as in Eq.* (1)*. If* $\mathrm{dp}(\mathcal{E})$ *is tautological, then* $\mathrm{dp}\left(E_1^D, \cdots, E_{n-1}^D \vdash R[\varphi]\right)$ *is tautological as well.*

*Proof.* Let $I$ be a counter-model for $\mathrm{dp}\left(E_1^D, \cdots, E_{n-1}^D \vdash R[\varphi]\right)$. Assume without loss of generality that $I$ is not defined for any of the atoms $D_n(\bar{t})$; we extend $I$ by setting $I(D_n(\overline{t_i})) = I(\mathrm{dp}(E_i))$ for all $i$, and $I(D_n(\bar{t})) = 0$ otherwise. We now have $I(\mathrm{dp}(E_n^D)) = 1$ as well as $I(\mathrm{dp}(R[\varphi_n(\bar{t}) +_{\mathrm{def}} D_n(\bar{t})^+])) = I(D_n(\bar{t}))$ and thus $I(\mathrm{dp}(R[E_\varphi])) = I(\mathrm{dp}(E_\varphi))$ since $D_n$ only occurs in $E_\varphi$ in definition nodes. Hence $I$ is a counter-model for $\mathrm{dp}(\mathcal{E})$ as well.

*Quasi-tautological deep sequent.* In general, $\mathrm{dp}(\mathcal{E})$ may be just quasi-tautological. In particular the following congruence scheme is quasi-tautological:

$$t_1 = s_1 \wedge \cdots \wedge t_m = s_m \to (D(t_1, \cdots, t_m) \leftrightarrow D(s_1, \cdots, s_m))$$

Hence it is no longer sufficient to replace a definition node $+_{\mathrm{def}}\, D_n(\overline{t_i})^+$ by just the expansion tree where the arguments are syntactically equal. We replace it by *all* the expansion trees $E_1, \cdots, E_n$, suitably replacing the term vectors $\overline{t_i}$ so that the shallow formulas match. To perform this replacement, we define a generalization operation $G$. We first define the generalized tree $E$, and then define the replacement operation $R^{\mathrm{eq}}$ uniformly for all definition atoms:

$$E = G_{\varphi(\overline{x})}(E_1) \sqcup \cdots \sqcup G_{\varphi(\overline{x})}(E_n)$$

$$R^{\mathrm{eq}}[\varphi_n(\overline{t}) +_{\mathrm{def}} D_n(\overline{t})^+] = E[\overline{x}\backslash\overline{t}]$$

**Definition 8.** *Let $\varphi, \psi$ be first-order formulas such that $\psi\sigma = \varphi$ for some substitution $\sigma$, and $E \in \mathrm{ET}^p(\varphi)$ an expansion tree without definition nodes. Then we recursively define its generalization $G_\psi(E) \in \mathrm{ET}^p(\psi)$:*

$$G_\psi(A^p) = \psi^p \qquad G_{\neg\psi}(\neg E) = \neg G_\psi(E) \qquad G_\psi(\mathrm{wk}^p(\varphi)) = \mathrm{wk}^p(\psi)$$
$$G_\psi(E_1 \sqcup E_2) = G_\psi(E_1) \sqcup G_\psi(E_2)$$
$$G_{\psi_1\wedge\psi_2}(E_1 \wedge E_2) = G_{\psi_1}(E_1) \wedge G_{\psi_2}(E_2)$$
$$G_{\forall x\psi}(\forall x\varphi +^y_{\mathrm{ev}} E) = \forall x\varphi +^y_{\mathrm{ev}} G_{\psi[x\backslash y]}(E)$$
$$G_{\forall x\psi}(\forall x\varphi +^t_{\mathrm{sk}} E) = \forall x\varphi +^t_{\mathrm{sk}} G_{\psi[x\backslash t]}(E)$$
$$G_{\forall x\psi}(\forall x\varphi +^{t_1} E_1 \cdots +^{t_n} E_n) = \forall x\varphi +^{t_1} G_{\psi[x\backslash t_1]}(E_1) \cdots +^{t_n} G_{\psi[x\backslash t_n]}(E_n)$$

**Lemma 8.** *Let $\mathcal{E}$ be an expansion sequent as in Eq. (1). If $\mathrm{dp}(\mathcal{E})$ is quasi-tautological, then $\mathrm{dp}\left(E_1^D, \cdots, E_{n-1}^D \vdash R^{\mathrm{eq}}[\varphi]\right)$ is quasi-tautological as well.*

*Proof.* Similar to Lemma 7, except we now set $I(D_n(\overline{a})) = I(\mathrm{dp}(E)[\overline{x}\backslash\overline{a}])$ for all $\overline{a}$ in the domain of the counter-model.

## 7 Complexity

We will now give a double-exponential upper bound on the complexity of the whole algorithm as summarized in Algorithm 1. Without Avatar inferences and definition inferences, this bound would be just single-exponential. We do not know whether this double-exponential bound is actually attained in the worst case, the best known lower bound is single-exponential (as is necessary in any conversion from resolution proofs to expansion proofs).

**Lemma 9.** *Whenever $\mathcal{C}_\pi \rightsquigarrow^* (\emptyset; \mathcal{E})$, then $|\mathcal{E}| \leq 4^{|\pi|_d+1}$.*

*Proof.* Let $\mathcal{P}; \mathcal{S}$ be a extraction state, we define its size as $|\mathcal{P}; \mathcal{S}| = |\mathcal{S}| + \sum_{(\pi,\mathcal{E})\in\mathcal{P}} 2|\pi|_t|\pi|_m + |\mathcal{E}|$, where $|\pi|_m$ is the maximum length of a sequent occurring in $\pi$. No case of the relation $\rightsquigarrow$ increases this size. Observe that $|\pi|_m \leq 2|\pi|_d$, and hence $|\mathcal{E}| \leq |\mathcal{C}_\pi| \leq 2|\pi|_t|\pi|_m \leq 2^{|\pi|_d+2}|\pi|_d \leq 4^{|\pi|_d+1}$.

**Lemma 10.** *Let $\pi$ be a resolution proof, then* $|\textsc{Transform}(\pi)| \leq 2^{4^{|\pi|_d}}$.

*Proof.* Let $E_1, E_2, E_3$ be as in Algorithm 1. From Lemma 9 we know that $|E_1| \leq 4^{|\pi|_d+1}$. Note that there are fewer than $|\pi|_d$ splitting and definition inferences in $\pi$, and hence fewer than $|\pi|_d$ cuts and definitions in $E_1$ and $E_2$. The cost of cut-elimination is shown in Lemma 3, we have $|E_2| \leq |E_1|^{2^c}$ where $c < |\pi|_d$ is the number of cuts. We have a similar bound for definition elimination: each step increases the size at most quadratically and we have $|E_3| \leq |E_2|^{2^d}$ where $d < |\pi|_d$ is the number of definitions. Hence $|\textsc{Transform}(\pi)| \leq 4^{(|\pi|_d+1)2^{c+d}} \leq 2^{4^{|\pi|_d}}$.

## 8 Empirical evaluation

GAPT[1] is an open source framework for proof transformations [6] and contains an implementation of Algorithm 1. We evaluated the implementation in three experiments using GAPT 2.9.

**Algorithm 1** Transforming expansion proofs to resolution proofs

```
function Transform($\pi$)
$E_1 \leftarrow$ Extract($\pi$)                          (see Section 5)
$E_2 \leftarrow$ EliminateCuts($E_1$)                    (see Lemma 2)
$E_3 \leftarrow$ EliminateDefinitions($E_2$)             (see Lemmas 7 and 8)
return $E_3$
```

First, we compared it against the LK-based method for expansion proof extraction from resolution proofs described in [11], which is also implemented in GAPT. The TSTP library (Thousands of Solutions from Theorem Provers, see [19]) contains proofs from a variety of automated theorem provers. We loaded all 6341 Prover9 proofs into GAPT, and measured the runtime of each of the algorithms with a timeout of 120 seconds. GAPT is unable to read 40 of the proofs at all since Prover9's `prooftrans` executable fails. Of the remaining proofs, the LK-based method in [11] imports 5479 proofs (87.0%). Algorithm 1 imports 6125 proofs (97.2%), it only fails on 176 proofs due to the timeout. There is no proof where the LK-based method is successful, but Algorithm 1 fails.

Figure 1 plots the number of successfully imported proofs using each of the algorithms against the DAG-like size of the resolution proof. The algorithm in this paper manages to import much larger proofs, proofs between 1000 and 10000 inferences can not be imported by the LK-based method at all.

As the second experiment, we evaluated the quality of the clausification allowed by the resolution calculus used in this paper. We took the 662 problems in the first-order FEQ, FNE, FNN, and FNQ divisions of the CASC-26 competition whose size was less than one megabyte after including the separate axiom files. On these 662 problems, we compared the performance of the E theorem prover[2] [18] version 2.1 (as submitted to CASC) when directly running on the problems, and when running on the clausification produced by GAPT.

[1] available at `https://logic.at/gapt`

[2] We picked the highest-ranking prover in the first-order theorem category of the CASC-26 competition whose license allows competitive evaluation.

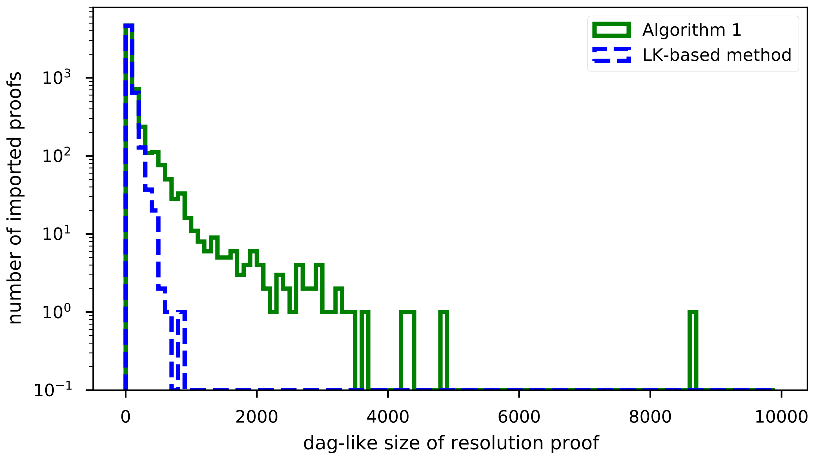

**Fig. 1.** Comparison of the number of successfully imported Prover9 proofs in the TSTP using Algorithm 1 and the LK-based algorithm in [11].

GAPT fails to clausify 19 of the problems due to excessive runtime. These problems (e.g. `HWV053+1`) have blocks of more than thousand quantifiers. On the remaining ones we ran E in both the default configuration, and the auto-scheduling mode (as used in the CASC), both with a timeout of 60 seconds. In the default mode the E clausification results in 81 found proofs, the GAPT clausification in 89. With auto-scheduling enabled the result is reversed but still close, and E's own clausification produces more proofs (308) than GAPT's clausification (285). These results show that the clausification allowed by the calculus presented in this paper is competitive with the ones implemented in state-of-the-art automated theorem provers. We believe that the reversed results for the auto-scheduling mode are indicative of a larger trend in first-order theorem provers: many provers train their strategy selection algorithms on the exact problems from the TPTP and are thus rely very closely on the syntactic features of these problems.

Finally, we wanted to compare the runtime of Algorithm 1 with the runtime of the first-order prover. To this end, we again used the same 662 first-order problems from CASC-26 and used GAPT's external prover interface to obtain expansion proofs from E. This prover interface uses the clausification of the resolution calculus described in this paper to create a CNF, sends this CNF to E, then parses and reconstructs a resolution proof using proof replay, and finally constructs an expansion proof using Algorithm 1. We measured the mean runtime of each phase on the successfully imported proofs (only 7 proofs could not be imported in the time limit of 2 minutes). The E prover itself takes up 62%, the largest part of the runtime. Algorithm 1 only takes 7.2% of the total runtime. The rest of the runtime is spent mainly in proof replay (15.6%) and clausification (4.6%).

Since the expansion proof extraction is only a fraction of the prover runtime, we believe that is practically feasible to generate expansion proofs instead of resolution proofs. Even though expansion proofs can be exponentially larger in the worst case, this situation seems to occur only rarely.

## 9 Conclusion

The transformation described in this paper has been used as the default method to generate expansion proofs from resolution proofs in GAPT since version 2.2. Using it, GAPT can effectively interface with six different external resolution-based provers, including SPASS and Vampire with their splitting rules. The modular integration of structural clausification makes it possible to reuse it for non-resolution provers as well: the interface to the connection-based prover LeanCoP [15,17] uses the same code for clausification and definition elimination.

A limitation of the clausification is that it can expand each definition only in a single polarity. Lifting this restriction would produce cuts in the definition elimination phase. However, these cuts would contain Skolem nodes and cannot be eliminated directly. In the equality-free case, there is a reliable deskolemization procedure [2] which can be used as a preprocessing step to enable cut-elimination. Such a procedure is yet missing for proofs with equational reasoning. Reliable deskolemization would also enable a straightforward adaptation of external clausifications via proof replay.

Many of the techniques used here come from higher-order logic, both Andrew's calculus $\mathcal{R}$ and expansion trees originate in that setting. It seems only natural to extend this transformation to higher-order logic, and there seem to be no immediate obstacles except for the built-in equational inferences. But these could be straightforwardly translated to explicit Leibniz equality.

Many clausification procedures perform propositional simplification rules as a pre-processing step, for example rewriting $\varphi \land \top \mapsto \varphi$ or converting to negation normal form. These simplifications could be helpful in the clausification here as well, since they potentially enable sharing of subformula definitions and Skolem functions. These could be supported by adding new inference rules to the resolution calculus and adapting the expansion tree extraction in the natural way.

Finally, for simplicity we convert the SMT refutations in proofs using Avatar to resolution proofs first. However there is no fundamental reason why we need to perform this costly conversion, since the SMT refutation is purely ground and is essentially discarded in this translation. Adding a new rule to represent this part of the proof in a single inference would deliver even greater performance.

## References

1. Andrews, P.B.: Resolution in type theory. Journal of Symbolic Logic 36(3), 414–432 (1971)
2. Baaz, M., Hetzl, S., Weller, D.: On the complexity of proof deskolemization. Journal of Symbolic Logic 77(2), 669–686 (2012)

3. Buss, S.R.: On Herbrand's Theorem. In: Logic and Computational Complexity, Lecture Notes in Computer Science, vol. 960, pp. 195–209. Springer (1995)
4. Eberhard, S., Hetzl, S.: Inductive theorem proving based on tree grammars. Annals of Pure and Applied Logic 166(6), 665–700 (2015)
5. Ebner, G., Hetzl, S., Leitsch, A., Reis, G., Weller, D.: On the generation of quantified lemmas (2017), submitted
6. Ebner, G., Hetzl, S., Reis, G., Riener, M., Wolfsteiner, S., Zivota, S.: System description: GAPT 2.0. In: Olivetti, N., Tiwari, A. (eds.) International Joint Conference on Automated Reasoning (IJCAR). Lecture Notes in Computer Science, vol. 9706, pp. 293–301. Springer (2016)
7. Herbrand, J.: Recherches sur la théorie de la démonstration. Ph.D. thesis, Université de Paris (1930)
8. Hetzl, S., Leitsch, A., Reis, G., Tapolczai, J., Weller, D.: Introducing quantified cuts in logic with equality. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) International Joint Conference on Automated Reasonin (IJCAR). Lecture Notes in Computer Science, vol. 8562, pp. 240–254. Springer (2014)
9. Hetzl, S., Leitsch, A., Reis, G., Weller, D.: Algorithmic introduction of quantified cuts. Theoretical Computer Science 549, 1–16 (2014)
10. Hetzl, S., Leitsch, A., Weller, D.: Towards Algorithmic Cut-Introduction. In: Logic for Programming, Artificial Intelligence and Reasoning (LPAR-18). Lecture Notes in Computer Science, vol. 7180, pp. 228–242. Springer (2012)
11. Hetzl, S., Libal, T., Riener, M., Rukhaia, M.: Understanding Resolution Proofs through Herbrand's Theorem. In: 22$^{\text{nd}}$ International Conference on Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX. pp. 157–171 (2013)
12. Hetzl, S., Weller, D.: Expansion trees with cut (2013), `https://arxiv.org/abs/1308.0428`
13. Luckhardt, H.: Herbrand-Analysen zweier Beweise des Satzes von Roth: Polynomiale Anzahlschranken. The Journal of Symbolic Logic 54(1), 234–263 (1989)
14. Miller, D.A.: A compact representation of proofs. Studia Logica 46(4), 347–370 (1987)
15. Otten, J.: leancop 2.0 and ileancop 1.2: High performance lean theorem proving in classical and intuitionistic logic. In: 4th International Joint Conference on Automated Reasoning, IJCAR. pp. 283–291 (2008)
16. Pfenning, F.: Analytic and non-analytic proofs. In: Shostak, R.E. (ed.) 7$^{\text{th}}$ International Conference on Automated Deduction, CADE. Lecture Notes in Computer Science, vol. 170, pp. 394–413. Springer (1984)
17. Reis, G.: Importing SMT and connection proofs as expansion trees. In: Fourth Workshop on Proof eXchange for Theorem Proving, PxTP. pp. 3–10 (2015)
18. Schulz, S.: System Description: E 1.8. In: McMillan, K., Middeldorp, A., Voronkov, A. (eds.) 19th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR). LNCS, vol. 8312. Springer (2013)
19. Sutcliffe, G.: The TPTP problem library and associated infrastructure: The FOF and CNF parts, v3.5.0. Journal of Automated Reasoning 43(4), 337–362 (2009)
20. Voronkov, A.: AVATAR: the architecture for first-order theorem provers. In: Biere, A., Bloem, R. (eds.) 26$^{\text{th}}$ International Conference on Computer Aided Verification, CAV 2014. Lecture Notes in Computer Science, vol. 8559, pp. 696–710. Springer (2014)
21. Weidenbach, C.: Combining superposition, sorts and splitting. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 2, chap. 27, pp. 1965–2013 (2001)