Under consideration for publication in Math. Struct. in Comp. Science

Complexity of Translations from Resolution to Sequent Calculus

Giselle Reis[1]† and Bruno Woltzenlogel Paleo[2]

[1] *Carnegie Mellon University, Doha, Qatar* – giselle@cmu.edu
[2] bruno.wp@gmail.com

Received 9 July 2019

Resolution and sequent calculus are two well-known formal proof systems. Their differences make them suitable for distinct tasks. Resolution and its variants are very efficient for automated reasoning and are in fact the theoretical basis of many theorem provers. However, being intentionally machine-oriented, the resolution calculus is not as natural for human beings and the input problem needs to be pre-processed to clause normal form. Sequent calculus, on the other hand, is a modular formalism that is useful for analyzing meta-properties of various logics and is, therefore, popular among proof-theorists. The input problem does not need to be pre-processed, and proofs are more detailed. However, proofs also tend to be larger and more verbose. When the worlds of proof theory and automated theorem proving meet, translations between resolution and sequent calculus are often necessary. In this paper we compare three translation methods and analyze their complexity.

1. Introduction

The representation of proofs as structured mathematical objects is in the core of proof theory. Nevertheless, it is unlikely that a single best representation will ever be developed. Depending on what one needs proofs for, it makes sense to prefer one proof system over another. Two widely used formalisms are the resolution calculus and the sequent calculus.

Variations, refinements and extensions of resolution are used in many contemporary theorem provers (McCune 2005–2010, Schulz 2013, Bouton et al. 2009, Weidenbach et al. 2009, Korovin 2008, Benzmüller et al. 2015, Itegulov et al. 2017, Kovács & Voronkov 2013) due to their simplicity and efficiency in proof search. Simplicity is achieved by requiring the input problem to be transformed to clause normal form (i.e. conjunction of disjunctions of literals that are either atomic formulas or negated atomic formulas), which allows the calculus to have only two inference rules (resolution and factoring). Efficiency in proof search is achieved by restricting instantiation through unification and

† This paper was made possible by grant NPRP 7-988-1-178, from the Qatar National Research Fund (a member of the Qatar Foundation). The statements made herein are solely the responsibility of the authors.

by using various refinements that restrict the application of the inference rules while retaining completeness. As a result, proofs are relatively compact, but do not hold much information. To begin with, a resolution refutation is a proof of the *unsatisfiability* of the *negation of the theorem*. This means that the theorem is valid, but the refutation is not a direct validity proof. Then, since the need for a clause normal form requires modification of the conjecture in a number of ways (negations are pushed deeper, quantifiers are prenexified and skolemized, and disjunctions are distributed over conjunctions – or new symbols are introduced to avoid the exponential blow-up of the distribution), it might be hard to map each resolution step into some insight about the original problem statement.

Sequent calculus was introduced by Gentzen (Gentzen 1969) as a meta-calculus to reason about natural deduction derivations and it continues to be used by most proof theorists for proof analysis and for meta-analysis of a logic's properties. In principle, the calculus is composed of two or more rules for each connective, which represent the semantics of the connective when it appears in a goal or in a hypothesis (e.g. $\wedge$ in a goal means one needs to prove two subgoals, whereas $\wedge$ in a hypothesis means one has two subhypotheses available to use). The proximity to a semantic interpretation makes it convenient to show the calculus' soundness and completeness. Proving the logic's consistency is also straightforward (usually as a corollary of cut elimination). Additionally, sequent calculi have been used as proof systems for many different logics, as the formalism is modular and easily adaptable. The existence of many rules for connectives in different contexts makes it possible to work on a theorem without having to transform it. This characteristic also enables a better mapping of human reasoning steps to the formal proof steps. Consequently, much more information can be extracted from a sequent calculus proof. It is not a coincidence that many proof assistants (where proofs are constructed through scripts written by humans) are based on (higher-order) natural deduction and their basic tactic commands resemble sequent calculus rules.

Occasionally, translations between the two systems are necessary. One situation where this was the case was in studies of the compression of sequent calculus proofs via the introduction of cuts, as in the methods proposed in (Woltzenlogel Paleo 2010) (atomic cuts) and (Hetzl et al. 2014) (first order cuts). In the former, the cuts to be introduced are obtained from a resolution refutation of a clause set extracted from the cut-free proof: the sequent calculus proof with cuts is the result of combining the resolution refutation with parts of the original cut-free proof. In the latter, the cut-introduction method was validated and evaluated on a large database of sequent calculus proofs that were obtained by translating resolution-based proofs (Reis 2015).

Secondly, translations from resolution to sequent calculus are necessary in some approaches to proof checking. As automated theorem provers are complex pieces of software and therefore vulnerable to bugs, the output of proofs is a common solution to the issue of lack of trust arising from this complexity. Proofs certify that the answer given by the prover is correct, even if the prover itself might not be completely correct. If a user can successfully and independently check the proof, the user can trust the prover's answer even without fully trusting the prover. The *foundational proof certificate* initiative (Miller 2013) adheres to this approach and proposes a conceptual framework to uniformly check proofs in a variety of calculi and formats. Due to its versatility, a sequent calculus was chosen as the meta-calculus for this general proof-checking task and, in (Chihani et al. 2017) in particular, an embedding of resolution into LKF (focused sequent calculus for classical logic) was defined. *Focusing* is used to obtain a fine-grained correspondence between the sequent calculus proof and the resolution proof, but focusing is not essential. Leaving focusing aside, their embedding can be seen as a translation from resolution to regular sequent calculus.

Yet a third situation in which a translation from resolution to sequent calculus proofs may arise is when proving soundness and completeness of one system with respect to the other. This is the goal in (Hermant 2010). Although there the translation is defined between calculi with *deduction modulo*, this is not essential and can be left aside as well. It should be noted that, given the theoretical context in which the third translation was conceived, complexity was not as relevant a concern as it was for the first and second translations. The third translation is nevertheless included in our analysis, because it can be used to translate resolution refutations into *cut-free* sequent calculus proofs; a feature the other two translations lack.

The translations proposed in the situations mentioned previously appeared as a side-product of the main work, and thus not much attention has been paid to them. In this paper, we redefine all translations in a common setting such that their differences and commonalities are more evident. Interestingly, each uses a different interpretation of resolution in sequent calculus: as cuts on literals, as cuts on resolvents, and as axioms. Moreover we present a complexity analysis of each translation and discuss how they are related.

2. Preliminaries

We work with classical first-order logic without equality. *Terms* are variables $(x, y, \dots)$ or functions $(f, g, \dots)$ applied to terms (a constant is a 0-ary function). *Formulas* $(P, Q, \dots)$ are composed of predicate symbols $(p, q, \dots)$ applied to terms, and the connectives $\neg$, $\vee$, $\wedge$, $\forall$, $\exists$, $\top$, and $\bot$. A variable is *free* in a formula if it is not quantified over. Otherwise it is *bound*. An *atom* A is a formula composed of one n-ary predicate symbol and no connective. A *literal* is an atom (A) or its negation $(\neg A)$. $\overline{L}$ denotes the dual of a literal L (i.e. if A is an atom, $\overline{A} = \neg A$ and $\overline{\neg A} = A$). A formula is said to be in *conjunctive normal form* (i.e., CNF) if it is a prenexed (all quantifiers occur at the head) conjunction of disjunction of literals. Every formula in classical logic can be transformed into CNF form using logical equivalences. A *clause* $(C, D, \dots)$ is a disjunction of literals, possibly with free variables. Let $x_1, \dots, x_n$ denote the free variables of a clause C^*. The *universal closure* of C^* is defined as $C = \forall x_1 \dots \forall x_n . C^*$. In what follows, we use the $*$ superscript to distinguish between a clause (with free variables) and its universal closure. While this may seem counter-intuitive, we choose this notation because most of the time we will deal with the universal closure C of a clause instead of its open form C^*.

2.1. *Resolution*

Resolution is a calculus used for proving unsatisfiability of a formula in propositional or first-order (classical) logic. It works on skolemized formulas in conjunctive normal form (CNF) and it is used in most first-order automated theorem provers in some modified and extended form. A formula F is unsatisfiable iff there exists a resolution refutation of F (i.e. a derivation of the empty clause $\Box$ from the clauses in the CNF of F) (Robinson 1965). Due to the duality between unsatisfiability and validity in classical logic, one can show the validity of a formula F by presenting a resolution refutation of $\neg F$.

Definition 1 (Resolution calculus). Let C_i be a disjunction of literals and σ be the most general unifier (m.g.u.) of A and A'. The *resolution* and *factoring* rules of the resolution calculus are:

$$\frac{C_1 \vee A \vee C_2 \quad D_1 \vee \overline{A'} \vee D_2}{(C_1 \vee C_2 \vee D_1 \vee D_2)\sigma}\, R \qquad \frac{C_1 \vee A \vee C_2 \vee A' \vee C_3}{(C_1 \vee A \vee C_2 \vee C_3)\sigma}\, F$$

Example 1. The following is the specification of a family of unsatisfiable clause sets, where a, b are constants and x_i are variables:

$$\begin{array}{l}
p_1(x_1) \vee \cdots \vee p_n(x_n)\\
\quad q_1 \vee \neg p_1(a)\\
\quad \neg q_1 \vee \neg p_1(b)\\
\quad q_2 \vee p_1(x_1) \vee \neg p_2(a)\\
\quad \neg q_2 \vee p_1(x_1) \vee \neg p_2(b)\\
\quad \vdots\\
\quad q_n \vee p_1(x_1) \vee \cdots \vee p_{n-1}(x_{n-1}) \vee \neg p_n(a)\\
\quad \neg q_n \vee p_1(x_1) \vee \cdots \vee p_{n-1}(x_{n-1}) \vee \neg p_n(b)
\end{array}$$

We take the clause set when $n = 2$ for our example:

$$\{p_1(x_1) \vee p_2(x_2),\ q_1 \vee \neg p_1(a),\ \neg q_1 \vee \neg p_1(b),\ q_2 \vee p_1(x_1) \vee \neg p_2(a),\ \neg q_2 \vee p_1(x_1) \vee \neg p_2(b)\}$$

The resolution refutation (with omitted parentheses) is:

$$\frac{\overset{\rho_1}{q_1} \quad \overset{\rho_2}{\neg q_1}}{\Box}\, R$$

Where ρ_1 and ρ_2 are, respectively:

$$\frac{q_1 \vee \neg p_1 a \quad \overset{\eta_1}{p_1 x_1}}{q_1}\, R \qquad \frac{\neg q_1 \vee \neg p_1 b \quad \overset{\eta_1}{p_1 x_1}}{\neg q_1}\, R$$

And η_1 is:

$$\dfrac{\dfrac{q_2 \vee p_1 x_1 \vee \neg p_2 a \quad p_1 x_1 \vee p_2 x_2}{q_2 \vee p_1 x_1}\, R, F \quad \dfrac{\neg q_2 \vee p_1 x_1 \vee \neg p_2 b \quad p_1 x_1 \vee p_2 x_2}{\neg q_2 \vee p_1 x_1}\, R, F}{p_1 x_1}\, R, F$$

Notice how the sub-derivation η_1 of $p_1(x_1)$ is used by both ρ_1 and ρ_2. By representing this directed acyclic graph (DAG) graphically, this can be seen more clearly:

$$\begin{tikzcd}
q_2 \vee p_1(x_1) \vee \neg p_2(a) \arrow[dr, no head] & & p_1(x_1) \vee p_2(x_2) \arrow[dl, no head] \arrow[dr, no head] & & \neg q_2 \vee p_1(x_1) \vee \neg p_2(b) \arrow[dl, no head] \\
& q_2 \vee p_1(x_1) \vee p_1(x_1) \arrow[d, no head] & & \neg q_2 \vee p_1(x_1) \vee p_1(x_1) \arrow[d, no head] & \\
& q_2 \vee p_1(x_1) \arrow[dr, no head] & & \neg q_2 \vee p_1(x_1) \arrow[dl, no head] & \\
& & p_1(x_1) \vee p_1(x_1) \arrow[d, no head] & & \\
q_1 \vee \neg p_1(a) \arrow[dr, no head] & & p_1(x_1) \arrow[dl, no head] \arrow[dr, no head] & & \neg q_1 \vee \neg p_1(b) \arrow[dl, no head] \\
& q_1 \arrow[dr, no head] & & \neg q_1 \arrow[dl, no head] & \\
& & \Box & &
\end{tikzcd}$$

It is straightforward to generalize the example DAG proof shown above to other values of n. For any n, the DAG proof is a tower of fixed width, height $2n$, length (i.e. number of nodes) $O(n)$ and size (i.e. number of symbols) $O(n^2)$. If, however, the DAG were expanded to a tree, its length and its size would be $\Omega(2^n)$ (i.e. an exponential blow up would occur), because the sub-derivations η_k $(1 \leq k \leq n)$ would have to be duplicated.

Because of the worst-case exponential blow-up that can happen if DAG proofs are expanded to proof trees, as illustrated in Example 1, automated theorem provers invariably represent resolution proofs as DAGs during and after proof search. Since the translations analyzed here use resolution proofs produced by automated theorem provers, their complexities will be parametrized by the length of resolution proofs as DAGs.

Moreover, the result of all translations are sequent calculus proofs of the refuted clause set, i.e. skolemized and in CNF. Therefore, we shall not account for any blow-up on the input size due to normalization.

2.2. *Sequent Calculus*

Sequent calculus proof systems were proposed by Gentzen (Gentzen 1969) in order to study normalization of proofs in classical and intuitionistic first-order logics. Their adaptability to many logics and the uniformity with which one could prove the system's consistency made the formalism very popular among logicians. A sequent is a structure $\Gamma \vdash \Delta$, where Γ and Δ are multi-sets of formulas and $\vdash$ denotes the entailment relation. Its meaning is that the conjunction of the formulas in Γ implies the disjunction of the formulas in Δ. A sequent calculus is a collection of inference rules on sequents. In this paper we will use the sequent calculus **LK** for classical logic in Figure 1. An **LK** *proof* is a tree of inference rule applications where all leaves are axioms init, $\bot_l$ or $\top_r$, otherwise we call it an **LK** *derivation.*

$$
\begin{array}{cc}
\dfrac{}{\Gamma, A \vdash \Delta, A}\ \textsf{init} & \dfrac{\Gamma \vdash \Delta, P \quad \Gamma, P \vdash \Delta}{\Gamma \vdash \Delta}\ \textsf{cut} \\[2ex]
\dfrac{}{\Gamma, \bot \vdash \Delta}\ \bot_l & \dfrac{}{\Gamma \vdash \Delta, \top}\ \top_r \\[2ex]
\dfrac{\Gamma \vdash \Delta, P}{\Gamma, \neg P \vdash \Delta}\ \neg_l & \dfrac{\Gamma, P \vdash \Delta}{\Gamma \vdash \Delta, \neg P}\ \neg_r \\[2ex]
\dfrac{P, Q, \Gamma \vdash \Delta}{P \wedge Q, \Gamma \vdash \Delta}\ \wedge_l & \dfrac{\Gamma \vdash \Delta, P \quad \Gamma \vdash \Delta, Q}{\Gamma \vdash \Delta, P \wedge Q}\ \wedge_r \\[2ex]
\dfrac{P, \Gamma \vdash \Delta \quad Q, \Gamma \vdash \Delta}{P \vee Q, \Gamma \vdash \Delta}\ \vee_l & \dfrac{\Gamma \vdash \Delta, P, Q}{\Gamma \vdash \Delta, P \vee Q}\ \vee_r \\[2ex]
\dfrac{P\{x \leftarrow \alpha\}, \Gamma \vdash \Delta}{\exists x.P, \Gamma \vdash \Delta}\ \exists_l & \dfrac{\Gamma \vdash \Delta, P\{x \leftarrow t\}}{\Gamma \vdash \Delta, \exists x.P}\ \exists_r \\[2ex]
\dfrac{P\{x \leftarrow t\}, \Gamma \vdash \Delta}{\forall x.P, \Gamma \vdash \Delta}\ \forall_l & \dfrac{\Gamma \vdash \Delta, P\{x \leftarrow \alpha\}}{\Gamma \vdash \Delta, \forall x.P}\ \forall_r \\[2ex]
\dfrac{P, P, \Gamma \vdash \Delta}{P, \Gamma \vdash \Delta}\ c_l & \dfrac{\Gamma \vdash \Delta, P, P}{\Gamma \vdash \Delta, P}\ c_r
\end{array}
$$

Fig. 1. **LK**: Sequent calculus for classical logic (A is an atom, α is a variable not contained in P, Γ or Δ, and t does not contain variables bound in P).

Note that we are using the additive version of the binary rules. For certain translations, a multiplicative version of the cut-rule is needed (i.e. one that splits the conclusion contexts between the premises). In this case, we can safely assume the use of weakening, since this rule is length-preserving admissible. We formally show this property after defining proof length[†].

Remark 1. For the sake of space, the main formula of a rule is sometimes omitted and considered to be in the context Γ.

Example 2. We show a sequent calculus proof for the same example as before with $n = 1$. In this case, the clause set is: $\{p_1(x_1),\ q_1 \vee \neg p_1(a),\ \neg q_1 \vee \neg p_1(b)\}$.

To show unsatisfiability of a set of formulas Γ in sequent calculus, we need to show a proof of $\Gamma \vdash$. Let Γ be $\{\forall x_1.p_1(x_1),\ q_1 \vee \neg p_1(a),\ \neg q_1 \vee \neg p_1(b)\}$. One of the possible sequent calculus proofs of Γ's unsatisfiability is:

[†] The dual argument also works: with multiplicative rules, contraction is admissible. Choosing additive rules maximizes the number of invertible rules and eases the definition of the translation described in Section 4.

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{init}
\UnaryInfC{$\Gamma, p_1(a), p_1(b), q_1 \vdash q_1$}
\RightLabel{$\neg_l$}
\UnaryInfC{$\Gamma, p_1(a), p_1(b), q_1, \neg q_1 \vdash$}
\AxiomC{}
\RightLabel{init}
\UnaryInfC{$\Gamma, p_1(a), p_1(b), q_1 \vdash p_1(b)$}
\RightLabel{$\neg_l$}
\UnaryInfC{$\Gamma, p_1(a), p_1(b), q_1, \neg p_1(b) \vdash$}
\RightLabel{$\vee_l$}
\BinaryInfC{$\Gamma, p_1(a), p_1(b), q_1 \vdash$}
\AxiomC{}
\RightLabel{init}
\UnaryInfC{$\Gamma, p_1(a), p_1(b) \vdash p_1(a)$}
\RightLabel{$\neg_l$}
\UnaryInfC{$\Gamma, p_1(a), p_1(b), \neg p_1(a) \vdash$}
\RightLabel{$\vee_l$}
\BinaryInfC{$\Gamma, p_1(a), p_1(b) \vdash$}
\RightLabel{$\forall_l$}
\UnaryInfC{$\Gamma, \forall x_1.p_1(x_1), p_1(a) \vdash$}
\RightLabel{$\forall_l$}
\UnaryInfC{$\Gamma, \forall x_1.p_1(x_1), \forall x_1.p_1(x_1) \vdash$}
\RightLabel{c_l}
\UnaryInfC{$\Gamma \vdash$}
\end{prooftree}
$$

2.3. *Length measures*

Proofs are measured by number of nodes. Although the sizes of first-order terms are not always negligible, all translations use the same term instantiation as the resolution proof and, therefore, term size has no impact in comparing them.

Definition 2. The *length* $|\psi|$ of a proof ψ is the number of nodes in the proof. In the case of resolution, each node is a clause occurring in the DAG. In the case of sequent calculus, each node is a sequent occurring in the proof tree.

Definition 3. The *length* $|F|$ of a formula F is the number of logical connectives and quantifiers ($\neg$, $\vee$, $\wedge$, $\forall$, $\exists$) occurring in F[‡].

Remarks about Weakening: Note that we use a sequent calculus where weakening is implicit in the init, $\top_r$ and $\bot_l$ rules. In such a calculus, the following lemma can be proven:

Lemma 1. Let φ be a proof of $\Gamma \vdash \Delta$, then there exist proofs φ_l and φ_r of $\Gamma, P \vdash \Delta$ and $\Gamma \vdash \Delta, P$, respectively, such that $|\varphi| = |\varphi_r| = |\varphi_l|$.

Proof sketch We proceed by structural induction on the proof tree. For the base case, when φ contains a single nullary inference (init, $\bot_l$ or $\top_r$) with conclusion $\Gamma \vdash \Delta$, then φ_l and φ_r can be constructed by applying the corresponding nullary inference (init, $\bot_l$ or $\top_r$) with conclusions $\Gamma, P \vdash \Delta$ and $\Gamma \vdash \Delta, P$, respectively. For the inductive cases (for each rule), the induction hypothesis gives us subproofs with the desired extra formula P for the premises and then we reapply the rule normally. As, in each step, the construction of the proofs φ_l and φ_r use the same number of inferences as in the original proof φ, $|\varphi| = |\varphi_r| = |\varphi_l|$. □

[‡] Any other definition of length that is linearly related to the number of connectives and quantifiers (e.g. number of symbols) would suffice and would not change the complexity results presented here. Defining length by number of connectives makes Theorem 5 easier to state and to prove, because the length of a sequent calculus proof is more clearly related to the number of connectives in the formulas in its end-sequent than to the number of symbols.

This lemma is important, because the second and third translations defined in later sections of this paper result in proofs with extra formulas in the contexts of sequents. We shall also use the following notation to denote a proof φ' obtained from φ by adding Π and Θ to all sequents of φ (without length increase, as per Lemma 1):

$$\dfrac{\overset{\varphi}{\Gamma \vdash \Delta}}{\Pi, \Gamma \vdash \Delta, \Theta}\ \mathsf{weak}$$

Note that weak is not an inference rule, but a meta-notation that allows us to write proofs omitting context formulas.

2.4. *Complexity*

In this paper, proof complexity is analyzed using the asymptotic notations O or Ω. Since we are dealing with proof length, the assumption that the functions are non-negative in the definition below is reasonable.

Definition 4. Let $f(x)$ and $g(x)$ be two non-negative functions. We say that $f(x) \in O(g(x))$ iff there exists x_0 and $k > 0$ such that $f(x) \leq k \times g(x)$ for every $x \geq x_0$.

Intuitively, $f(x) \in O(g(x))$ means that the function f grows *at most* as fast as g. In other words, g is an *upper bound* on the growth rate of f.

Definition 5. Let $f(x)$ and $g(x)$ be two functions. We say that $f(x) \in \Omega(g(x))$ iff there exists x_0 and $k > 0$ such that $f(x) \geq k \times g(x)$ for every $x \geq x_0$.

Therefore, Ω is the dual of O: $f(x) \in \Omega(g(x))$ means that f grows *at least* as fast as g. Hence g is a *lower bound* on the growth rate of f.

Our analyses are comparing proof length, and in many cases we use the *worst* possible proof. This means the proof that would provide the highest growth rate. When using Ω, we have a lower bound for the growth rate of the worst case. When using O, we have an upper bound for all cases.

3. First Translation: Resolutions as Cuts on the Resolved Literal

The translation described in this section is basically the one defined by the algorithm from (Hetzl et al. 2013). One notable difference is that we consider the problem to be in CNF. Note that there is no complexity analysis in (Hetzl et al. 2013), and the translation is presented as a pseudo-code without a formal definition.

3.1. *Translation*

Let $\mathcal{R}$ be a resolution refutation of a set of clauses $C_1^*, \ldots, C_n^*$ with free variables. The translation defined will transform $\mathcal{R}$ into an **LK** proof of the sequent $C_1, \ldots, C_n \vdash$, where C_k is the universal closure of C_k^*. This is obtained by using the resolution refutation as a skeleton for the sequent calculus proof, where each clause is represented as a sequent and each resolution step is interpreted as an atomic cut. In order to do this, we must ground the resolution DAG (because the cut rule does not allow unification as the resolution rule does). In turn, grounding requires expansion of the DAG into a tree, because a DAG node that is used as a premise more than once may have to be distinctly instantiated more than once.

Definition 6. Let $\mathcal{R}$ be a resolution refutation DAG, we denote by $\widehat{\mathcal{R}}$ the representation of $\mathcal{R}$ as a ground derivation tree in the resolution calculus (Definition 1). This is achieved in two steps. First the DAG is transformed into a tree by duplicating sub-graphs with more than one parent. Grounding is achieved by propagating the m.g.u. computed at each rule application upwards, i.e., each resolution step is rewritten as:

$$\dfrac{\overset{\rho_1}{C_1 \vee A \vee C_2} \quad \overset{\rho_2}{D_1 \vee \overline{A'} \vee D_2}}{(C_1 \vee C_2 \vee D_1 \vee D_2)\sigma}\ R \quad \rightsquigarrow \quad \dfrac{\overset{\rho_1\sigma}{(C_1 \vee A \vee C_2)\sigma} \quad \overset{\rho_2\sigma}{(D_1 \vee \overline{A'} \vee D_2)\sigma}}{(C_1 \vee C_2 \vee D_1 \vee D_2)\sigma}\ R$$

Therefore, on the ground resolution, all m.g.u.s computed below a clause will be applied to it compositionally.

Theorem 1. In the worst case, $|\widehat{\mathcal{R}}| \in \Omega(2^{|\mathcal{R}|})$.

Proof. The $\widehat{\cdot}$ transformation expands the DAG into a tree, and it is well-known that certain DAGs (e.g. the one shown in Example 1) result in exponentially larger trees. □

The exponential blow-up is not a defect of $\widehat{\cdot}$. As the following stronger theorem states, any grounding operation must suffer an exponential blow-up.

Theorem 2. There is a sequence of clause sets S_n admitting DAG resolution refutations $\mathcal{R}_n$ with $|\mathcal{R}| \in O(n)$ such that any ground resolution refutation $\mathcal{R}'_n$ of a grounding of S_n is such that $|\mathcal{R}'_n| \in \Omega(2^n)$.

Proof. Let S_n be the sequence of clause sets described in Example 1. It admits DAG resolution refutations of linear size, as shown in Example 1. A grounding of S_n would require instantiating $p_1(x_1)$ to $p_1(a)$ and $p_1(b)$, and $p_1(x_1) \vee p_2(x_2)$ to $p_1(a) \vee p_2(a)$, $p_1(a) \vee p_2(b)$, $p_1(b) \vee p_2(a)$ and $p_1(b) \vee p_2(b)$, and so on. The clause/node $p_1(x_1) \vee \ldots \vee p_n(x_n)$ would need 2^n instances. □

Definition 7. Let C be a clause $\neg A_1 \vee \ldots \vee \neg A_n \vee B_1 \vee \ldots \vee B_m$ where A_i $(1 \leq i \leq n)$ and B_k $(1 \leq k \leq m)$ are atoms. Then $seq(C)$ is the sequent $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$.

Definition 8. Let $\mathcal{R}$ be a resolution refutation DAG. We define $seq(\mathcal{R})$ as the **LK** derivation obtained by taking $\widehat{\mathcal{R}}$ and transforming each clause C into $seq(C)$, and interpreting resolution and factoring inferences as cut and contraction, respectively.

Observe that the cuts obtained from resolution inferences will be multiplicative. Nevertheless, we can use additive cuts instead due to the weakening lemma (Lemma 1).

Note that $seq(\mathcal{R})$ is an **LK** derivation of the empty sequent $\cdot \vdash \cdot$ from non-tautological axioms. This structure is transformed into the desired proof of $C_1, \ldots, C_n \vdash$ via the operation of *context product*. In the definition below, the notation $\circ$ denotes merging of sequents, i.e. $(\Gamma \vdash \Delta) \circ (\Lambda \vdash \Pi)$ is $\Gamma, \Lambda \vdash \Delta, \Pi$.

Definition 9 (Context product). Let T be a sequent and φ be an **LK** derivation with end-sequent S such that no free variable in T occurs as eigen-variable in φ. We define the *context product* $T \star \varphi$ (which yields a derivation of $T \circ S$) inductively:

— If φ consists of a leaf, then $T \star \varphi$ is composed of one sequent: $T \circ S$.

— If φ ends with a unary rule ξ:

$$\cfrac{\begin{matrix}\varphi' \\ S'\end{matrix}}{S}\ \xi$$

then $T \star \varphi$ is defined as:

$$\cfrac{\begin{matrix}T \star \varphi' \\ T \circ S'\end{matrix}}{T \circ S}\ \xi$$

Since T does not contain free variables which are eigen-variables of φ, the context product is well defined also if ξ is $\forall_r$ or $\exists_l$, although this case does not occur in our application since formulas need to be skolemized for using resolution.

— If φ ends with a binary rule ξ:

$$\cfrac{\begin{matrix}\varphi_1 \\ S_1\end{matrix} \quad \begin{matrix}\varphi_2 \\ S_2\end{matrix}}{S}\ \xi$$

then $T \star \varphi$ is defined as:

$$\cfrac{\begin{matrix}T \star \varphi_1 \\ T \circ S_1\end{matrix} \quad \begin{matrix}T \star \varphi_2 \\ T \circ S_2\end{matrix}}{T \circ S}\ \xi$$

Since T is part of the context, and all binary rules are additive in the calculus considered, their formulas are copied to both premises. S_1 and S_2 differ from each other at most on auxiliary formulas.

The result of $(C_1, \ldots, C_n \vdash) \star seq(\mathcal{R})$ is a derivation of $C_1, \ldots, C_n \vdash$ from $(C_1, \ldots, C_n \vdash) \circ seq(C_i')$ for $i \in \{1, \ldots, n\}$, where C_i' is a ground instance of C_i. These axioms are tautological and can be proved easily.

Theorem 3. Let $C_1, \ldots, C_n$ be the universally closed formulas of a refutable clause set. Let C_i' be a ground instance of one of those formulas. Then the sequent $(C_1, \ldots, C_n \vdash) \circ seq(C_i')$ has an **LK** proof φ such that $|\varphi| \in O(|C_i|)$.

Proof. To see that the sequent is provable, it suffices to observe that all atoms occurring in $seq(C_i')$ occur in the defined sequent in a dual position because of C_i on the left (remember that in $seq(C_i')$ positive atoms are on the right and negative atoms are on the left). A proof of the sequent can be obtained by instantiating the variables properly and decomposing C_i exhaustively, until its atomic parts. Since this formula has $|C_i|$ connectives, this will be the length of the proof. $\square$

All those steps are summarized in the definition of the translation below.

Definition 10. Let $\mathcal{R}$ be a resolution refutation DAG of a clause set $C_1^*, \ldots, C_n^*$. We define $T_L(\mathcal{R})$ as the **LK** proof obtained from $(C_1, \ldots, C_n \vdash) \star seq(\mathcal{R})$, where each C_i is the universal closure of C_i^* and all axioms are proved (according to the **LK** proof provided by Theorem 3).

Example 3. For brevity, we use the clause set of Example 1 for $n = 1$. Let $\mathcal{R}$ be its resolution refutation:

$$\begin{tikzcd}
q_1 \lor \neg p_1(a) \arrow[dr, dash] & & p_1(x_1) \arrow[dl, dash] \arrow[dr, dash] & & \neg q_1 \lor \neg p_1(b) \arrow[dl, dash] \\
& q_1 \arrow[dr, dash] & & \neg q_1 \arrow[dl, dash] & \\
& & \Box & &
\end{tikzcd}$$

According to Definition 8, $seq(\mathcal{R})$ (using additive cuts) is:

$$\dfrac{\dfrac{\vdash p_1(a), q_1 \qquad p_1(a) \vdash q_1}{\vdash q_1}\,\mathsf{cut} \qquad \dfrac{q_1 \vdash p_1(b) \qquad p_1(b), q_1 \vdash}{q_1 \vdash}\,\mathsf{cut}}{\vdash}\,\mathsf{cut}$$

Let $\Gamma = \{\neg q_1 \lor \neg p_1(b), q_1 \lor \neg p_1(a), \forall x.p_1(x)\}$. Next, we compute the context product (Definition 9) $(\Gamma \vdash) \star seq(\mathcal{R})$, resulting in:

$$\dfrac{\dfrac{\Gamma \vdash p_1(a), q_1 \qquad \Gamma, p_1(a) \vdash q_1}{\Gamma \vdash q_1}\,\mathsf{cut} \qquad \dfrac{\Gamma, q_1 \vdash p_1(b) \qquad \Gamma, p_1(b), q_1 \vdash}{\Gamma, q_1 \vdash}\,\mathsf{cut}}{\Gamma \vdash}\,\mathsf{cut}$$

Proofs of the leaves are straighforward:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, p_1(a) \vdash p_1(a), q_1}\,\mathsf{init}}{\Gamma \vdash p_1(a), q_1}\,\forall_l \qquad \dfrac{\dfrac{}{\Gamma, p_1(a), q_1 \vdash q_1}\,\mathsf{init} \qquad \dfrac{\dfrac{}{\Gamma, p_1(a) \vdash p_1(a), q_1}\,\mathsf{init}}{\Gamma, p_1(a), \neg p_1(a) \vdash q_1}\,\neg_l}{\Gamma, p_1(a) \vdash q_1}\,\lor_l}{\Gamma \vdash q_1}\,\mathsf{cut} \qquad \begin{matrix}\varphi \\ \Gamma, q_1 \vdash\end{matrix}}{\Gamma \vdash}\,\mathsf{cut}$$

Where φ is:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, q_1, p_1(b) \vdash p_1(b)}\,\mathsf{init}}{\Gamma, q_1 \vdash p_1(b)}\,\forall_l \qquad \dfrac{\dfrac{\dfrac{}{\Gamma, p_1(b), q_1 \vdash q_1}\,\mathsf{init}}{\Gamma, p_1(b), q_1, \neg q_1 \vdash}\,\neg_l \qquad \dfrac{\dfrac{}{\Gamma, p_1(b), q_1 \vdash p_1(b)}\,\mathsf{init}}{\Gamma, p_1(b), q_1, \neg p_1(b) \vdash}\,\neg_l}{\Gamma, p_1(b), q_1 \vdash}\,\lor_l}{\Gamma, q_1 \vdash}\,\mathsf{cut}$$

3.2. *Complexity*

Theorem 4. If $\mathcal{R}$ is a resolution refutation, then $|T_L(\mathcal{R})| \in \Omega(2^{|\mathcal{R}|})$ in the worst case.

Proof. The first step of $T_L(\mathcal{R})$ consists of obtaining $seq(\mathcal{R})$, which requires expanding the DAG. According to Theorem 1, this operation may cause an exponential blow-up in the length of the proof. □

4. Second Translation: Resolutions as Cuts on the Resolvent

The second translation of resolution to sequent calculus analyzed here is essentially the one used in the *foundational proof certificates* (FPC) framework for checking resolution proofs (Chihani et al. 2017), with only one minor difference. Whereas the FPC framework uses the one-sided polarized focused calculus LKF (Miller 2017*b*), here the two-sided calculus for classical logic without focusing (as defined in Section 2.2) is used.

4.1. *Translation*

This translation also interprets resolution steps as cuts, but this time the cut formula is the resolvent, including factoring, instead of the resolved atom. The key idea is that each resolution (plus factoring) step deriving a clause C^* from clauses D^* and E^* can be represented in sequent calculus by a derivation of the following form

$$\dfrac{\overset{\varphi}{\Delta \vdash C} \qquad \overset{\vdots}{\Delta, C \vdash}}{\Delta \vdash}\; cut$$

where C, D, E denote the universal closure of clauses C^*, D^*, E^* and Δ is a set of formulas containing D and E. Note that this cut can be quantified, as C^* may contain free variables. $\Delta \vdash C$ is provable (not surprisingly, because C^* is derived from D^* and E^*, which are in Δ and resolution is sound and sequent calculus is complete), and the construction of its proof φ is explained in the demonstration of Theorem 5. On the right branch, the same construction is repeated for the next resolvent, and this continues until the empty clause (i.e. $\perp$) is reached, in which case the right branch can be closed by the rule $\perp_l$. The translation procedure based on this idea is formally defined below.

Definition 11. Let $\mathcal{R}$ be a resolution refutation of the clause set $C_1^*, \ldots, C_n^*$, with resolvents (plus factoring) $C_{n+1}^*, \ldots, C_{n+m}^*$ and $C_{n+m}^* = \perp$. Letting Γ be the set of universally closed clauses $C_1, \ldots, C_n$, the sequence of proofs ψ_j $(0 \leq j < m)$ is defined as:

$$\dfrac{\overset{\varphi_{j+1}}{\Gamma, \ldots, C_{n+j} \vdash C_{n+j+1}} \qquad \overset{\psi_{j+1}}{\Gamma, \ldots, C_{n+j}, C_{n+j+1} \vdash}}{\Gamma, \ldots, C_{n+j} \vdash}\; cut$$

with φ_{j+1} being any linearly sized proof of $\Gamma, \ldots, C_{n+j} \vdash C_{n+j+1}$ (cf. Theorem 5) and ψ_m being defined as:

$$\dfrac{}{\Gamma, C_{n+1}, \ldots, C_{n+m} \vdash}\; \perp_l$$

Finally, $T_R(\mathcal{R})$ is defined as the sequent calculus proof ψ_0 of $\Gamma \vdash$.

The existence of the linearly sized proofs φ_k needed in Definition 11 is shown in the following theorem.

Theorem 5. Let C_1^* and C_2^* be two clauses that resolve to C_3^*, and let C_i denote the universal closure of a clause C_i^*. Then the sequent $C_1, C_2 \vdash C_3$ has an **LK** proof φ such that $|\varphi| \in O(|C_1| + |C_2| + |C_3|)$.

Proof. The proof φ can be constructed in a bottom-up manner as follows. Begin by instantiating the quantified variables of C_3 and decomposing C_3 until only atoms are left. Then instantiate the variables of C_1 and C_2 using either the eigen-variable used for C_3 or terms used in the unifier of the resolution step that derives C_3^* from C_1^* and C_2^*. Finally, apply $\vee_l$ to C_1 and then to C_2 exhaustively. Note that after C_1 is completely decomposed, all branches will be closed (the dual atom is available from C_3), except the one that contains the resolved literal. This is continued by the decomposition of C_2 and eventually the dual of the resolved atom will be in the sequent.

The total number of nodes in this proof is equal to the number of logical connectives and quantifiers occurring in C_1, C_2 and C_3. Therefore, $|\varphi| \in O(|C_1| + |C_2| + |C_3|)$. $\square$

Due to the weakening lemma (Lemma 1), the sequent $\Gamma, C_1, C_2 \vdash \Delta, C_3$ is provable.

Example 4. We translate part of the proof from Example 1, namely:

$$\begin{tikzcd}
 & q_2 \vee p_1(x_1) \arrow[rd, dash] & & \neg q_2 \vee p_1(x_1) \arrow[ld, dash] & \\
 & & p_1(x_1) \vee p_1(x_1) \arrow[d, dash] & & \\
q_1 \vee \neg p_1(a) \arrow[rd, dash] & & p_1(x_1) \arrow[ld, dash] \arrow[rd, dash] & & \neg q_1 \vee \neg p_1(b) \arrow[ld, dash] \\
 & q_1 \arrow[rd, dash] & & \neg q_1 \arrow[ld, dash] & \\
 & & \square & &
\end{tikzcd}$$

Let $\Gamma = \{q_1 \vee \neg p_1(a), \neg q_1 \vee \neg p_1(b), \forall x.(q_2 \vee p_1(x)), \forall x.(\neg q_2 \vee p_1(x)), \}$. According to Definition 11, ψ_0 is:

$$\cfrac{\overset{\varphi_1}{\Gamma \vdash \forall x.p_1(x)} \qquad \overset{\psi_1}{\Gamma, \forall x.p_1(x) \vdash}}{\Gamma \vdash}\ \mathsf{cut}$$

Where φ_1 is the proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, q_2 \vdash p_1(\alpha), q_2}\,\textsf{init}}{\Gamma, q_2, \neg q_2 \vdash p_1(\alpha)}\,\neg_l \quad \dfrac{}{\Gamma, q_2, p_1(\alpha) \vdash p_1(\alpha)}\,\textsf{init}}{\Gamma, q_2, \neg q_2 \vee p_1(\alpha) \vdash p_1(\alpha)}\,\vee_l \quad \dfrac{}{\Gamma, p_1(\alpha), \neg q_2 \vee p_1(\alpha) \vdash p_1(\alpha)}\,\textsf{init}}{\dfrac{\dfrac{\Gamma, q_2 \vee p_1(\alpha), \neg q_2 \vee p_1(\alpha) \vdash p_1(\alpha)}{\Gamma \vdash p_1(\alpha)}\,\forall_l}{\Gamma \vdash \forall x.p_1(x)}\,\forall_r}\,\vee_l$$

Observe that the factoring step is considered as part of the resolution (cf. (Chihani et al. 2017, Section 7.3)). ψ_1 begins with a cut on another resolvent (φ_2 is explicit):

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, p_1(a), q_1 \vdash q_1}\,\textsf{init} \quad \dfrac{\dfrac{}{\Gamma, p_1(a) \vdash q_1, p_1(a)}\,\textsf{init}}{\Gamma, p_1(a), \neg p_1(a) \vdash q_1}\,\neg_l}{\Gamma, p_1(a) \vdash q_1}\,\vee_l}{\Gamma, \forall x.p_1(x) \vdash q_1}\,\forall_l \quad \begin{matrix}\psi_2 \\ \Gamma, \forall x.p_1(x), q_1 \vdash\end{matrix}}{\Gamma, \forall x.p_1(x) \vdash}\,\textsf{cut}$$

ψ_2 continues as:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, p_1(b), q_1 \vdash \neg q_1, q_1}\,\textsf{init}}{\Gamma, p_1(b), q_1, \neg q_1 \vdash \neg q_1}\,\neg_l \quad \dfrac{\dfrac{}{\Gamma, p_1(b), q_1 \vdash \neg q_1, p_1(b)}\,\textsf{init}}{\Gamma, p_1(b), q_1, \neg p_1(b) \vdash \neg q_1}\,\neg_l}{\Gamma, p_1(b), q_1 \vdash \neg q_1}\,\vee_l}{\Gamma, \forall x.p_1(x), q_1 \vdash \neg q_1}\,\forall_l \quad \begin{matrix}\psi_3 \\ \Gamma, \forall x.p_1(x), q_1, \neg q_1 \vdash\end{matrix}}{\Gamma, \forall x.p_1(x), q_1 \vdash}\,\textsf{cut}$$

Finally, ψ_3 cuts on $\perp$, corresponding to the empty clause:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, \forall x.p_1(x), q_1 \vdash \perp, q_1}\,\textsf{init}}{\Gamma, \forall x.p_1(x), q_1, \neg q_1 \vdash \perp}\,\neg_l \quad \dfrac{}{\Gamma, \forall x.p_1(x), q_1, \neg q_1, \perp \vdash}\,\perp_l}{\Gamma, \forall x.p_1(x), q_1, \neg q_1 \vdash}\,\textsf{cut}$$

Observe how, even though the clause $p_1(x)$ was used twice in the resolution refutation, we only need to cut on it once in the sequent calculus proof. Since it persists in the context, it can be used to prove the left branch of the cuts on both q_1 and $\neg q_1$.

4.2. *Complexity*

In this translation, the resolution refutation DAG does not need to be expanded. Each resolution step in the DAG is translated to a single cut in the sequent calculus proof. Resolvents become universally closed clauses in the antecedent of the sequent in the right branch of the proof being constructed, which remain in the context and can be reused as many times as needed. Consequently, the length of the sequent calculus proof is polynomial in the length of the DAG resolution refutation.

Theorem 6. Let $\mathcal{R}$ be a DAG resolution refutation. Then $|T_R(\mathcal{R})| \in O(|\mathcal{R}|^2)$ in the worst case.

Proof. The DAG resolution refutation $\mathcal{R}$ contains at most $O(|\mathcal{R}|)$ resolution steps (each node is either an input clause or the result of a resolution step). In the result of $T_R(\mathcal{R})$, each of these steps will be a cut, whose left branch has a proof of size at most $3 * k$, where k is the size of the biggest universally closed clause in $\mathcal{R}$. It is known that $|k| \in O(|\mathcal{R}|)$ (see (Ben-Sasson & Wigderson 2001)). Consequently, $|T_R(\mathcal{R})| \in O(|\mathcal{R}|^2)$. □

5. Third Translation: Resolutions as Axioms

The third translation defined and analyzed here is inspired by (and essentially the same as) the translation used in the proof of relative soundness of Resolution Modulo (i.e. Extended Narrowing and Resolution, more precisely) with respect to (cut-free) Sequent Calculus Modulo in (Hermant 2010) (which was itself inspired by works on the inverse method (Maslov 1964, Degtyarev & Voronkov 2001, Mints 1990, 1993)). For the sake of simplicity, deduction modulo is left aside, as it is (like focusing, in the case of the second translation) outside the scope and unnecessary to the goals of this paper.

5.1. *Translation*

The intuition of this translation is the following. Assume two clauses $a \vee b$ and $\neg b \vee c$ are resolved to obtain $a \vee c$. Moreover, let Γ denote a set with the other clauses in the leaves of this resolution refutation. Our goal is to find a proof of $\Gamma, a \vee b, \neg b \vee c \vdash$. This proof begins like this:

$$\dfrac{\dfrac{\dfrac{\Gamma, a \vdash}{\Gamma, a, \neg b \vdash}\,\mathsf{weak} \qquad \dfrac{\dfrac{}{\Gamma, b \vdash b}\,\mathsf{init}}{\Gamma, b, \neg b \vdash}\,\neg_l}{\Gamma, a \vee b, \neg b \vdash}\,\vee_l \qquad \dfrac{\Gamma, c \vdash}{\Gamma, a \vee b, c \vdash}\,\mathsf{weak}}{\Gamma, a \vee b, \neg b \vee c \vdash}\,\vee_l$$

Note how the resolution step on the atom b became an init (a.k.a. axiom) inference with main formula b. Now we need to find proofs for the open leaves. Imagine the fringe of the resolution refutation. If, for a moment, we ignore the resolution of $a \vee b, \neg b \vee c$ into $a \vee c$, the set of "leaves" of the refutation will be $\Gamma, a \vee c$. Since the sequent calculus proof is built by induction on the resolution steps (starting from the lower most one), we know that that there exists a derivation of $\Gamma, a \vee c \vdash$. By invertibility of $\vee_l$, we can obtain derivations of $\Gamma, a \vdash$ and $\Gamma, c \vdash$, which are precisely what we need to finish the proof above.

The definition of this translation is more involved when quantification and factoring are considered, but the idea is essentially the same. It relies on the following lemmas (Hermant 2010, Lemma 14).

Lemma 2. Let C_a^* and C_b^* be two clauses that resolve into C^*, and let π be a proof of $\Gamma, C \vdash \Delta$. Then there exists a proof π' of $\Gamma, C_a, C_b \vdash \Delta$.

Proof. The proof proceeds by structural induction on π.

1 BASE CASE: π consists of an axiom. Then it is either an application of init, $\bot_l$ or $\top_r$. We distinguish two cases:

(a) The main formula is not C.
In this case, π' consists of the same rule applied to the sequent $\Gamma, C_a, C_b \vdash \Delta$.

(b) The main formula is C.
Then the applied rule is either init or $\bot_l$. We treat both these cases separately:

i π ends with $\bot_l$.
Then, without loss of generality, we have that:

$$\begin{aligned} C &= \bot \\ C_a &= \forall\overline{x}.p(\overline{x}) \\ C_b &= \forall\overline{y}.\neg p'(\overline{y}) \end{aligned}$$

With p and p' unifiable by m.g.u. σ. Therefore $p(\overline{x})\sigma = p(\overline{t}) = p'(\overline{t'}) = p'(\overline{y})\sigma$, and π' is constructed by instantiating the variables with the m.g.u.:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, p(\overline{t}), \vdash \Delta, p'(\overline{t'})}\,\mathsf{init}}{\Gamma, p(\overline{t}), \neg p'(\overline{t'}) \vdash \Delta}\,\neg_l}{\Gamma, \forall\overline{x}.p(\overline{x}), \forall\overline{y}.\neg p'(\overline{y}) \vdash \Delta}\,\forall_l$$

ii π ends with init.
Then, without loss of generality, we have that:

$$\begin{aligned} C &= A\sigma \\ C_a &= \forall\overline{x}.(p(\overline{x}) \vee A) \\ C_b &= \forall\overline{y}.\neg p'(\overline{y}) \end{aligned}$$

With p and p' unifiable by m.g.u. σ. Therefore $p(\overline{x})\sigma = p(\overline{t}) = p'(\overline{t'}) = p'(\overline{y})\sigma$, and π' is:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, p(\overline{t}) \vdash \Delta, p'(\overline{t'})}\,\mathsf{init}}{\Gamma, \neg p'(t), p(\overline{t}) \vdash \Delta}\,\neg_l \quad \dfrac{\dfrac{}{\Gamma, A\sigma \vdash \Delta}\,\mathsf{init}}{\Gamma, \neg p'(t), A\sigma \vdash \Delta}\,\mathsf{weak}}{\Gamma, p(\overline{t}) \vee A\sigma, \neg p'(t) \vdash \Delta}\,\vee_l}{\Gamma, \forall\overline{x}.(p(\overline{x}) \vee A), \forall\overline{y}.\neg p'(\overline{y}) \vdash \Delta}\,\forall_l$$

The rightmost application of init is the same one as in π.

2 INDUCTIVE CASE: π ends with an application of ρ. We distinguish three cases:

(a) ρ operates on a formula in Γ or Δ. In this case, the rule is simply duplicated on π', whether it is unary or binary:
Unary ρ:

$$\dfrac{\overset{\varphi}{\Gamma', C \vdash \Delta'}}{\Gamma, C \vdash \Delta}\ \rho \quad \rightsquigarrow \quad \dfrac{\overset{\varphi'}{\Gamma', C_a, C_b \vdash \Delta'}}{\Gamma, C_a, C_b \vdash \Delta}\ \rho$$

Binary ρ:

$$\dfrac{\overset{\varphi_1}{\Gamma_1, C \vdash \Delta_1} \quad \overset{\varphi_2}{\Gamma_2, C \vdash \Delta_2}}{\Gamma, C \vdash \Delta}\ \rho \quad \rightsquigarrow \quad \dfrac{\overset{\varphi_1'}{\Gamma_1, C_a, C_b \vdash \Delta_1} \quad \overset{\varphi_2'}{\Gamma_2, C_a, C_b \vdash \Delta_2}}{\Gamma, C_a, C_b \vdash \Delta}\ \rho$$

By induction hypothesis, φ', φ_1', and φ_2' can be constructed.

(b) ρ is a contraction on C. In this case, the contraction is applied twice to obtain π':

$$\dfrac{\overset{\varphi}{\Gamma, C, C \vdash \Delta}}{\Gamma, C \vdash \Delta}\ c_l \quad \rightsquigarrow \quad \dfrac{\dfrac{\overset{\varphi'}{\Gamma, C_a, C_a, C_b, C_b \vdash \Delta}}{\Gamma, C_a, C_a, C_b \vdash \Delta}\ c_l}{\Gamma, C_a, C_b \vdash \Delta}\ c_l$$

Applying the induction hypothesis once to φ yields a derivation of $\Gamma, C_a, C_b, C \vdash \Delta$. Technically, the IH cannot be applied to this derivation because the lemma changes the structure of the proof, so we lose the guarantee that the inductive measure has decreased. This case can be solved by generalizing the lemma to handle an arbitrary number of occurrences of C. All cases above will proceed analogously. In the present case, the IH could be applied to φ obtaining φ' without problems. The next case would be slightly more complicated, though, since the IH would need to be applied to φ_1 and φ_2. Moreover, the notation would be considerably heavier. We choose to maintain this cleaner proof, so that the process is easier to understand. The interested reader is invited to go through the exercise of generalizing the proof.

(c) ρ is a logical rule operating on C.
Then, without loss of generality, we have that:

$$\begin{aligned} C &= \forall \overline{z}.(D_1\sigma \vee D_2\sigma) \\ C_a &= \forall \overline{x}.(D_1 \vee p(\overline{x})) \\ C_b &= \forall \overline{y}.(D_2 \vee \neg p'(\overline{y})) \end{aligned}$$

Where σ is the m.g.u. of p and p'. Therefore $p(\overline{x})\sigma = p(\overline{t}) = p'(\overline{t'}) = p'(\overline{y})\sigma$.

Since all quantifiers can be introduced at once[§], we can assume π ends with:

$$\frac{\Gamma, D_1\sigma' \vee D_2\sigma' \vdash \Delta}{\Gamma, \forall \overline{z}.(D_1\sigma \vee D_2\sigma) \vdash \Delta}\ \forall_l$$

Where σ' is the composition of σ with the instantiation of the variables in $\overline{z}$. Because of associativity of $\vee$, invertibility of $\vee_l$, and admissibility of weakening (Lemma 1), we can assume the existence of the following proofs:

$$\overset{\varphi_1}{\Gamma, D_1\sigma' \vdash \Delta} \qquad \overset{\varphi_2}{\Gamma, D_2\sigma' \vdash \Delta}$$

Using those proofs, we can construct π':

$$\frac{\dfrac{\dfrac{\overset{\varphi_1}{\Gamma, D_1\sigma' \vdash \Delta}}{\Gamma, D_1\sigma', D_2\sigma' \vee \neg p'(\overline{t'}) \vdash \Delta}\ \mathsf{weak} \qquad \dfrac{\dfrac{\overset{\varphi_2}{\Gamma, D_2\sigma' \vdash \Delta}}{\Gamma, p(\overline{t}), D_2\sigma' \vdash \Delta}\ \mathsf{weak} \qquad \dfrac{\dfrac{}{\Gamma, p(\overline{t}) \vdash \Delta, p'(\overline{t'})}\ \mathsf{init}}{\Gamma, p(\overline{t}), \neg p'(\overline{t'}) \vdash \Delta}\ \neg_l}{\Gamma, p(\overline{t}), D_2\sigma' \vee \neg p'(\overline{t'}) \vdash \Delta}\ \vee_l}{\Gamma, D_1\sigma' \vee p(\overline{t}), D_2\sigma' \vee \neg p'(\overline{t'}) \vdash \Delta}\ \vee_l}{\Gamma, \forall \overline{x}.(D_1 \vee p(\overline{x})), \forall \overline{y}.(D_2 \vee \neg p'(\overline{y})) \vdash \Delta}\ \forall_l$$

□

The proof of the next Lemma for factoring rules is essentially the same as before.

Lemma 3. Let C_a^* be a clause that factors into C^*, and let π be a proof of $\Gamma, C \vdash \Delta$. Then there exists a proof of $\Gamma, C_a \vdash \Delta$.

Proof. The proof proceeds by structural induction on π.

1 Base case: π consists of an axiom. Then it is either an application of init, $\bot_l$ or $\top_r$. We distinguish two cases:

(a) The main formula is not C.
In this case, π' consists of the same rule applied to the sequent $\Gamma, C_a \vdash \Delta$.

(b) The main formula is C.
Then the applied rule must be init, since it is impossible to get the empty clause by factoring.
Then, without loss of generality, we have that:

$$C = p(\overline{t})$$
$$C_a = \forall \overline{x}.(p(\overline{x}) \vee p'(\overline{x}))$$

With p and p' unifiable with m.g.u. σ. Therefore $p(\overline{x})\sigma = p(\overline{t}) = p'(\overline{t'}) = p'(\overline{x})\sigma$,

[§] One way to see this is by *focusing* on the formula.

and π' is:

$$\dfrac{\dfrac{\dfrac{}{\Gamma, p(\overline{t}) \vdash \Delta}\ \mathsf{init} \qquad \dfrac{}{\Gamma, p'(\overline{t'}) \vdash \Delta}\ \mathsf{init}}{\Gamma, p(\overline{t}) \vee p'(\overline{t'}) \vdash \Delta}\ \vee_l}{\Gamma, \forall \overline{x}.(p(\overline{x}) \vee p'(\overline{x})) \vdash \Delta}\ \forall_l$$

Both applications of init are the same as in π, since $p(\overline{t}) = p'(\overline{t'})$.

2 Inductive case: π ends with an application of ρ. We distinguish three cases:

(a) ρ operates on a formula in Γ or Δ. In this case, the rule is simply duplicated on π', whether it is unary or binary:
Unary ρ:

$$\dfrac{\overset{\varphi}{\Gamma', C \vdash \Delta'}}{\Gamma, C \vdash \Delta}\ \rho \quad \rightsquigarrow \quad \dfrac{\overset{\varphi'}{\Gamma', C_a \vdash \Delta'}}{\Gamma, C_a \vdash \Delta}\ \rho$$

Binary ρ:

$$\dfrac{\overset{\varphi_1}{\Gamma_1, C \vdash \Delta_1} \quad \overset{\varphi_2}{\Gamma_2, C \vdash \Delta_2}}{\Gamma, C \vdash \Delta}\ \rho \quad \rightsquigarrow \quad \dfrac{\overset{\varphi_1'}{\Gamma_1, C_a \vdash \Delta_1} \quad \overset{\varphi_2'}{\Gamma_2, C_a \vdash \Delta_2}}{\Gamma, C_a \vdash \Delta}\ \rho$$

By induction hypothesis, φ', φ_1', and φ_2' can be constructed.

(b) ρ is a contraction on C. In this case, the contraction is applied twice to obtain π':

$$\dfrac{\overset{\varphi}{\Gamma, C, C \vdash \Delta}}{\Gamma, C \vdash \Delta}\ c_l \quad \rightsquigarrow \quad \dfrac{\dfrac{\overset{\varphi'}{\Gamma, C_a, C_a, C_b, C_b \vdash \Delta}}{\Gamma, C_a, C_a, C_b \vdash \Delta}\ c_l}{\Gamma, C_a, C_b \vdash \Delta}\ c_l$$

Analogously to the previous case, φ' can be obtained by generalizing the lemma for an arbitrary number of occurrences of C.

(c) ρ is a logical rule operating on C.
Then, without loss of generality, we have that:

$$C = \forall \overline{y}.(D_1\sigma \vee p(\overline{t}) \vee D_2\sigma)$$
$$C_a = \forall \overline{x}.(D_1 \vee p(\overline{x}) \vee p'(\overline{x}) \vee D_2)$$

Where σ is the m.g.u. of p and p'. Therefore $p(\overline{x})\sigma = p(\overline{t}) = p'(\overline{t'}) = p'(\overline{x})\sigma$. Since all quantifiers can be introduced at once[¶], we can assume π ends with:

$$\dfrac{\Gamma, D_1\sigma' \vee p(\overline{t})\sigma' \vee D_2\sigma' \vdash \Delta}{\Gamma, \forall \overline{y}.(D_1\sigma \vee p(\overline{t}) \vee D_2\sigma) \vdash \Delta}\ \forall_l$$

Where σ' is the composition of σ with the instantiation of the variables in $\overline{y}$.

[¶] One way to see this is by *focusing* on the formula.

Because of associativity of $\vee$, invertibility of $\vee_l$, and admissibility of weakening (Lemma 1), we can assume the existence of the following proofs:

$$\overset{\varphi_1}{\Gamma, D_1\sigma' \vdash \Delta} \qquad \overset{\varphi_2}{\Gamma, p(\overline{t})\sigma' \vdash \Delta} \qquad \overset{\varphi_3}{\Gamma, D_2\sigma' \vdash \Delta}$$

Using those proofs, we can construct π':

$$\dfrac{\dfrac{\overset{\varphi_1}{\Gamma, D_1\sigma' \vdash \Delta} \quad \dfrac{\overset{\varphi_2}{\Gamma, p(\overline{t})\sigma' \vdash \Delta} \quad \dfrac{\overset{\varphi_2}{\Gamma, p'(\overline{t'})\sigma' \vdash \Delta} \quad \overset{\varphi_3}{\Gamma, D_2\sigma' \vdash \Delta}}{\Gamma, p'(\overline{t'})\sigma' \vee D_2\sigma' \vdash \Delta}\vee_l}{\Gamma, p(\overline{t})\sigma' \vee p'(\overline{t'})\sigma' \vee D_2\sigma' \vdash \Delta}\vee_l}{\Gamma, D_1\sigma' \vee p(\overline{t})\sigma' \vee p'(\overline{t'})\sigma' \vee D_2\sigma' \vdash \Delta}\vee_l}{\Gamma, \forall\overline{x}.(D_1 \vee p(\overline{x}) \vee p'(\overline{x}) \vee D_2) \vdash \Delta}\forall_l$$

□

Using the construction in the proofs of Lemmas 2 and 3 we can define the following translation.

Definition 12. Let $\mathcal{R}$ be a resolution refutation of the clause set $C_1^*, \ldots, C_n^*$ and let Γ be the set of universally closed clauses $C_1, \ldots, C_n$. We define $T_A(\mathcal{R})$, a cut-free sequent calculus proof of $\Gamma \vdash$, by traversing the rules of $\mathcal{R}$ (bottom-up). We start from the trivial proof:

$$\dfrac{}{\bot \vdash}\bot_l$$

corresponding to the empty clause. This proof is transformed as we traverse $\mathcal{R}$ upwards, depending on the inferences we encounter:

— If the inference is a resolution, then it looks like:

$$\dfrac{C_a \quad C_b}{C}$$

We can assume that there exists a proof of $\Gamma, C \vdash$. By Lemma 2, we can transform this proof into a proof of $\Gamma, C_a, C_b \vdash$.

— If the inference is factoring, then we proceed analogously, only using Lemma 3.

It is important to note that the proofs of Lemmas 2 and 3 are completely constructive apart from one small detail: on the relevant inductive cases, we need to argue the "existence" of proofs due to invertibility of $\vee_l$ and associativity of $\vee$. The examples below show that, since all the proofs in the process are constructed using the same process, finding the relevant sequents and their proofs is straightforward (they are the highlighted ones).

Example 5. Let us apply this translation to a part of our running example, namely:

$$\begin{tikzcd}[arrows=-]
 & & p_1(x_1) \vee p_1(x_1) \arrow[d] & & \\
q_1 \vee \neg p_1(a) \arrow[dr] & & p_1(x_1) \arrow[dl] \arrow[dr] & & \neg q_1 \vee \neg p_1(b) \arrow[dl] \\
 & q_1 \arrow[dr] & & \neg q_1 \arrow[dl] & \\
 & & \square & &
\end{tikzcd}$$

In the proofs below, highlighted sequents indicate that the proofs above them will be used later on.

We start with the trivial proof:

$$\dfrac{}{\bot \vdash}\,\bot_l$$

corresponding to the empty clause, and proceed upwards. The next rule is a resolution on q_1 and $\neg q_1$, hence a proof of $q_1, \neg q_1 \vdash$ must be constructed. This corresponds to case 1(b)i of Lemma 2, which yields π_0:

$$\dfrac{\dfrac{}{q_1 \vdash q_1}\,\mathsf{init}}{q_1, \neg q_1 \vdash}\,\neg_l$$

There is a choice for the next resolution to be processed. We (arbitrarily) chose the one resulting in q_1. The clause q_1 was obtained from $q_1 \vee \neg p_1(a)$ and $\forall x.p_1(x)$ (note that we need the universal closure of the clauses). According to Lemma 2, we can construct a derivation of $q_1 \vee \neg p_1(a), \forall x.p_1(x), \neg q_1 \vdash$ from a derivation of $q_1, \neg q_1 \vdash$ (i.e., π_0). Since the lowermost rule ($\neg_l$) in π_0 does not operate on the resolvent in question, we apply this rule to our new proof first (Lemma 2, case 1a). The next rule (init) does operate on q_1, so, following case 1(b)ii of Lemma 2, we apply $\forall_l$ and $\vee_l$ to the clauses that generated q_1 and close, yielding π_1:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{q_1 \vdash q_1}\,\mathsf{init}}{q_1, p_1(a) \vdash q_1}\,\mathsf{weak} \quad \dfrac{\dfrac{}{p_1(a) \vdash q_1, p_1(a)}\,\mathsf{init}}{\neg p_1(a), p_1(a) \vdash q_1}\,\neg_l}{q_1 \vee \neg p_1(a), p_1(a) \vdash q_1}\,\vee_l}{q_1 \vee \neg p_1(a), \forall x.p_1(x) \vdash q_1}\,\forall_l}{q_1 \vee \neg p_1(a), \forall x.p_1(x), \neg q_1 \vdash}\,\neg_l$$

Note how the new init rule operates on $p_1(a)$, the resolved literal to obtain q_1. We proceed with the resolution of $\neg q_1$. We need to obtain a proof of

$$q_1 \vee \neg p_1(a), \forall x.p_1(x), \forall x.p_1(x), \neg q_1 \vee \neg p_1(b) \vdash$$

from π_1. Since the first rule is already one operating on $\neg q_1$, case 2c of Lemma 2 is applicable. In this particular resolution step, D_1 does not exist so only one application of $\vee_l$ is needed. The leftmost premise is closed using π_1 (note the highlighted sequents). Thus π_2 is:

$$\dfrac{\dfrac{q_1 \lor \neg p_1(a), \forall x.p_1(x), \neg q_1 \vdash}{q_1 \lor \neg p_1(a), \forall x.p_1(x), p_1(b), \neg q_1 \vdash}\ \text{weak} \qquad \dfrac{\dfrac{}{q_1 \lor \neg p_1(a), \forall x.p_1(x), p_1(b) \vdash p_1(b)}\ \text{init}}{q_1 \lor \neg p_1(a), \forall x.p_1(x), p_1(b), \neg p_1(b) \vdash}\ \neg_l}{\dfrac{q_1 \lor \neg p_1(a), \forall x.p_1(x), p_1(b), \neg q_1 \lor \neg p_1(b) \vdash}{q_1 \lor \neg p_1(a), \forall x.p_1(x), \forall x.p_1(x), \neg q_1 \lor \neg p_1(b) \vdash}\ \forall_l}\ \lor_l$$

Observe how $\forall x.p_1(x)$ occurs twice in the end-sequent (and it must), representing the fact that it was used twice in the resolution proof. On the next step, one occurrence of $\forall x.p_1(x)$ is replaced by its premise $\forall x.(p_1(x) \lor p_1(x))$. Let $\Gamma = \{q_1 \lor \neg p_1(a), \neg q_1 \lor \neg p_1(b)\}$. According to Lemma 3, we can construct a proof of:

$$\Gamma, \forall x.p_1(x), \forall x.(p_1(x) \lor p_1(x)) \vdash$$

from π_2. The lowermost rule in π_2 operates on one of the occurrences of $\forall x.p_1(x)$. Let us assume it is on the occurrence that we are considering. Then, case 2c of Lemma 3 is applicable. This situation is simplified due to the fact that D_1 and D_2 are empty, so only one application of $\lor_l$ is needed. The (duplicated) proofs of the premises (φ_2 in the Lemma) are obtained from π_2 (the sequent highlighted in blue). Observe that these are the ones whose existence is argued in the proof of the Lemma. The resulting proof is π_3:

$$\dfrac{\dfrac{\Gamma, \forall x.p_1(x), p_1(b) \vdash \qquad \Gamma, \forall x.p_1(x), p_1(b) \vdash}{\Gamma, \forall x.p_1(x), p_1(b) \lor p_1(b) \vdash}\ \lor_l}{\Gamma, \forall x.p_1(x), \forall x.(p_1(x) \lor p_1(x)) \vdash}\ \forall_l$$

The last step is to transform π_3 into a proof of:

$$\Gamma, \forall x.(p_1(x) \lor p_1(x)), \forall x.(p_1(x) \lor p_1(x)) \vdash$$

Following case 2a of Lemma 3, all rules that do not apply to $\forall x.p_1(x)$ are simply copied in the new proof π_4:

$$\dfrac{\dfrac{\overset{\varphi}{\Gamma, \forall x.(p_1(x) \lor p_1(x)), p_1(b) \vdash} \quad \overset{\varphi}{\Gamma, \forall x.(p_1(x) \lor p_1(x)), p_1(b) \vdash}}{\Gamma, \forall x.(p_1(x) \lor p_1(x)), p_1(b) \lor p_1(b) \vdash}\ \lor_l}{\Gamma, \forall x.(p_1(x) \lor p_1(x)), \forall x.(p_1(x) \lor p_1(x)) \vdash}\ \forall_l$$

φ continues as:

$$\dfrac{\dfrac{\dfrac{\overset{\varphi'}{\Gamma, \forall x.(p_1(x) \lor p_1(x)), \vdash q_1}}{\Gamma, \forall x.(p_1(x) \lor p_1(x)), \neg q_1 \vdash}\ \neg_l}{\Gamma, \forall x.(p_1(x) \lor p_1(x)), p_1(b), \neg q_1 \vdash}\ \text{weak} \qquad \dfrac{\dfrac{}{\Gamma, \forall x.(p_1(x) \lor p_1(x)), p_1(b) \vdash p_1(b)}\ \text{init}}{\Gamma, \forall x.(p_1(x) \lor p_1(x)), p_1(b), \neg p_1(b) \vdash}\ \neg_l}{\Gamma, \forall x.(p_1(x) \lor p_1(x)), p_1(b) \vdash}\ \lor_l$$

φ' is at the point where $\forall_l$ is applied to $\forall x.p_1(x)$ in π_3, which falls under case 2c of Lemma 3. We perform the transformation getting to:

$$\dfrac{\dfrac{\dfrac{q_1 \vee \neg p_1(a), p_1(a) \vdash q_1}{\Gamma, p_1(a), p_1(b) \vdash q_1}\,\mathsf{weak} \qquad \dfrac{q_1 \vee \neg p_1(a), p_1(a) \vdash q_1}{\Gamma, p_1(a), p_1(b) \vdash q_1}\,\mathsf{weak}}{\Gamma, p_1(a) \vee p_1(a), p_1(b) \vdash q_1}\,\vee_l}{\Gamma, \forall x.(p_1(x) \vee p_1(x)), p_1(b) \vdash q_1}\,\forall_l$$

Note that the proofs for the green sequents come from the point of π_3 where we stopped copying the application of rules. Also, since φ was used twice, there are four copies of this sub-proof.

Example 6. We use a clause set C_m that is associated with the complete binary tree of depth m, as described in (Cook & Reckhow 1974, Urquhart 1995). C_m contains 2^m disjunctions of the form[||]

$$\circ p^1 \vee \circ p^2_{\pm} \vee \circ p^3_{\pm\pm} \vee \ldots \vee \circ p^m_{\pm\ldots\pm}$$

where $\circ$ is either empty or $\neg$ and the i-th $\pm$ of $p^k_{\pm\ldots\pm}$ (with $i < k$) is either $+$, if the $\circ$ preceding $p^i_{\pm\ldots\pm}$ is empty, or $-$, if the $\circ$ preceding $p^i_{\pm\ldots\pm}$ is $\neg$. For example[††],

$$C_2 = \{p^1 \vee p^2_+, \neg p^1 \vee p^2_-, p^1 \vee \neg p^2_+, \neg p^1 \vee \neg p^2_-\}$$

For this example, the resolution refutation $\mathcal{R}$ we will use is the following (note that it is actually a tree, since no clause is used more than once):

$$\dfrac{\dfrac{\dfrac{p^1 \vee p^2_+ \quad p^1 \vee \neg p^2_+}{p^1 \vee p^1}\,R}{p^1}\,F \qquad \dfrac{\dfrac{\neg p^1 \vee p^2_- \quad \neg p^1 \vee \neg p^2_-}{\neg p^1 \vee \neg p^1}\,R}{\neg p^1}\,F}{\Box}\,R$$

Let $\Gamma = C_2$ (since this is a propositional clause set, there is no need to universally close the clauses). For the sake of comparison, we present the proof $T_L(\mathcal{R})$ obtained from the first translation:

$$\dfrac{\dfrac{\dfrac{\overset{\varphi_1}{\Gamma \vdash p^1, p^1, p^2_+} \quad \overset{\varphi_2}{\Gamma, p^2_+ \vdash p^1, p^1}}{\Gamma \vdash p^1, p^1}\,\mathsf{cut}}{\Gamma \vdash p^1}\,c_r \qquad \dfrac{\dfrac{\overset{\varphi_3}{\Gamma, p^1, p^1 \vdash p^2_-} \quad \overset{\varphi_4}{\Gamma, p^1, p^1, p^2_- \vdash}}{\Gamma, p^1, p^1 \vdash}\,\mathsf{cut}}{\Gamma, p^1 \vdash}\,c_l}{\Gamma \vdash}\,\mathsf{cut}$$

[||] Parentheses have been omitted from these disjunctions, and the $\vee$ connective is assumed to be left-associative.

[††] Note that, in the clause $p^1 \vee p^2_+$, the subscript of the second disjunct is $+$, because the first disjunct is not negated, whereas in the clause $\neg p^1 \vee p^2_-$, the subscript is $-$, because the first disjunct is negated. For a clause with m disjuncts, the k-th disjunct (for $0 < k \leq m$) will contain $k-1$ subscripted $+$ or $-$ signs. For example: $\neg p^1 \vee p^2_- \vee p^3_{-+}$.

Where $\varphi_1, \varphi_2, \varphi_3, \varphi_4$ are:

φ_1 :

$$\dfrac{\dfrac{}{\Gamma, p^1 \vdash p^1, p^1, p^2_+}\,\text{init} \qquad \dfrac{}{\Gamma, p^2_+ \vdash p^1, p^1, p^2_+}\,\text{init}}{\Gamma \vdash p^1, p^1, p^2_+}\,\vee_l$$

φ_2 :

$$\dfrac{\dfrac{}{\Gamma, p^2_+, p^1 \vdash p^1, p^1}\,\text{init} \qquad \dfrac{\dfrac{}{\Gamma, p^2_+ \vdash p^1, p^1, p^2_+}\,\text{init}}{\Gamma, p^2_+, \neg p^2_+ \vdash p^1, p^1}\,\neg_l}{\Gamma, p^2_+ \vdash p^1, p^1}\,\vee_l$$

φ_3 :

$$\dfrac{\dfrac{\dfrac{}{\Gamma, p^1, p^1 \vdash p^2_-, p^1}\,\text{init}}{\Gamma, p^1, p^1, \neg p^1 \vdash p^2_-}\,\neg_l \qquad \dfrac{}{\Gamma, p^1, p^1, p^2_- \vdash p^2_-}\,\text{init}}{\Gamma, p^1, p^1 \vdash p^2_-}\,\vee_l$$

φ_4 :

$$\dfrac{\dfrac{\dfrac{}{\Gamma, p^1, p^1, p^2_- \vdash p^1}\,\text{init}}{\Gamma, p^1, p^1, p^2_-, \neg p^1 \vdash}\,\neg_l \qquad \dfrac{\dfrac{}{\Gamma, p^1, p^1, p^2_- \vdash p^2_-}\,\text{init}}{\Gamma, p^1, p^1, p^2_-, \neg p^2_- \vdash}\,\neg_l}{\Gamma, p^1, p^1, p^2_- \vdash}\,\vee_l$$

We compute now $T_A(\mathcal{R})$, according to Definition 12 and following the same idea of highlighted sequents from the previous example. We begin with the lowermost resolution step, between p^1 and $\neg p^1$ and obtain π_0:

$$\dfrac{\dfrac{}{p^1 \vdash p^1}\,\text{init}}{p^1, \neg p^1 \vdash}\,\neg_l$$

Next we choose the negative literal $\neg p^1$ and obtain a proof of $p^1, \neg p^1 \vee \neg p^1 \vdash$. We call this proof π_1:

$$\dfrac{p^1, \neg p^1 \vdash \qquad p^1, \neg p^1 \vdash}{p^1, \neg p^1 \vee \neg p^1 \vdash}\,\vee_l$$

We will continue on the factoring of p^1 and construct a proof of $p^1 \vee p^1, \neg p^1 \vee \neg p^1 \vdash$. The last step consists on case 1b of Lemma 3. We call this proof π_2:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{p^1 \vdash p^1}\,\text{init} \qquad \dfrac{}{p^1 \vdash p^1}\,\text{init}}{p^1 \vee p^1 \vdash p^1}\,\vee_l}{p^1 \vee p^1, \neg p^1 \vdash}\,\neg_l \qquad \dfrac{\dfrac{\dfrac{}{p^1 \vdash p^1}\,\text{init} \qquad \dfrac{}{p^1 \vdash p^1}\,\text{init}}{p^1 \vee p^1 \vdash p^1}\,\vee_l}{p^1 \vee p^1, \neg p^1 \vdash}\,\neg_l}{p^1 \vee p^1, \neg p^1 \vee \neg p^1 \vdash}\,\vee_l$$

We now process the resolution step resulting in $\neg p^1 \vee \neg p^1$ and construct a proof of $p^1 \vee p^1, \neg p^1 \vee p^2_-, \neg p_1 \vee \neg p^2_- \vdash$. This is π_3:

$$\dfrac{\dfrac{p^1 \vee p^1, \neg p^1 \vdash}{p^1 \vee p^1, \neg p^1, \neg p_1 \vee \neg p^2_- \vdash}\,\text{weak} \qquad \dfrac{\dfrac{p^1 \vee p^1, \neg p_1 \vdash}{p^1 \vee p^1, p^2_-, \neg p_1 \vdash}\,\text{weak} \qquad \dfrac{\dfrac{}{p^1 \vee p^1, p^2_- \vdash p^2_-}\,\text{init}}{p^1 \vee p^1, p^2_-, \neg p^2_- \vdash}\,\neg_l}{p^1 \vee p^1, p^2_-, \neg p_1 \vee \neg p^2_- \vdash}\,\vee_l}{p^1 \vee p^1, \neg p^1 \vee p^2_-, \neg p_1 \vee \neg p^2_- \vdash}\,\vee_l$$

The last remaining inference is the resolution step resulting in $p^1 \vee p^1$. Using π_3 we construct a proof of $p^1 \vee p^2_+, p^1 \vee \neg p^2_+, \neg p^1 \vee p^2_-, \neg p_1 \vee \neg p^2_- \vdash$, which is the final cut-free proof. Let $\Gamma = \{p^1 \vee p^2_+, p^1 \vee \neg p^2_+\}$. The proof π_4 starts by mimicking the inferences of π_3 until $p^1 \vee p^1$ becomes the main formula:

$$\dfrac{\dfrac{\dfrac{\overset{\varphi}{\Gamma \vdash p^1}}{\Gamma, \neg p^1 \vdash}\,\neg_l}{\Gamma, \neg p^1, \neg p_1 \vee \neg p^2_- \vdash}\,\text{weak} \qquad \dfrac{\dfrac{\dfrac{\overset{\varphi}{\Gamma \vdash p^1}}{\Gamma, \neg p_1 \vdash}\,\neg_l}{\Gamma, p^2_-, \neg p_1 \vdash}\,\text{weak} \quad \dfrac{\dfrac{}{\Gamma, p^2_- \vdash p^2_-}\,\text{init}}{\Gamma, p^2_-, \neg p^2_- \vdash}\,\neg_l}{\Gamma, p^2_-, \neg p_1 \vee \neg p^2_- \vdash}\,\vee_l}{\Gamma, \neg p^1 \vee p^2_-, \neg p_1 \vee \neg p^2_- \vdash}\,\vee_l$$

At this point we use the case 2c of Lemma 2 and continue φ with (expanding Γ):

$$\dfrac{\dfrac{p^1 \vdash p^1}{p^1, p^1 \vee \neg p^2_+ \vdash p^1}\,\text{weak} \qquad \dfrac{\dfrac{p^1 \vdash p^1}{p^2_+, p^1 \vdash p^1}\,\text{weak} \quad \dfrac{\dfrac{}{p^2_+ \vdash p^1, p^2_+}\,\text{init}}{p^2_+, \neg p^2_+ \vdash p^1}\,\neg_l}{p^2_+, p^1 \vee \neg p^2_+ \vdash p^1}\,\vee_l}{p^1 \vee p^2_+, p^1 \vee \neg p^2_+ \vdash p^1}\,\vee_l$$

The final parts of the proof are obtained from π_3 at the highlighted sequents.

5.2. *Complexity*

This translation gives a cut-free proof, but at a high cost in the worst case.

In Lemma 2, a clause C_i is replaced by its premises C_k and C_j regardless of whether they are already in the context or not. This is necessary because it might have been the case that, even if, for example, C_k was already in the context, its quantifiers are being instantiated by other variables (that is the case with $\forall x.p_1(x)$ and its premises on Example 5). At the end, the end-sequent will contain as many formulas as leaves in a grounding of the resolution refutation, which has to be a full *tree* expansion of the DAG, in the worst case.

Even if the resolution refutation is a tree, an exponential blow up may still occur in the presence of factoring, as shown in Example 6. This is so because the subproof ϕ_2 is duplicated in the case of factoring in Lemma 3.

Theorem 7. Let $\mathcal{R}$ be a DAG resolution refutation. Then $|T_A(\mathcal{R})| \in \Omega(2^{|\mathcal{R}|})$ in the worst case.

Proof. To prove this theorem formally, it suffices to exhibit a sequence of proofs $\mathcal{R}_n$ with the desired lengths. Note that the sequence of proofs from Example 1 is such that $|\mathcal{R}_n| \in O(n)$ and $|T_A(\mathcal{R}_n)| \in \Omega(2^n)$, for the reasons discussed above and illustrated in Example 5. Additionally, note that the sequence of proofs from Example 6 is such that $|\mathcal{R}_n| \in O(2^n)$ and $|T_A(\mathcal{R}_n)| \in \Omega(2^{2^n})$ (also an exponential blow-up, but of a different kind). We refer the reader to (Woltzenlogel Paleo 2010) for more technical details about the complexity of the sequence of proofs used in Example 6. □

Observe that the worst case lower bound Ω of translation T_L is the same as T_A. Nevertheless, it is possible to exhibit an exponential separation between T_L and T_A for a particular sequence of proofs.

Theorem 8. There exists a sequence of resolution proofs $\mathcal{R}_n$ such that $|T_L(\mathcal{R}_n)| \in O(|\mathcal{R}_n|)$ whereas $|T_A(\mathcal{R}_n)| \in \Omega(2^{|\mathcal{R}_n|})$.

Proof. Let $\mathcal{R}_n$ be the sequence of proofs defined in Example 6. As $\mathcal{R}_n$ are already trees, no duplications occur in the phase when DAGs are expanded to trees, in the first translation. Therefore, $|T_L(\mathcal{R}_n)| \in O(|\mathcal{R}_n|)$. As discussed in the proof of Theorem 7, $|T_A(\mathcal{R}_n)| \in \Omega(2^{2^n}) = \Omega(2^{|\mathcal{R}_n|})$. □

6. Discussion

In the previous sections, we have surveyed and compared (from a complexity perspective) three different translations from resolution to sequent calculus. The first one translates resolution steps to cuts having (the grounding of) the resolved atom as cut-formula. The second one translates resolution steps to cuts having (a universal closure of) the whole resolvent as the cut-formula. And, finally, the third one translates resolution steps to axiom/init inferences having the resolved atom as main formula.

To ease comparison, all three translations were defined here using the same resolution calculus and the same simple sequent calculus, with neither focusing nor deduction modulo. This is the first time that the first translation is formally defined. And we hope that the re-definition of the second and third translations using simpler calculi will make them more accessible to people working on applications where focusing and deduction modulo are not essential.

Complexity-wise, the second translation is clearly superior to the other two. It avoids the expansion and the worst-case exponential blow-up by using universally quantified cuts. However, it is important to note that a (worst-case exponential) blow-up would occur if these cuts were eliminated (using Gentzen's cut-elimination procedure). This is so because duplications occur whenever a cut inference has to be moved above a sequence of contractions and universal quantifications. Such sequences occur in the translated sequent calculus proofs when a resolvent is used more than once in the DAG resolution refutation.

Nevertheless, the other two translations have their advantages too, which become clear when we look at the contexts in which they were developed. The first translation was developed in the context of proof analysis, where there is an interest in extracting Herbrand sequents (Hetzl et al. 2008, Woltzenlogel Paleo 2008), expansion trees (Miller 2017*a*, Hetzl et al. 2013) and generate potentially interesting new lemmas (Hetzl et al. 2014). For these goals, a sequent calculus proof without quantified cuts is essential. For the third translation, the goal was to prove relative soundness of a resolution calculus with deduction modulo. A soundness proof under the assumption that cut is admissible was already known, but cut-admissibility in sequent calculi with deduction modulo is tricky and depends on the rewrite system. By providing a direct translation to a cut-free sequent calculus, the third translation strengthened the soundness for the resolution calculus with deduction modulo, making it independent of assumptions about the rewrite system.

Finally, we hope that the comparison pursued here will shed some additional light on the debate of whether resolution steps are better seen as cuts or axioms. Two of the three known translations see resolution steps as cuts, albeit as cuts of crucially different kinds; the third translation sees resolutions as axioms and, in (Hermant 2010), the view of resolution steps as cuts is considered to be confusing and misleading.

In our (subjective) view, the first translation is the most straightforward: the resulting cut preserves as much as possible the local structure of the resolution step (i.e. exploiting a natural analogy, a resolution with resolved atom p becomes a cut with cut-formula $p\sigma$ for some σ and with a context that is (an instantiation of) a super-set of the context in the resolution step); however, the need for the substitution σ requires expansion of the DAG and hence breaks the global structure of the proof. The preservation of the local structure is a strong support for the *resolution-as-cut* view. On the other hand, the breaking of the global structure is a clear (although unsurprising) indication that the analogy between resolution steps and cuts is not perfect. This imperfection is closely related to Hermant's observation that resolution is forward-chaining, whereas sequent calculus is backward-chaining. It is resolution's forward-chaining nature that naturally gives rise to non-tree DAG proofs.

The second translation also uses cuts for resolution steps, but in a way that does not exploit the natural analogy between resolution steps and cuts. The whole sequent calculus is used more as a *meta* calculus in this translation, with one premise of the cut storing information that a whole resolvent is derivable from previously derived clauses and the other premise continuing the procedure, now with that resolvent added to the derived clauses. The global structure of the proof is preserved and a shorter polynomially bound sequent calculus proof is obtained, at the cost of having more complex cut-formulas.

In the third translation, a resolution step resolving a literal in a clause and its dual in another clause becomes an axiom/init inference connecting these two literals. However, it is important to note that, in fact, each resolution step may become *several* axiom/init inferences. This one-to-many correspondence speaks against the *resolution-as-axiom* view (although note that the first translation also suffers this problem to a lesser extent). Moreover, from a complexity perspective, the third translation may produce proofs that are exponentially longer than those obtained with the first translation (at least for one particular sequence of proofs, as shown in Theorem 8), whereas it is not known whether the converse would be possible (for some other sequence of proofs).

The relative complexities of the translations and the distinct shape of the cut formulas that they use lead us to conjecture that, for a suitable notion of *proof equality*, all three translations are essentially the same but in different stages of cut-elimination. It seems that: if we start with the second translation and partially eliminate the quantified cuts until only atomic cuts are left, the resulting sequent calculus proof is essentially the same as the proof obtained through the first translation; furthermore, if we then eliminate the atomic cuts completely, the result is essentially the same as the proof obtained through the third translation. In each of the cut-elimination steps (i.e. from quantified cuts to atomic cuts, and then to no cuts), an exponential blow-up in proof length may occur.

Finding a suitable notion of *proof equality* is a major challenge for showing this conjecture. Take for example a proof resulting from the first translation. Indeed it contains only atomic cuts, but they occur at the lower part of the proof. But all known reductive cut-elimination methods push cuts up in the proof and, when used to reduce arbitrary cuts to atomic cuts, will yield a proof with atomic cuts at the upper part of the proof. Thus, applying any known reductive cut-elimination procedure to the proofs obtained by the second translation does *not* result in proofs that are syntactically equal to the proofs obtained with the first translation. Therefore, the conjecture would be trivially false if syntactically equality were taken as the notion of proof equality. For the conjecture to possibly be true, we would certainly need a more sophisticated notion of proof equality, taking the permutatibility of inferences into account[‡‡]. And permutability is only one of many non-trivial details about proofs that we may need to take into account in a definition of proof equality to be able to show this conjecture. In principle, we see that some of the complexity theorems shown here might follow as corollaries from this conjecture. However, given the conjecture's non-triviality, this alternative route to showing the complexity theorems would be unnecessarily indirect and more difficult than the route taken here.

We wonder if, by combining ideas from Examples 5 (exponential blow-up due to DAG shape of the refutation) and 6 (exponential blow-up due to factoring), a sequence of refutations could be constructed for which the length of the proofs obtained through the third translation would be doubly-exponentially longer than the proofs obtained through the second translation.

We conjecture that the worst-case bounds for the first and second translations (Theorems 4 and 6) are tight. We are unsure whether the worst-case bounds for the third translation is tight. If there are resolution proofs of elementary size for sequences of formulas that admit only non-elementarily long cut-free sequent calculus proofs (e.g. (Statman 1979, Orevkov 1982)), then a stronger non-elementary worst-case lower-bound for the third translation should be possible.

Another possible direction for future work is to investigate the complexity of the translations in the average case, to complement the worst-case analysis presented here. An average case analysis would require knowledge of the probabilistic distribution of resolution proofs, which will depend on the proof search method used to generate the resolution proofs and on the application domain. Different resolution search refinements tend to generate proofs of certain shapes more frequently than others[§§], and the average complexity

[‡‡] Notice that focusing provides a notion of equality of proofs modulo *some* permutation of inferences, but more pervasive permutations would be needed to equate a proof with atomic cuts in the bottom and a proof with cuts in the top.

[§§] For example, the CDCL proof search method used by sat-solvers generates long tree-like chains of resolution steps, and only the conclusion of these chains may be used more than once and give rise to non-tree-like proofs.

of the translations will depend on how non-tree-like the generated resolution proofs are. The application domain also matters because problems of certain domains may tend to have easy solutions with tree-like resolution proofs, whereas in other domains (e.g. combinatorial problems) this is not the case. Because of these issues, a theoretical average case analysis would not only be difficult to pursue, but would also be of limited use in practice. An experimental average case analysis focusing on problems of a particular domain and with proofs generated according to particular search method would be feasible and useful in helping to decide which translation to choose in these particular cases.

The complexity analysis presented here considered only first-order logic without equality. Whether our results generalize to the case with equality may depend on how equality is incorporated into the resolution calculus. For example, some smt-solvers (Bouton et al. 2009) generate resolution proofs where equality is handled simply by having leaf clauses that are instances of equality axioms. In this case, the complexity results trivially generalize without any change, because these equality clauses are not different from ordinary clauses from the point of view of the translations. If the resolution calculus incorporates equality reasoning through a paramodulation rule (of which the superposition rule used by many modern first-order theorem provers is a special case), then the worst-case lower bounds proven here still apply, because the sequences of resolution proofs used in the demonstrations of these lower-bounds are still correct proofs in a resolution calculus with paramodulation; they just do not use the paramodulation rule. Nevertheless, one may wonder whether it would be possible to prove higher lower bounds using sequences of proofs that do use the paramodulation rule. The first translation has been defined (Baaz & Leitsch 2011) and implemented (Dunchev et al. 2010, Ebner et al. 2016) for the resolution calculus with paramodulation and into a sequent calculus extended with a paramodulation-like equality rule. Because of the direct one-to-one correspondence between paramodulation and these paramodulation-like sequent calculus rules in the translation, it would not be possible to prove a worse lower-bound for this translation by using paramodulation rules. We conjecture that, as for the first translation, with a suitable choice of target sequent calculus, these translations could be extended to first-order logic with equality without worsening the worst-case complexity lower-bounds.

Acknowledgements: Bruno would like to thank Gilles Dowek for pointing out, back in 2010, that resolution steps could also be translated as axioms, right after a talk about cut-introduction by resolution (Woltzenlogel Paleo 2010) given by Bruno at INRIA-Paris to Gilles Dowek's research team. Giselle would like to thank Sonia Marin for checking some of the proofs. The authors would also like to thank Olivier Hermant, Hans de Nivelle, the anonymous reviewers from PxTP (for useful comments on a shorter version of this paper), and the anonymous reviewers from MSCS (for the careful and detailed check and suggestions for improvement).

References

Baaz, M. & Leitsch, A. (2011), *Methods of Cut-Elimination*, Trends in Logic, Springer.

Ben-Sasson, E. & Wigderson, A. (2001), 'Short Proofs Are Narrow – Resolution Made Simple', *J. ACM* **48**(2), 149–169.

Benzmüller, C., Sultana, N., Paulson, L. C. & Theiß, F. (2015), 'The Higher-Order Prover Leo-II', *Journal of Automated Reasoning* **55**(4), 389–404.

Bouton, T., Caminha B. De Oliveira, D., Déharbe, D. & Fontaine, P. (2009), veriT: An Open, Trustable and Efficient SMT-Solver, *in* 'Proceedings of the 22nd International Conference on Automated Deduction', CADE-22, Springer-Verlag, pp. 151–156.

Chihani, Z., Miller, D. & Renaud, F. (2017), 'A semantic framework for proof evidence', *Journal of Automated Reasoning* **59**(3), 287–330.

Cook, S. & Reckhow, R. (1974), On the Lengths of Proofs in the Propositional Calculus (Preliminary Version), *in* 'Proceedings of the Sixth Annual ACM Symposium on Theory of Computing', STOC '74, ACM, pp. 135–148.

Degtyarev, A. & Voronkov, A. (2001), The Inverse Method, *in* J. A. Robinson & A. Voronkov, eds, 'Handbook of Automated Reasoning', Elsevier and MIT Press, pp. 179–272.

Dunchev, T., Leitsch, A., Libal, T., Weller, D. & Woltzenlogel Paleo, B. (2010), System Description: The Proof Transformation System CERES, *in* J. Giesl & R. Hähnle, eds, 'Automated Reasoning', Springer, pp. 427–433.

Ebner, G., Hetzl, S., Reis, G., Riener, M., Wolfsteiner, S. & Zivota, S. (2016), System Description: GAPT 2.0, *in* 'Proceedings of the 8th International Joint Conference on Automated Reasoning - Volume 9706', Springer-Verlag, pp. 293–301.

Gentzen, G. (1969), Investigations into Logical Deductions, *in* M. E. Szabo, ed., 'The Collected Papers of Gerhard Gentzen', North-Holland, pp. 68–131.

Hermant, O. (2010), 'Resolution is Cut-Free', *Journal of Automated Reasoning* **44**(3), 245–276.

Hetzl, S., Leitsch, A., Reis, G. & Weller, D. (2014), 'Algorithmic introduction of quantified cuts', *Theoretical Computer Science* **549**, 1–16.

Hetzl, S., Leitsch, A., Weller, D. & Woltzenlogel Paleo, B. (2008), Herbrand Sequent Extraction, *in* 'Intelligent Computer Mathematics, 9th Int. Conference, AISC, 15th Symposium, Calculemus, 7th Int. Conference, MKM. Proceedings', pp. 462–477.

Hetzl, S., Libal, T., Riener, M. & Rukhaia, M. (2013), Understanding Resolution Proofs through Herbrand's Theorem, *in* 'Automated Reasoning with Analytic Tableaux and Related Methods: 22nd International Conference, TABLEAUX 2013, Proceedings', Springer, pp. 157–171.

Itegulov, D., Slaney, J. & Woltzenlogel Paleo, B. (2017), Scavenger 0.1: A Theorem Prover Based on Conflict Resolution, *in* 'Automated Deduction – CADE 26', Springer International Publishing, pp. 344–356.

Korovin, K. (2008), iProver – An Instantiation-Based Theorem Prover for First-Order Logic (System Description), *in* 'Automated Reasoning', Springer, pp. 292–298.

Kovács, L. & Voronkov, A. (2013), First-Order Theorem Proving and Vampire, *in* 'Computer Aided Verification', Springer, pp. 1–35.

Maslov, S. J. (1964), *An inverse method of establishing deducibilities in the classical predicate calculus*, Reprinted in Siekmann, Wrightson: Automation of reasoning 1: classical papers on computational logic 1957-1966, 1983, pp. 17–20.

McCune, W. (2005–2010), ‘Prover9 and Mace4’, `http://www.cs.unm.edu/~mccune/prover9/`.

Miller, D. (2013), Foundational proof certificates: making proof universal and permanent, *in* ‘Proceedings of the Eighth ACM SIGPLAN International Workshop on Logical Frameworks & Meta-languages: Theory & Practice, LFMTP’, pp. 1–2.

Miller, D. (2017*a*), Expansion Proofs, *in* B. Woltzenlogel Paleo, ed., ‘Towards an Encyclopaedia of Proof Systems’, 1 edn, College Publications, p. 18.

Miller, D. (2017*b*), Focused LK, *in* B. Woltzenlogel Paleo, ed., ‘Towards an Encyclopaedia of Proof Systems’, 1 edn, College Publications, pp. 75–76.

Mints, G. (1990), Gentzen-type systems and resolution rules part I propositional logic, *in* P. Martin-Löf & G. Mints, eds, ‘COLOG-88: International Conference on Computer Logic Tallinn, USSR, December 12–16, 1988 Proceedings’, Springer, pp. 198–231.

Mints, G. (1993), Gentzen-type systems and resolution rule, part II: predicate logic, *in* J. Oikkonen & J. Väänänen, eds, ‘Proc. of ASL Summer Meeting, Logic Colloquium ’90, volume 2 of Lecture Notes in Logic’, Springer-Verlag, pp. 163–190.

Orevkov, V. P. (1982), ‘Lower bounds for increasing complexity of derivations after cut elimination’, *Journal of Mathematical Sciences* **20**(4), 2337–2350.

Reis, G. (2015), Importing SMT and Connection proofs as expansion trees, *in* ‘Proceedings Fourth Workshop on Proof eXchange for Theorem Proving, PxTP’, pp. 3–10.

Robinson, J. A. (1965), ‘A Machine-Oriented Logic Based on the Resolution Principle’, *Journal of the ACM* **12**(1), 23–41.

Schulz, S. (2013), System Description: E 1.8, *in* K. McMillan, A. Middeldorp & A. Voronkov, eds, ‘Proc. of the 19th LPAR’, Vol. 8312 of *LNCS*, Springer, pp. 735–743.

Statman, R. (1979), ‘Lower bounds on herbrand’s theorem’, *Proceedings of the American Mathematical Society* **75**(1), 104–107.

Urquhart, A. (1995), ‘The complexity of propositional proofs’, *Bulletin of Symbolic Logic* **1**(4), 425–467.

Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M. & Wischnewski, P. (2009), SPASS Version 3.5, *in* R. A. Schmidt, ed., ‘Automated Deduction – CADE-22’, Springer, pp. 140–145.

Woltzenlogel Paleo, B. (2008), *Herbrand Sequent Extraction [M.Sc. Thesis]*, VDM-Verlag.

Woltzenlogel Paleo, B. (2010), Atomic Cut Introduction by Resolution: Proof Structuring and Compression, *in* ‘Logic for Programming, Artificial Intelligence, and Reasoning - 16th International Conference, LPAR-16, Revised Selected Papers’, pp. 463–480.