EPiC Series in Computing
Volume 73, 2020, Pages 423–448
LPAR23. LPAR-23: 23rd International Conference on Logic for Programming, Artificial Intelligence and Reasoning

RAT Elimination

Adrián Rebola-Pardo and Georg Weissenbacher

TU Wien

Abstract

Inprocessing techniques have become one of the most promising advancements in SAT solving over the last decade. Some inprocessing techniques modify a propositional formula in non model-perserving ways. These operations are very problematic when Craig interpolants must be extracted: existing methods take resolution proofs as an input, but these inferences require stronger proof systems; state-of-the-art solvers generate DRAT proofs. We present the first method to transform DRAT proofs into resolution-like proofs by eliminating satisfiability-preserving RAT inferences. This solves the problem of extracting interpolants from DRAT proofs.

1 Introduction

Satisfiability solvers [8] are the cornerstone of most symbolic model checking algorithms [38]. Over the last two decades, huge scalability leaps of SAT solvers enabled the verification of ever more complex hardware designs. The impressive evolution of solving techniques has yet to stop: recent advances such as preprocessing [17, 29], inprocessing [25, 6], and symmetry breaking [1] enable SAT solvers to tackle problems of hitherto unimaginable size [24].

Some model checking algorithms such as bounded model checking [7] immediately benefit from these advancements without requiring adaptation. In the case of interpolation-based model checking [30], a widely-used approach that uses Craig interpolants [13] to compute inductive invariants, the above-mentioned techniques are not readily applicable. Interpolants are extracted from resolution proofs generated by SAT solvers. Operations such as symmetry breaking, however, cannot be expressed in terms of resolution but require stronger proof systems. Hence, many contemporary SAT solvers emit DRAT proofs [23], which owe their name to the resolution asymmetric tautology (RAT) inference [21] and are equivalent in power to extended resolution proofs [26].

DRAT proofs pose a challenge for interpolation. Existing interpolation systems require derived clauses to be implied by the original formula [16, 27], as is the case for resolution proofs. DRAT proofs violate this assumption by infering RAT clauses, which do not preserve the models of (i.e. do not follow from) the premise formula [31, 34], rendering known interpolation systems useless. Moreover, interpolants can be extracted from resolution proofs in polynomial time [32]. Extended resolution (and DRAT) proofs can however be exponentially smaller than resolution proofs [36]; feasible interpolation results for them are unavailable, quite possibly even unattainable [28, 10].

As a consequence, interpolation-based model checkers currently need to forego the use of the latest SAT techniques. To address this, we propose an approach based on isolating and eliminating RATs from DRAT proofs by means of proof transformation (at the cost of a potential increase in size, which, as discussed above, is likely unavoidable). The resulting proofs are Reverse Unit Propagation (RUP) proofs [19], a highly compact representation of resolution proofs, for which interpolation algorithms are readily available [35, 20].

Section 2 covers resolution, RUP, and DRAT proofs. Section 3 explains the idea of clause isolation, our weapon of choice for RAT elimination. Section 4 explains how clause isolation can be achieved by exploiting distributivity of a generalized version of resolution, and Section 5 adapts and enhances distributivity for RUPs. Finally, Section 6 deploys RUP distributivity to eliminate RATs from DRAT proofs. Section 7 discusses related work and concludes the paper.

2 Preliminaries

We consider a countably infinite set of *propositional variables*. A *literal* is either a variable or its negation; we denote the *complement* of a literal l by $\bar{l}$. A *clause* is a finite, complement-free disjunction of literals; we denote clauses by juxtaposition, i.e. we write $xy\bar{z}$ for the clause $x \vee y \vee \bar{z}$. For this paper the condition that clauses do not contain complementary literals is very relevant. We refer to finite sets of literals without this requirement as *generalized clauses*; when we want to emphasize that a generalized clause is complement-free, we will refer to *proper clauses*, and we call generalized clauses containing complementary literals *tautologies*. We denote the unsatisfiable empty clause by $\square$. A *conjunctive normal form (CNF) formula* is a conjunction of clauses.

The *conflict literals* of two generalized clauses C and D are $C \parallel D = \{l \in C \mid \bar{l} \in D\}$. We say that C subsumes D whenever $C \subseteq D$. Given a literal l, we call the *resolvent* of C and D upon l the generalized clause $C \setminus \{l\} \cup D \setminus \{\bar{l}\}$, which we denote by $C \otimes_l D$. Let us highlight that, rather than a variable, we write a literal as the pivot for the resolvent; this makes the order of resolvents important, e.g. $C \otimes_l D = D \otimes_{\bar{l}} C$. We do not impose the usual requirement that $l \in C$ and $\bar{l} \in D$; this greatly simplifies the exposition, and preserves the usual intuition about resolution, as shown by the following result:

Proposition 1. *Let C and D be generalized clauses and l be a literal. Then, $\{C, D\} \vDash C \otimes_l D$. Furthermore, if C and D are proper clauses and $C \parallel D = \{l\}$ holds, the resolvent $C \otimes_l D$ is proper.*

A resolution inference $C \otimes_l D = E$ is called a *merge resolution* [2] whenever $D \setminus \{\bar{l}\} \subseteq C$. Whenever C and D are proper clauses, then in this case we have $C \otimes_l D = C \setminus \{l\}$.

Proposition 1 allows us to define the resolution and subsumption inference rules, shown in Figures 1a and 1b. As above, we do not require any condition on premises of the resolution inference. We call a derivation using these rules a *generalized resolution-subsumption derivation*, and the clauses from the refuted formula are its *proof premises*; in a *(proper) resolution-subsumption derivation*, only proper clauses are involved, and $C \parallel D = \{l\}$ holds for each resolution $C \otimes_l D$. A derivation of $\square$ from premises in F is a *refutation* of F.

RUP proofs In practice, resolution-subsumption derivations are insufficient for the needs of the SAT solving community: generating them at solving time is complicated, and the need to record every inference implies an unacceptable increase in proof size [18]. To alleviate this problem, *Reverse Unit Propagation* (RUP) proofs were developed [19]. Instead of resolution

(a) Subsumption inference

$$\sqsubseteq \frac{C}{C \vee D}$$

(b) Resolution inference upon l

$$l \frac{C \qquad D}{C \otimes_l D}$$

(c) Subsumption-resolution chain

$$\cfrac{\cfrac{\cfrac{\cfrac{\sqsubseteq \dfrac{E_0}{A_0} \qquad E_1}{k_1 \quad A_1} \qquad E_2}{k_2 \quad \ddots}}{k_n \quad A_{n-1} \qquad E_n}}{A_n}$$

Figure 1: Clausal inferences

and subsumption inferences, RUP proofs only allow one homonymous inference rule defined in [19] in terms of unit propagation. Here we introduce RUP clauses from a proof-theoretical perspective; both definitions are equivalent [31].

A *generalized subsumption-resolution chain* is a derivation of the form shown in Figure 1c, where a subsumption inference $E_0 \sqsubseteq A_0$ is optionally followed by any number of iterative resolution inferences $A_{i-1} \otimes_{k_i} E_i = A_i$. *Generalized subsumption-merge chains* additionally require that these resolutions are merges. We simply refer to *subsumption-resolution chains* and *subsumption-merge chains* for the cases where the E_i and A_i are all proper, and $A_{i-1} \| E_i = \{k_i\}$ holds; sometimes we will call these chains *proper* to avoid ambiguity. In the case of a (proper) subsumption-merge chain, the *intermediate clauses* A_i are determined by the *chain premises* E_i and the *derived clause* A_n, so sometimes we will write the chain in a more compact way as $A_n \colon E_0 \otimes_{k_1} E_1 \otimes_{k_2} \ldots \otimes_{k_n} E_n$.

A clause C is a RUP in a CNF formula F whenever C can be derived from F by a subsumption-merge chain (called *subsumption self-subsuming resolution* chains in [31]). RUP proofs are simply lists of RUP clauses; the corresponding subsumption-merge chains are implicitly found by proof-checking procedures in an efficient way [22].

Lemma 1. *Consider a generalized subsumption-merge chain as in Figure 1c. Then, for all* $0 \leq i < j \leq n$*, we have* $A_j \subseteq A_i$*, and furthermore* $A_i \setminus A_j \subseteq \{k_{i+1}, \ldots, k_j\}$.

Proof. It suffices to show that for all $0 < i \leq n$ we have either $A_i = A_{i-1}$ or $A_i = A_{i-1} \setminus \{k_i\}$. We know that $A_i = A_{i-1} \otimes_{k_i} E_i = A_{i-1} \setminus \{k_i\} \cup E_i \setminus \{\overline{k_i}\}$. Because this resolvent is a merge, $E_{i-1} \setminus \{\overline{k_i}\} \subseteq A_{i-1}$ holds. This shows the claim, by considering that one case or the other holds depending on whether $k_i \in E_i$ or not. □

Lemma 2. *Given a subsumption-merge chain as in Figure 1c, we have the following:*

1. *For all* $0 \leq i < j \leq n$*, the underlying variables to literals* k_i *and* k_j *are distinct, i.e.* $k_i \neq k_j \neq \overline{k_i}$.
2. *For all* $0 \leq i < j \leq n$*, the literal* k_j *occurs in* A_i.
3. *For all* $0 < i < j \leq n$*, neither of the literals* k_i *nor* $\overline{k_i}$ *occurs in* E_j *or* A_j.

Proof. These all follow from the observation that, for all $0 \leq i < j \leq n$ we have $A_i = A_j \setminus \{k_{i+1}, \ldots, k_j\}$. To show this, it suffices to prove $A_i = A_{i-1} \setminus \{k_i\}$ for all $0 < i \leq n$. Since

we have a proper subsumption-merge chain, we know that $A_{i-1} \parallel E_i = \{k_i\}$, so in particular $E_i = E_i \setminus \{k_i\}$. Then, because the resolvent $A_{i-1} \otimes_{k_i} E_i$ is a merge, we have

$$A_i = A_{i-1} \otimes_{k_i} E_i = A_{i-1} \setminus \{k_i\} \cup E_i \setminus \{\overline{k_i}\} = A_{i-1} \setminus \{k_i\} \cup E_i \setminus \{\overline{k_i}, k_i\} = A_{i-1} \setminus \{k_i\}$$

We now proceed to show the claims.

1. Because $A_{j-1} \parallel E_j = \{k_j\}$, we have $k_j \in A_{j-1} = A_{i-1} \setminus \{k_i, \ldots, k_{j-1}\}$, which precludes $k_i = k_j$. Furthermore, if $k_j = \overline{k_i}$ held true, we would have $\overline{k_i} \in A_{j-1} \subseteq A_{i-1}$ and $k_i \in A_{i-1}$, which would contradict A_i being proper.
2. Since $A_{j-1} \parallel E_j = \{k_j\}$, we have $k_j \in A_{j-1} \subseteq A_i$.
3. We know that $A_j = A_{i-1} \setminus \{k_i, \ldots, k_j\}$, so this implies $k_i \notin A_j$. Furthermore, since $k_i \in A_{i-1}$, we have $\overline{k_i} \notin A_{i-1} \supseteq A_j$. Now, because the resolvent $A_{i-1} \otimes_{k_i} E_i$ is a merge, we know that $E_j \setminus \{\overline{k_j}\} \subseteq A_{j-1}$. If either k_i or $\overline{k_i}$ occurred in E_j, then since $k_i \neq k_j \neq \overline{k_i}$, we would conclude that this literal occurs in A_j as well, but this would contradict what we have shown before. □

DRAT proofs An extension of RUP proofs, called *Delete Resolution Asymmetric Tautology* (DRAT) [23], was introduced due to shortcomings of RUP proofs when dealing with inprocessing techniques in SAT solving [25]. In addition to RUP inferences, DRAT proofs allow to forget clauses through a *deletion* inference, as well as resolution asymmetric tautology (RAT) [21] inferences. A clause C is a RAT in a CNF formula F if there is a literal $l \in C$ such that the resolvent $C \otimes_l D$ is a RUP clause in F for each $D \in F$; in this case we call C a RAT clause in F upon l. Observe that, in practice, it is sufficient to check that $C \otimes_l D$ is a RUP in F for clauses D containing the literal $\overline{l}$; otherwise the resolvent is subsumed by $D \in F$ (c.f. Example 1).

In general, RAT inferences are not model-preserving, but rather satisfiability-preserving. Although a good understanding of how semantics are preserved through RAT introduction has been developed [31, 34], there is no known method to generate Craig interpolants from satisfiability-preserving proofs, since the invariants maintained by existing methods rely strongly on model-preserving inferences [35, 27].

Strictly speaking, DRAT proofs do not contain themselves the derivation chains. Instead, proof checkers [39, 33] generate *LRAT proofs* which contain them [14]. Furthermore, solvers that directly generate LRAT proofs recently became available, such as `Varisat`[1]. We assume in our work that DRAT proofs are provided with the corresponding chains.

Example 1. Let us consider the unsatisfiable CNF formula containing the following clauses:

$$\begin{array}{cccccccc} \overline{x}yz & x\overline{y}z & xy\overline{z} & \overline{xyz} & \overline{z}uv & z\overline{u}v & zu\overline{v} & \overline{zuv} \\ \overline{x}yw & x\overline{y}w & xy\overline{w} & \overline{xyw} & u\overline{v}\overline{w} & \overline{u}v\overline{w} & \overline{u}\overline{v}w & uvw \end{array}$$

A DRAT refutation for this formula is given by the following clauses and their corresponding subsumption-merge chains:

[1] https://github.com/jix/varisat

xu (RAT upon x)
- $xu \otimes_x \overline{x}yz = yzu$: $xy\overline{w} \otimes_{\overline{w}} uvw \otimes_x \overline{x}yz \otimes_v zu\overline{v}$
- $xu \otimes_x \overline{xyz} = \overline{yz}u$: $x\overline{y}w \otimes_w u\overline{vw} \otimes_x \overline{xyz} \otimes_{\overline{v}} \overline{z}uv$
- $xu \otimes_x \overline{x}yw = yuw$: $zu\overline{v} \otimes_z xy\overline{z} \otimes_x \overline{x}yw \otimes_{\overline{v}} uvw$
- $xu \otimes_x \overline{xyw} = \overline{y}u\overline{w}$: $\overline{z}uv \otimes_{\overline{z}} x\overline{y}z \otimes_x \overline{xyw} \otimes_v u\overline{vw}$

$x\overline{y}$: $\overline{u}v\overline{w} \otimes_v \overline{zuv} \otimes_{\overline{u}} xu \otimes_{\overline{w}} x\overline{y}w \otimes_{\overline{z}} x\overline{y}z$

$z\overline{u}w$: $\overline{uv}w \otimes_{\overline{v}} z\overline{u}v$

x: $z\overline{u}w \otimes_w xy\overline{w} \otimes_{\overline{u}} xu \otimes_z xy\overline{z} \otimes_y x\overline{y}$

$\overline{y}$: $uvw \otimes_v zu\overline{v} \otimes_u z\overline{u}w \otimes_w \overline{xyw} \otimes_z \overline{xyz} \otimes_{\overline{x}} x\overline{y}$

u: $\overline{z}uv \otimes_v u\overline{vw} \otimes_{\overline{w}} \overline{x}yw \otimes_{\overline{z}} \overline{x}yz \otimes_{\overline{x}} x \otimes_y \overline{y}$

$\square$: $\overline{zuv} \otimes_{\overline{v}} \overline{u}v\overline{w} \otimes_{\overline{w}} \overline{x}yw \otimes_{\overline{z}} \overline{x}yz \otimes_{\overline{x}} x \otimes_y \overline{y} \otimes_{\overline{u}} u$

3 RAT elimination through clause isolation

Our goal is to transform a DRAT refutation of a CNF formula F into a RUP refutation of F. To attain this, we will iteratively modify the fragment after the last RAT inference in the proof so that the clause derived in that inference becomes unnecessary. This is done by a process we call *clause isolation*, where we can restrict how a given clause is used in a proof. After eliminating all RAT inferences and deletions from the input proof, we obtain a RUP refutation of F.

Given a clause C, a literal $l \in C$ and a refutation π of a CNF formula $F \cup \{C\}$, *isolating* a clause C in the proof π upon l refers to constructing a refutation of $F \cup \{C \otimes_l D \mid D \in F\}$. In other words, our goal is to obtain a refutation of $F \cup \{C\}$ such that C is used in the proof exclusively by immediately resolving it with a premise from F upon l. As we will see in Section 4, we can always isolate a clause in a RUP refutation.

Let us assume that we know a procedure for clause isolation; we explain now how to lift it into a RAT elimination procedure. Consider a DRAT refutation π of a CNF formula F, given by instructions $I_1, \ldots, I_n$ where $I_n = \square$. The I_i may be either clause introductions (RUP or RAT inferences), or clause deletions. Let us assume that I_i is the last RAT inference in the proof, and the CNF formula accumulated before I_i is F_{i-1}. The instruction I_i introduces a clause C as a RAT in F_i upon a literal l, and so we can split the DRAT refutation π as $\pi' C \sigma$, where σ only contains RUP introductions and deletions, and is itself a refutation of $F_{i-1} \cup \{C\}$.

Our method now proceeds in three stages. First, we remove deletions from σ; this is possible because, unlike RAT inferences, RUP inferences are monotonic [31]. The result is still a RUP refutation of $F_{i-1} \cup \{C\}$. We then apply a clause isolation procedure, which yields a RUP refutation σ' of the CNF formula $F_{i-1} \cup \{C \otimes_l D \mid D \in F_{i-1}\}$. Finally, in order to bridge the gap between F_{i-1} and the formula above, we derive the missing resolvents $C \otimes_l D$, which are RUPs by the definition of RAT clauses.

This procedure, shown in Figure 2 eliminates a single RAT from the DRAT refutation. The prefix π' of the proof is preserved, so no new RATs are created. All RATs in a proof can be iteratively eliminated with this method, and so a RUP refutation is obtained and an interpolant can then be generated.

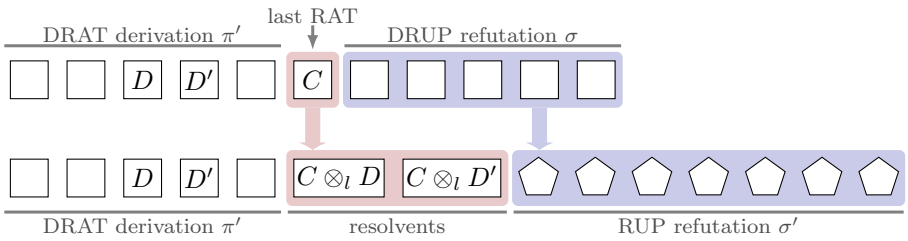


Figure 2: Elimination of a single RAT using a clause isolation procedure

4 Clause isolation using distributivity

The method explained in Section 3 for RAT elimination requires a procedure for clause isolation over a RUP proof. These proofs can be regarded as generalized resolution-subsumption proofs; we present a method that, by iteratively performing correctness-preserving proof transformations, generates a clause-isolated generalized resolution-subsumption proof.

In the following we fix a CNF formula F, and a clause C to be isolated upon a literal l in a generalized subsumption-resolution refutation π of $F \cup \{C\}$. Given a clause D derived from C (possibly using clauses from F), there is a descending path in the proof from C to D. If C is not isolated in π upon l, then there must exist one such path of non-zero length (i.e. involving at least one inference) in which the literal l is never eliminated. This motivates the notion of impure clauses: we call all those clauses derived from C for which the literal l has not been eliminated *impure*, for they are the reason why C is not isolated in π upon l. Formally, we recursively classify clause occurrences in the proof as *pure* or *impure* as follows:

- A proof premise D is pure if and only if $D \neq C$.
- For subsumption inferences $D \sqsubseteq E$, the clause E is pure if and only if D is pure.
- For resolution inferences $D \otimes_k D' = E$ with $l \neq k \neq \bar{l}$, the clause E is pure if and only if both D and D' are.
- For resolution inferences $D \otimes_l D' = E$, the clause E is pure if and only if D' is pure.
- For resolution inferences $D \otimes_{\bar{l}} D' = E$, the clause E is pure if and only if D is pure.

From this definition it is easy to see that the only way we can derive a pure clause from an impure one is by the resolution inferences from cases 4 and 4. A *left-purifier* is a resolution inference $D \otimes_l D' = E$ such that D is impure; and a *right-purifier* is a resolution inference $D \otimes_{\bar{l}} D' = E$ such that D' is impure; we refer to both of them as *purifiers*. For each purifier X given by $D \otimes_k D' = E$, we define its *purification depth* $\mathrm{dp}(X)$ as the sum of the number of ancestors of D and D'; and then the *purification measure* of the refutation π is given by the multiset of the non-zero purification depths of purifiers in π, which we denote by $\mathrm{ms}(\pi)$. When considered under the multiset ordering, the purification measure of a refutation gauges how far is C from being isolated upon l. In particular the following result is straightforward:

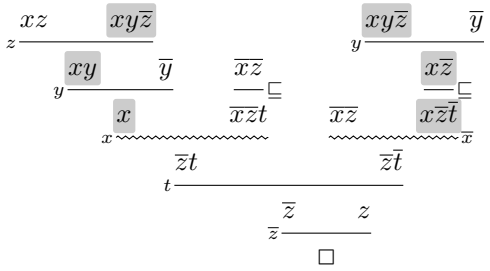
Proposition 2. *The clause C is isolated in a generalized resolution-subsumption refutation of $F \cup \{C\}$ upon l if and only if its purification measure is $\emptyset$.*

Proof. We first show "only if". If C is isolated upon l in the refutation, we can easily show by induction that its only impure clauses are occurrences of C as a proof premise. Hence, its only

left-purifiers are inferences of the form $C \otimes_l D = E$ where C is a proof premise; and because C is isolated in the refutation upon l, the clause D must also be a proof premise. We can thus conclude that the purification depth of each left-purifier in the proof is 0; we can analogously show the same for right-purifiers. Therefore, the purification measure of the refutation is $\emptyset$.

We now show "if". Observe that if the purification measure of the proof upon l is $\emptyset$, then all its purifiers have purification depth 0. In particular, every purifier is a resolution inference whose two premises are proof axioms. Consider now an occurrence of C as a proof premise, which needs to be impure. The clause C is not the empty clause, so another clause E is derived from it by an inference in the proof, and E then needs to be pure. According to the definition of pure and impure clauses, this can only happen via a left-purifier $C \otimes_l D = E$ or via a right-purifier $D \otimes_{\overline{l}} C = E$. Because such purifiers have purification depth 0, the clause D must be a proof premise too. Since this happens for each occurrence of C as a proof premise, C is isolated in the refutation upon l. □

Example 2. Consider the following refutation π of the formula $F = \{xz, xy\overline{z}, \overline{y}, \overline{xz}, z\}$, and let us take $C = xy\overline{z}$ and $l = x$.



Impure clauses in this refutation have been shaded. Observe that impure "threads" start with C, and then continue down the refutation until the literal x is eliminated by a resolution. This happens at purifiers, which are denoted by zigzag inferences; the purification measure of this proof is then $\{5, 3\}$.

Proposition 2 means that the clause isolation problem can be solved by simply finding a correctness-preserving proof transformation which reduces the purification measure of its input w.r.t. the multiset ordering. Because the multiset ordering induced by a well-founded ordering is itself well-ordered [3], it only takes finitely many iterative applications of this transformation to isolate C upon l. Our method is based in the following result, which we refer to as *distributive laws*:

Theorem 1. *Let D, E' and E'' be generalized clauses, and k, l be literals. Then,*

1. *If $E' \sqsubseteq E''$, then $D \otimes_l E' \sqsubseteq D \otimes_l E''$.*
2. *$D \otimes_l (E' \otimes_k E'') = (D \otimes_l E') \otimes_k (D \otimes_l E'')$.*

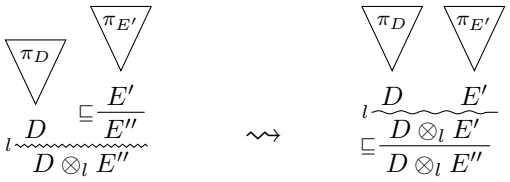
Proof. To show the first claim, let us assume that $E' \subseteq E''$. Then,

$$D \otimes_l E' = D \setminus \{l\} \cup E' \setminus \{\overline{l}\} \subseteq D \setminus \{l\} \cup E'' \setminus \{\overline{l}\} = D \otimes_l E''$$

To show the second identity, we know that

$$D \otimes_l (E' \otimes_k E'') = D \setminus \{l\} \cup E' \setminus \{\overline{l}, k\} \cup E'' \setminus \{\overline{l}, \overline{k}\}$$
$$(D \otimes_l E') \otimes_k (D \otimes_l E'') = (D \setminus \{l, k\} \cup D \setminus \{l, \overline{k}\}) \cup E' \setminus \{\overline{l}, k\} \cup E'' \setminus \{\overline{l}, \overline{k}\}$$

(a) Subsumption rule: a purifier uses the conclusion of a subsumption inference



(b) Resolution rule: a purifier uses the conclusion of a resolution inference

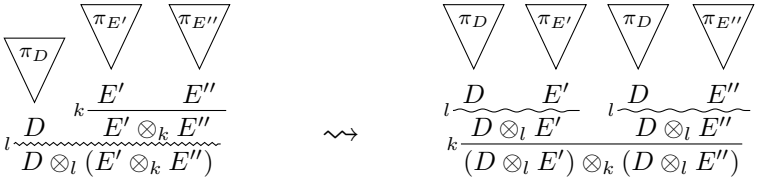


Figure 3: Reducing the purification measure of a refutation through the distributive laws from Theorem 1. Only the cases for inferences above the right-hand premise of purifiers upon l are shown; symmetrical rules are analogous. Wavy inferences in the right-hand side may or may not be purifiers themselves, depending on the location of C in the proof.

Now, observe that given a set A and two elements $x \neq y$, the identity $A \setminus \{x\} \cup A \setminus \{y\} = X$ holds. The first claim then follows by observing that $k \neq \overline{k}$. □

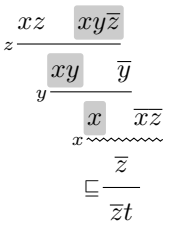
These rules can be applied to hoist purifiers throughout the proof, as shown in Figure 3. Doing so reduces the purification measure of the resulting purifiers, albeit it may duplicate a part of the proof. Since left-purifiers (resp. right-purifiers) are always resolution inferences upon l (resp. $\bar{l}$), Theorem 1 allows moving the purifier "one step" upwards in the proof. The purification measure is then decreased w.r.t. the multiset ordering, because the new purifiers have fewer total ancestors, as shown by Theorem 2. Example 3 shows the iterative application of Theorem 1 on remaining purifiers to attain clause isolation in a refutation.

Theorem 2. *As long as C is not isolated upon l in a generalized resolution-subsumption refutation π of $F \cup \{C\}$, it contains a purifier to which the rules in Figure 3 (or its symmetric versions) can be applied. In that case, applying the corresponding rule yields a correct generalized subsumption-resolution refutation of F. Additionally, if the selected purifier is maximal w.r.t. the subproof ordering (i.e. no inference below the purifier is a purifier itself), then the resulting refutation has a strictly smaller purification measure.*

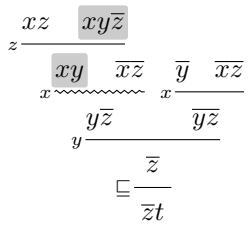
Proof. Let us assume that C is not isolated in π upon l. Proposition 2 then shows the existence of some purifier X with $\mathrm{dp}(X) > 0$. If X is a left-purifier, then it is of the form $D \otimes_l D' = E$, and so either D or D' is not a proof premise; let us assume w.l.o.g. that D is not a proof premise (otherwise a symmetric version of the rules applies), so it must be derived by the application of an inference rule. Rules from Figure 3 then apply depending on whether such inference is a resolution or a subsumption. The same similarly holds in the case of a right-purifier for the rules for a resolution upon $\bar{l}$. Furthermore, in both cases the obtained refutation is guaranteed to be correct by Theorem 1.

We now show that, if X is picked to be a maximal inference, then in each case the purification measure is reduced. Again w.l.o.g. we assume that X is a left-purifier of the form $D \otimes_l D' = E$,

(a) Applying the subsumption rule to the right premise in the left-purifier $x \otimes_x \overline{xz}t = \overline{z}t$:



(b) Applying the resolution rule to the left premise in the left-purifier $x \otimes_x \overline{xz} = \overline{z}$



(c) Applying the resolution rule to the right premise in the left-purifier $xy \otimes_x \overline{xz} = y\overline{z}$

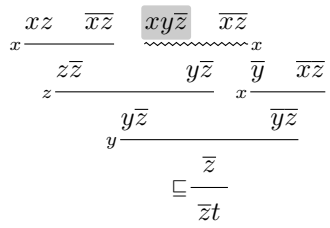


Figure 4: Applying distributivity laws to the left-purifier in the proof from Example 2.

where $D' = E' \otimes_k E''$ if the resolution rule applies, and $E' \sqsubseteq D'$ if the subsumption rule applies; the other cases are symmetric. Let us analyze how each purifier in the input refutation is transformed by the rules from Figure 3.

- Any purifier in the subproofs $\pi_{E'}$ and $\pi_{E''}$ results in a single, identical purifier after applying the rules, so both the amount of purifiers and their purification depths are preserved. The same holds for any purifier in π_D if the subsumption rule is applied.
- If the resolution rule is applied, then any purifier in π_D is duplicated with the same purification depth. Observe that, because the set of ancestors of such a purifier is a strict subset of the ancestors of D', its purification depth is strictly smaller than the purification depth of X.
- Because of the maximality condition, no purifier in the refutation is a descendant of X. Furthermore, any purifier which is not an ancestor nor a descendant of X is preserved in the result with the same purification depth.
- The purifier X itself is replaced in the proof by at most two purifiers of strictly smaller purification depth.

From the discussion above, it follows that the purification measure of the result can be obtained by replacing an element from the purification measure of the input with a number of strictly smaller depths; hence, the rules reduce the purification measure of the refutation w.r.t. the multiset ordering. □

Example 3. Let us showcase the application of distributivity for clause isolation on the proof from Example 2. We apply the rules from Figure 3 step by step for the subproof deriving the left-purifier $x \otimes_x \overline{xz}t = \overline{z}t$; the result is shown in Figure 4.

In this section we have presented a method that, by itself, already results in clause isolation. As explained in Section 3, this enables a procedure for RAT elimination. Nevertheless, this method is quite inefficient. In Sections 5 and 6, we analyze the reasons for such inefficiency, and we present a method that exploits the fact that in practice clause isolation operates over RUP proofs as opposed to subsumption-resolution proofs. The method in Section 5 aims at avoiding generation of useless inferences, whereas Section 6 tackles the problem of reusing inferences in a DAG-like proof.

5 Refining distributivity for RUPs

Since RUP proofs are in essence a sophisticated way to write subsumption-resolution proofs, the clause isolation method from Section 4 already solves the problem of RAT elimination in DRAT proofs. However, the complexity properties of that method are problematic: the proof doubles its size for *every* resolution inference above a purifier. As discussed in Section 1, part of this blow-up may be unavoidable. In the following sections, we find ways to alleviate this issue. One such way, explained in this section, is to reuse as much from the original proof as possible; it turns out this can be obtained as a side effect of generating proper refutations, i.e. without tautologies. For presentation purposes, we will present several refinements to the idea of attaining clause isolation through distributivity; at the end of this section, we condense all these improvements in a relatively simple method. We can also exploit clause sharing, which is an inherent feature of RUP proofs lost to the transformation into subsumption-resolution proofs; this is tackled in Section 6.

Since RUP inferences are conflations of a subsumption and several merges, we can simply apply the distributive laws from Theorem 1. The resulting proof is not in general a subsumption-merge chain itself, because distributive laws do not guarantee that the resolvents are proper. For example, $xy \otimes_x (\overline{x}y \otimes_y \overline{xy})$ can be distributed as $(xy \otimes_x \overline{x}y) \otimes_y (xy \otimes_y \overline{xy})$, but the intermediate resolvent $xy \otimes_y \overline{xy}$ is not a proper clause. Furthermore, all premises of the subsumption-merge chain are modified, and will spawn further applications of distributivity to find derivations of the resolvents for *every* premise; the computational cost then becomes overwhelming. Both these problems can be tackled by refining our distributive laws for RUP inferences. Let us first illustrate our goal with an example, and then present our optimizations for RUP distributivity.

Example 4. Consider the subsumption-merge chain deriving a RUP clause xw from clauses xy, $\overline{y}z$, $\overline{z}uv$, $x\bar{t}v$, $\overline{u}v$ and $\overline{v}w$ shown in Figure 5a, and the clause $D = \overline{x}y\bar{t}u$. We then have $D \parallel xw = \{\overline{x}\}$, so we can obtain a generalized subsumption-resolution derivation of the clause $D \otimes_{\overline{x}} xw = y\bar{t}uw$ by applying Theorem 1. This yields the generalized subsumption-merge chain shown in Figure 5b.

If this is done as a part of a clause isolation procedure, up to six recursive applications of distributivity are triggered, one for each chain premise E whose resolvent $D \otimes_{\overline{x}} E$ is a purifier in the resulting proof. Furthermore, this derivation is not a proper subsumption-merge chain, since it contains tautologies. However, in this case we can obtain a much shorter and convenient derivation for the same clause, shown in Figure 5e. This is a *proper* subsumption-merge chain, and only produces one resolvent of the form $D \otimes_{\overline{x}} E$ instead of six; the clause $\overline{v}w$ is unproblematic, since it was already a premise of the original subsumption-merge chain, and so either it is a refutation premise or we already have a derivation for it.

Example 4 shows that subsumption-merge chains for purifiers can be obtained in some cases, and that they can be smaller than the derivations produced by Theorem 1. We now describe how to obtain such subsumption-merge chains systematically. Figure 6 describes the process in abstract; a running example shows how the subsumption-merge chain from Example 4 is obtained with our method.

Dropping redundant merges The input to our process is a subsumption-merge chain $A_n \colon E_0 \otimes_{k_1} \ldots \otimes_{k_n} E_n$ as in Figure 1c, together with a clause D such that the resolvent $D \otimes_l A_n$ is proper. We start the process by applying Theorem 1, which yields the generalized subsumption-resolution chain $D \otimes_l A_n \colon (D \otimes_l E_0) \otimes_{k_1} \ldots \otimes_{k_n} (D \otimes_l E_n)$ shown in Figure 6a. Let us first show that this is in fact a generalized subsumption-merge chain.

(a) A (proper) subsumption-merge chain

$$\sqsubseteq \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{xy \qquad xyztuvw \quad \overline{y}z}{y}}{xztuvw \quad \overline{z}uv}}{z}}{xtuvw \quad x\bar{t}v}}{t}}{xuvw \quad \overline{u}v}}{u} \dfrac{xvw \quad \overline{v}w}{v}}{xw}$$

(b) A generalized subsumption-merge chain obtained by distributivity with $D = \overline{x}y\bar{t}u$

$$\sqsubseteq \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{y\bar{t}u \;= D \otimes_{\overline{x}} xy}{y}}{yzt\bar{t}uvw \quad y\overline{y}z\bar{t}u \;= D \otimes_{\overline{x}} \overline{y}z}}{z}}{yzt\bar{t}uvw \quad y\overline{z}\bar{t}uv \;= D \otimes_{\overline{x}} \overline{z}uv}}{t}}{yt\bar{t}uvw \quad y\bar{t}uv \;= D \otimes_{\overline{x}} x\bar{t}v}}{u} \dfrac{y\bar{t}uvw \quad y\bar{t}u\overline{u}v \;= D \otimes_{\overline{x}} \overline{u}v}{v}}{y\bar{t}uvw \quad y\bar{t}u\overline{v}w \;= D \otimes_{\overline{x}} \overline{v}w}}{y\bar{t}uw}$$

(c) The generalized subsumption-merge chain after dropping redundant merges

$$\sqsubseteq \dfrac{\dfrac{\dfrac{\dfrac{y\bar{t}u \;= D \otimes_{\overline{x}} xy}{z}}{yzt\bar{t}uvw \quad y\overline{z}\bar{t}uv \;= D \otimes_{\overline{x}} \overline{z}uv}}{t}}{yt\bar{t}uvw \quad y\bar{t}uv \;= D \otimes_{\overline{x}} x\bar{t}v}}{v} \dfrac{y\bar{t}uvw \quad y\bar{t}u\overline{v}w \;= D \otimes_{\overline{x}} \overline{v}w}{y\bar{t}uw}$$

(d) The generalized subsumption-merge chain after simplifying unnecessary resolvents

$$\sqsubseteq \dfrac{\dfrac{\dfrac{\dfrac{y\bar{t}u \;= D \otimes_{\overline{x}} xy}{z}}{yzt\bar{t}uvw \quad \overline{z}uv}}{t}}{yt\bar{t}uvw \quad y\bar{t}uv \;= D \otimes_{\overline{x}} x\bar{t}v}}{v} \dfrac{y\bar{t}uvw \quad \overline{v}w}{y\bar{t}uw}$$

(e) The proper subsumption-merge chain after cutting tautologies off

$$\sqsubseteq \dfrac{\dfrac{y\bar{t}uv \;= D \otimes_{\overline{x}} x\bar{t}v}{v}}{y\bar{t}uvw \quad \overline{v}w}}{y\bar{t}uw}$$

Figure 5: Applying distributivity laws to the left-purifier in the proof from Example 2.

Proposition 3. *Let A, E and D be proper clauses such that $l \notin A$ and the resolution $A \otimes_k E$ is proper and a merge. Then, the resolvent $(D \otimes_l A) \otimes_k (D \otimes_l E)$ is a merge. Furthermore, whenever $k \in D \cup \{\overline{l}\}$, that resolvent equals $D \otimes_l A$.*

Proof. Let us first show the resolvent is a merge. For this we need to show that $(D \otimes_l E) \setminus \{\overline{k}\} \subseteq D \otimes_l A$, or equivalently that $D \setminus \{l, \overline{k}\} \cup E \setminus \{\overline{l}, \overline{k}\} \subseteq D \setminus \{l\} \cup A \setminus \{\overline{l}\}$. Now, the first term of this union is a subset of $D \setminus \{l\}$; and the second term can be rewritten as $E \setminus \{\overline{k}\} \setminus \{\overline{l}\}$, which is then a subset of $A \setminus \{\overline{l}\}$ because the resolvent $A \otimes_k E$ is itself a merge. This shows the inclusion above.

We now show that, assumming $k \in D \cup \{\overline{l}\}$, the equality $(D \otimes_l A) \otimes_k (D \otimes_l E) = D \otimes_l A$ holds. Because $A \otimes_l E$ is itself a merge, we have $E \setminus \{\overline{k}\} \subseteq A$, so we conclude $E \setminus \{k, \overline{k}, \overline{l}\} \subseteq A \setminus \{k, \overline{l}\}$. Furthermore, because $A \parallel E = \{k\}$ and E is a proper clause, we obtain $k \notin E$. Hence, $E \setminus \{\overline{k}, \overline{l}\} \subseteq A \setminus \{\overline{l}, k\}$, and this implies $(D \otimes_l A) \otimes_k (D \otimes_l E) = D \setminus \{l\} \cup A \setminus \{\overline{l}, k\}$. Now, since $l \notin A$ and $k \in A$, we know that $k \neq l$. This means that either $k \in D \setminus \{l\}$, or $k = \overline{l}$. In both

(a) Generalized subsumption-resolution chain obtained by distributivity

$$\sqsubseteq \dfrac{D \otimes_l E_0}{\dfrac{D \otimes_l A_0}{k_1}\quad \dfrac{D \otimes_l E_1}{\dfrac{D \otimes_l A_1}{k_2}\quad \dfrac{D \otimes_l E_2}{\ddots\quad \dfrac{D \otimes_l A_{n-1}}{k_n}\quad D \otimes_l E_n}{D \otimes_l A_n}}}$$

(b) Generalized subsumption-merge chain after dropping merges

$$\sqsubseteq \dfrac{D \otimes_l E_{\varphi(0)}}{\dfrac{D \otimes_l A_{\varphi(0)}}{k_{\varphi(1)}}\quad \dfrac{D \otimes_l E_{\varphi(1)}}{\dfrac{D \otimes_l A_{\varphi(1)}}{k_{\varphi(2)}}\quad \dfrac{D \otimes_l E_{\varphi(2)}}{\ddots\quad \dfrac{D \otimes_l A_{\varphi(m-1)}}{k_{\varphi(m)}}\quad D \otimes_l E_{\varphi(m)}}{D \otimes_l A_{\varphi(m)}}}}$$

(c) Generalized subsumption-merge chain obtained by simplifying unnecessary resolvents

$$\sqsubseteq \dfrac{E^\star_{\varphi(0)}}{\dfrac{D \otimes_l A_{\varphi(0)}}{k_{\varphi(1)}}\quad \dfrac{E^\star_{\varphi(1)}}{\dfrac{D \otimes_l A_{\varphi(1)}}{k_{\varphi(2)}}\quad \dfrac{E^\star_{\varphi(2)}}{\ddots\quad \dfrac{D \otimes_l A_{\varphi(m-1)}}{k_{\varphi(m)}}\quad E^\star_{\varphi(m)}}{D \otimes_l A_{\varphi(m)}}}}$$

(d) Subsumption-merge chain obtained by cutting tautologies off

$$\sqsubseteq \dfrac{E^\star_{\varphi(s)}}{\dfrac{D \otimes_l A_{\varphi(s)}}{k_{\varphi(s+1)}}\quad \dfrac{E^\star_{\varphi(s+1)}}{\dfrac{D \otimes_l A_{\varphi(s+1)}}{k_{\varphi(s+2)}}\quad \dfrac{E^\star_{\varphi(s+2)}}{\ddots\quad \dfrac{D \otimes_l A_{\varphi(m-1)}}{k_{\varphi(m)}}\quad E^\star_{\varphi(m)}}{D \otimes_l A_{\varphi(m)}}}}$$

Figure 6: Refining generalized subsumption-merge chains obtained by distributivity into proper subsumption-merge chains.

cases, the equality $D \setminus \{l\} \cup A \setminus \{\bar{l}, k\} = D \setminus \{l\} \cup A \setminus \{\bar{l}\} = D \otimes_l A$ is straightforward. □

Proposition 3 can be applied to every resolvent in the derivation from Figure 6a to show it is a generalized subsumption-merge chain. Even more interesting is the second claim, which essentially states that some merges can be skipped altogether: whenever $k_i \in D \cup \{\bar{l}\}$, we have $D \otimes_l A_{i-1} = D \otimes_l A_i$, so the resolvent upon k_i can be omitted. Formally, we can enumerate the indices $\{0\} \cup \{0 < i \leq n \mid k_i \notin D \cup \{\bar{l}\}\}$ through a strictly increasing mapping $\varphi\colon \{0, \dots, m\} \longrightarrow \{0, \dots, n\}$ for some $m \in \mathbb{N}$.

Proposition 4. *The derivations from Figures 6a and 6b are generalized subsumption-merge chains deriving* $D \otimes_l A_n$.

Proof. We first show this for the derivation ρ from Figure 6a. From Theorem 1 we know that ρ is a generalized subsumption-resolution chain. We show by induction that for all $0 < i \leq n$ the resolution $(D \otimes_l A_{i-1}) \otimes_{k_i} (D \otimes_l E_i) = A_i$ is a merge, which then implies that ρ is a generalized subsumption-merge chain. We know that $A_{i-1} \otimes_{k_i} E_i = A_i$ is a proper merge; and since the clause A_n and the resolvent $D \otimes_l A_n$ are proper, we also know that $l \notin A_n$, which by Lemma 1 implies $k \notin A_{i-1}$. Therefore, Proposition 3 concludes that $(D \otimes_l A_{i-1}) \otimes_{k_i} (D \otimes_l E_i) = A_i$ is a merge.

Let us now show that the derivation ρ' from Figure 6b is a generalized subsumption-resolution chain. We show by induction that the subchain ρ'_i is a generalized subsumption-resolution chain for all $0 \leq i \leq m$. The definition of φ forces $\varphi(0) = 0$, so the claim trivially holds for $i = 0$. Now, let us assume the claim holds for some $0 \leq i < m$. Then, we simply need to show that

$$(D \otimes_l A_{\varphi(i)}) \otimes_{k_{\varphi(i+1)}} (D \otimes_l E_{\varphi(i+1)}) = D \otimes_l A_{\varphi(i+1)} \tag{1}$$

and that this is a merge.

The definition of φ constrains its values in such a way that $\varphi(i) < \varphi(i+1)$, and additionally for all $\varphi(i) < j < \varphi(i+1)$ we have $k_j \in D \cup \{\bar{l}\}$. Furthermore, since $D \parallel A_n = \{l\}$, and A_n is proper, we know that $l \notin A_n$, and by Lemma 1 this implies $l \notin A_j$. Proposition 3 then guarantees that

$$D \otimes_l A_j = (D \otimes_l A_{j-1}) \otimes_{k_j} (D \otimes_l E_j) = D \otimes_l A_{j-1} \tag{2}$$

Then we obtain

$$D \otimes_l A_{\varphi(i)} = D \otimes_l A_{\varphi(i)+1} = \cdots = D \otimes_l A_{\varphi(i+1)-1}$$

And so we conclude

$$(D \otimes_l A_{\varphi(i)}) \otimes_{k_{\varphi(i+1)}} (D \otimes_l E_{\varphi(i+1)}) = (D \otimes_l A_{\varphi(i+1)-1}) \otimes_{k_{\varphi(i+1)}} (D \otimes_l E_{\varphi(i+1)}) = D \otimes_l A_{\varphi(i+1)}$$

We have shown the equality (1); that it is a merge follows straightforward from the equality above, considering that ρ is a generalized subsumption-merge chain. Finally, we need to show that it derives $D \otimes_l A_n$; but this follows straightforward from (2). □

Example 5. The generalized subsumption-resolution chain from Figure 5b is a generalized subsumption-resolution chain. As predicted by Proposition 4, this is a generalized subsumption-merge chain. However, the merge resolutions $yzt\bar{t}uvw \otimes_y y\bar{y}z\bar{t}u = yzt\bar{t}uvw$ and $y\bar{t}uvw \otimes_u y\bar{t}u\bar{u}v = y\bar{t}uvw$ are redundant. This matches Proposition 3, since $y, u \in x\bar{x}y\bar{t}u$. By dropping those merges, we obtain the generalized subsumption-merge from Figure 5c.

Simplifying unnecessary resolvents The next step is to simplify some of the premises in the generalized subsumption-merge chain from Figure 6b. All premises there are of the form $D \otimes_l E_{\varphi(i)}$. However, it may happen that $D \parallel E_{\varphi(i)} \neq \{l\}$. It turns out some of these premises can be outright replaced by the premise $E_{\varphi(i)}$ itself, for which we already have a derivation. This is based on the following result:

Proposition 5. *Consider clauses A and E, and let us assume that $\bar{l} \notin E$ and $k \notin D$. Then, $(D \otimes_l A) \otimes_k (D \otimes_l E) = (D \otimes_l A) \otimes_k E$.*

Proof. Observe that the side conditions imply $D \setminus \{l\} = D \setminus \{l, k\}$ and $E \setminus \{\bar{l}, \bar{k}\} = E \setminus \{\bar{k}\}$. Hence, the equality is straightforward from:

$$(D \otimes_l A) \otimes_k (D \otimes_l E) = D \setminus \{l\} \cup A \setminus \{\bar{l}, k\} \cup E \setminus \{\bar{l}, \bar{k}\}$$
$$(D \otimes_l A) \otimes_k E = D \setminus \{l, k\} \cup A \setminus \{\bar{l}, k\} \cup E \setminus \{\bar{k}\}$$

□

Let us apply this result to our case. The condition $k_{\varphi(i)} \notin D$ is satisfied for every i, as per the previous stage in the transformation. Then, the result claims that, whenever $\bar{l} \notin E_{\varphi(i)}$, we can replace the chain premise $D \otimes_l E_{\varphi(i)}$ by $E_{\varphi(i)}$. Formally, we define $E^\star_{\varphi(i)}$ as $E_{\varphi(i)}$ if $\bar{l} \notin E_{\varphi(i)}$, and $D \otimes_l E_{\varphi(i)}$ otherwise. The chain from Figure 6c is then a generalized subsumption-merge chain.

Proposition 6. *The derivation from Figure 6c is a generalized subsumption-merge chain deriving $D \otimes_l A_n$.*

Proof. That this derivation derives $D \otimes_l A_n$ is straightforward from Proposition 4. We now need to show that the identities $(D \otimes_l A_{\varphi(i-1)}) \otimes_{k_{\varphi(i)}} E^\star_{\varphi(i)} = D \otimes_l A_{\varphi(i)}$ hold for $0 < i \leq m$ and that they are merges. Showing that they hold follows straightforward from Proposition 5 and the definition of $E^\star_{\varphi(i)}$; we now show they are merges. In the case where $\bar{l} \in E_{\varphi(i)}$ we have $E^\star_{\varphi(i)} = D \otimes_l E_{\varphi(i)}$, and then Proposition 4 yields the claim. Otherwise, $E^\star_{\varphi(i)} = E_{\varphi(i)}$. Because $A_{\varphi(i)-1} \otimes_l E_{\varphi(i)} = A_{\varphi(i)}$ is a merge resolution, Lemma 1 yields $E_{\varphi(i)} \setminus \{k_{\varphi(i)}\} \subseteq A_{\varphi(i)-1} \subseteq A_{\varphi(i-1)}$. Now, since $\bar{l}$ does not occur in $E_{\varphi(i)}$, we conclude $E_{\varphi(i)} \subseteq A_{\varphi(i-1)} \setminus \{\bar{l}\} \subseteq D \otimes_l A_{\varphi(i-1)}$, and this shows the missing case. □

Example 6. We now apply the transformation above to the generalized subsumption-merge chain from Figure 5c obtained in Example 5. Observe that $x \in x\bar{t}v$ and $x \notin \bar{z}uv$, $\bar{v}w$. Hence, we can replace the resolvents $y\bar{z}\bar{t}uv = \bar{x}y\bar{t}u \otimes_{\bar{x}} \bar{z}uv$ by $\bar{z}uv$; and $y\bar{t}u\bar{v}w = \bar{x}y\bar{t}u \otimes_{\bar{x}} \bar{v}w$ by $\bar{v}w$. We thus obtain the generalized subsumption-merge chain from Figure 5d.

Cutting tautologies off The derivation obtained in the previous stage is still not a proper subsumption-merge chain, due to the possible presence of tautologies. However, the proof has now a restricted enough form to solve this in a simple way. Let s be the largest $0 < s \leq m$ such that $\overline{k_{\varphi(s)}} \in D$, or just 0 if there is no such s. Then, $A_{\varphi(s)}$ is the first $A_{\varphi(i)}$ that is proper, and Lemma 1 guarantees that so is any subsequent $A_{\varphi(i)}$. By removing the inferences above $D \otimes_l A_{\varphi(s)}$ from the chain from Figure 6c, we obtain the chain from Figure 6d, which is this time a *proper* subsumption-merge chain, thus concluding the procedure.

Lemma 3. *For all $\varphi(s) \leq i \leq n$ we have $D \parallel A_i = \{l\}$. In particular, the resolvent $D \otimes_l A_{\varphi(i)}$ is a proper clause for all $s \leq i \leq m$.*

Proof. We proceed by reverse induction. We have been assumming from the beginning that $D \parallel A_n = \{l\}$, so the base case holds. We now show the induction claim: for all $\varphi(s) < i \leq n$ such that $D \parallel A_i = \{l\}$ holds, so does $D \parallel A_{i-1} = \{l\}$. Let us show first that $\overline{k_i} \notin D$.

- If $i \in \{\varphi(s+1), \ldots, \varphi(m)\}$, then the definition of s yields $\overline{k_i} \notin D$.
- Otherwise, from the definition of φ we know $k_i \in D \cup \{\bar{l}\}$. Observe that, because the original derivation from Figure 1c is a proper subsumption-merge chain, we have $k_i \notin A_i$ by Lemma 2. On the other hand, $\bar{l} \in A_n \subseteq A_i$ by Lemma 1, and since A_i is a proper clause, we obtain $k_i \neq \bar{l}$. Thus, $k_i \in D$ holds and since D is proper too, $\overline{k_i} \notin D$.

Then, because $A_{i-1} = A_i \cup \{k_i\}$, we conclude $D \parallel A_{i-1} = D \parallel A_i = \{l\}$, which shows the induction claim, and so the result holds. □

Lemma 4. *For all $s < i \leq m$, we have $k_{\varphi(i)} \in D \otimes_l A_{\varphi(i-1)}$ and $D \otimes_l A_{\varphi(i)} = D \otimes_l A_{\varphi(i-1)} \setminus \{k_{\varphi(i)}\}$.*

Proof. From Lemma 1 we know that $D \otimes_l A_{\varphi(i)} \subseteq D \otimes_l A_{\varphi(i-1)}$ and $D \otimes_l A_{\varphi(i-1)} \setminus D \otimes_l A_{\varphi(i)} \subseteq \{k_{\varphi(i)}\}$. We thus just need to show $k_{\varphi(i)} \in D \otimes_l A_{\varphi(i-1)} \setminus D \otimes_l A_{\varphi(i)}$. By the definition of φ we know that $k_{\varphi(i)} \notin D \cup \{l\}$. Furthermore, the original derivation from Figure 1c is a subsumption-merge chain, so Lemma 2 says that $k_{\varphi(i)} \in A_{\varphi(i-1)}$ and $k_{\varphi(i)} \notin A_{\varphi(i)}$ hold. Hence we conclude $k_{\varphi(i)} \in D \otimes_l A_{\varphi(i-1)}$ and $k_{\varphi(i)} \notin D \otimes_l A_{\varphi(i)}$. □

Lemma 5. *For all $s \leq i \leq m$, the clause $E^\star_{\varphi(i)}$ is proper.*

Proof. If $\bar{l} \notin E_{\varphi(i)}$, then we have $E^\star_{\varphi(i)} = E_{\varphi(i)}$, which is in any case proper. We now show the case when $\bar{l} \in E_{\varphi(i)}$, in which case $E^\star_{\varphi(i)} = D \otimes_l E_{\varphi(i)}$. Observe that both D and $E_{\varphi(i)}$ are proper clauses, so it suffices to show that $D \parallel E_{\varphi(i)} = \{l\}$. The "$\supseteq$" inclusion is straightforward; we now show the "$\subseteq$" inclusion. We discuss two different cases.

- If $i = 0$, then $E_{\varphi(i)} \subseteq A_{\varphi(i)}$, and Lemma 3 then yields $D \parallel E_{\varphi(i)} \subseteq D \parallel A_{\varphi(i)} = \{l\}$.
- If $i > 0$, then observe that $k_{\varphi(i)} \notin D$ by the definition of φ. Furthermore, the inclusion $E_{\varphi(i)} \setminus \{\overline{k_{\varphi(i)}}\} \subseteq A_{\varphi(i-1)}$ holds because the resolution $A_{\varphi(i-1)} \otimes_{k_{\varphi(i)}} E_{\varphi(i)}$ is a merge. Then, we conclude

$$D \parallel E_{\varphi(i)} = D \parallel (E_{\varphi(i)} \setminus \{\overline{k_{\varphi(i)}}\}) \subseteq D \parallel A_{\varphi(i-1)} = \{l\}$$

  as we wanted, where we have used Lemma 3. □

Lemma 6. *The clause $E^\star_{\varphi(s)}$ is subsumed by $D \otimes_l A_{\varphi(s)}$. Furthermore, for all $s < i \leq m$, we have $D \otimes_l A_{\varphi(i-1)} \parallel E_{\varphi(i)} = \{k_{\varphi(i)}\}$.*

Proof. Let us first show the subsumption. The case $s = 0$ follows straightforward because the derivation from Figure 6c is a generalized subsumption-merge chain. We now show the case $s > 0$, where the definition of s forces $\overline{k_{\varphi(s)}} \in D$. On the other hand, the definition of φ yields $k_{\varphi(s)} \notin D \cup \{\bar{l}\}$. From these two facts we can observe $\overline{k_{\varphi(s)}} \in D \setminus \{l\} \subseteq D \otimes_l A_{\varphi(s)}$. Furthermore, because the resolution $(D \otimes_l A_{\varphi(s)}) \otimes_{k_{\varphi(s)}} E^\star_{\varphi(s-1)}$ is a merge, we obtain

$$E^\star_{\varphi(s)} \setminus \{\overline{k_{\varphi(s)}}\} \subseteq (D \otimes_l A_{\varphi(s-1)}) \setminus \{\overline{k_{\varphi(s)}}\} = D \otimes_l A_{\varphi(s)}$$

using Lemma 4, which concludes the proof of the subsumption.

Now we show the second claim. Because the resolvent $(D \otimes_l A_{\varphi(i-1)}) \otimes_{k_{\varphi(i)}} E^\star_{\varphi(i)}$ is a merge, the inclusion $E^\star_{\varphi(i)} \setminus \{\overline{k_{\varphi(i)}}\} \subseteq D \otimes_l A_{\varphi(i-1)}$ holds. Now observe that $\overline{k_{\varphi(i)}} \in E^\star_{\varphi(i)}$, and furthermore Lemma 5 says that $E^\star_{\varphi(i)}$ is proper. Then we obtain the inclusion

$$E^\star_{\varphi(i)} \setminus \{\overline{k_{\varphi(i)}}\} \subseteq (D \otimes_l A_{\varphi(i-1)}) \setminus \{k_{\varphi(i)}\}$$

This implies that whenever $k \in D \otimes_l A_{\varphi(i-1)}$ and $k \in E^\star_{\varphi(i)}$ for some literal k, we have $k = k_{\varphi(i)}$; hence we have shown $D \otimes_l A_{\varphi(i-1)} \,\|\, E^\star_{\varphi(i)} \subseteq \{k_{\varphi(i)}\}$. To show the "$\supseteq$" inclusion, simply observe that the definition of φ implies $l \neq k_{\varphi(i)} \neq \bar{l}$; this implies:

$$k_{\varphi(i)} \in A_{\varphi(i-1)} \setminus \{l\} \subseteq D \otimes_l A_{\varphi(i-1)}$$
$$\overline{k_{\varphi(i)}} \in E_{\varphi(i)} \setminus \{\bar{l}\} \subseteq E^\star_{\varphi(i)}$$

□

Proposition 7. *The derivation from Figure 6d is a subsumption-merge chain deriving* $D \otimes_l A_n$.

Proof. To show the claim, we proceed in several stages. The first stage is to show that our derivation is a generalized subsumption-merge chain. Proposition 6 makes this a mere matter of proving that $E^\star_{\varphi(s)} \sqsubseteq D \otimes_l A_{\varphi(s)}$; this is in turn shown by Lemma 6. The second stage is to show that this chain is in fact proper. We need to show that the clauses $E^\star_{\varphi(i)}$ and $D \otimes_l A_{\varphi(i)}$ are proper for $s \leq i \leq m$, which is given by Lemmas 5 and 3; and that the resolutions $(D \otimes_l A_{\varphi(i-1)}) \otimes_l (E^\star_{\varphi(i)})$ are proper for $s < i \leq m$, which is shown in Lemma 6. Finally, we need to show that this chain derives $D \otimes_l A_n$, but this is straightforward from Proposition 6. □

Example 7. In the generalized subsumption-merge chain from Figure 5d, the merge pivots are z, t and v. We can remove every inference above the merge upon t because the clause $D = \bar{x}y\bar{t}u$ contains $\bar{t}$. The obtained derivation, shown in Figure 5e is again a generalized subsumption-merge chain, and this time a proper one since we got rid of all tautologies.

Putting all the pieces together Observe that all the conditions above are expressed exclusively in terms of the clause D, the premises E_i and the merge pivots k_i. In particular, it is not necessary to compute the intermediate clauses A_i to generate the subsumption-merge chain that derives $D \otimes_l A_n$. All in all, we can write the process above as a relatively simple recursive function, which we call the *pseudo-distributive operator*. If ρ is the subsumption-merge chain from Figure 1c, we call $\mathcal{D}(D, l, \rho)$ the subsumption-merge deriving $D \otimes_l A_n$ from Figure 6d. By carefully checking the conditions above, we can define $\mathcal{D}(D, l, \rho)$ recursively on ρ as follows:

$$\mathcal{D}(D, l, A_n \colon E_0) = A_n \colon E^\star_0$$
$$\mathcal{D}(D, l, A_n \colon \rho \otimes_{k_n} E_n) = \begin{cases} \mathcal{D}(D, l, A_n \colon \rho) & \text{if } k_n \in D \cup \{\bar{l}\} \\ E^\star_n & \text{if } \overline{k_n} \in D \setminus \{l\} \\ \mathcal{D}(D, l, A_n \colon \rho) \otimes_{k_n} E^\star_n & \text{if } k_n, \overline{k_n} \notin D \end{cases}$$

where as before $E^\star_i$ is E_i if $\bar{l} \notin E_i$; and $D \otimes_l E_i$ if $\bar{l} \in E_i$; this encodes the paragraph *Simplifying unnecesary resolvents*. The third case corresponds to the default; the first case skips some resolvents as per the *Dropping redundant merges* paragraph, and the second case represents the early cutoff in *Cutting tautologies off*.

The development of the pseudo-distributive operator reduces the appearance of undesired redundant inferences, hence reducing the size of the resulting proof. Pseudo-distributivity

can be directly applied to subsumption-resolution proofs; however, part of its utility lies on its reuse of some already present premises. Explaining how to efficiently use the pseudo-distributive operator over DAG-like proofs (as RUP proofs are) as opposed to tree-like proofs is still missing, though; since enhanced clause sharing is a feature of pseudo-distributivity, it makes much sense to get advantage of this. The next section is devoted to this optimization in the context of RUP proofs.

6 Clause isolation in RUP proofs

Now that we are equipped with an analogue of Theorem 1 for subsumption-merge chains, we use it to generate isolated RUP refutations without first transforming the input proof into a subsumption-resolution refutation. So far we have relied on the notion of purifiers, which does not readily transfer to RUP proofs. For the time being, we will regard RUP inferences in a RUP refutation as merely fragments of a subsumption-resolution refutation; later on we show that we can forego this view and work directly on RUP inferences. For a subsumption-merge chain ρ as in Figure 1c, we denote by $\rho^{(i)}$ its subchain $A_i \colon E_0 \otimes_{k_1} \ldots \otimes_{k_i} E_i$. We can use the pseudo-distributive operator from Section 5 to achieve clause isolation.

Example 8. We assume that we are trying to achieve clause isolation for some clause C upon the literal x. For the moment we are regarding subsumption-merge chains as fragments of a global resolution proof; assume we know which chain premises are impure. In the chain in the left, the premise xu is impure, and it is immediately involved in a right-purifier.

$$\sqsubseteq\dfrac{\overline{x}yz}{t\dfrac{\overline{x}yztu \qquad \overline{x}\bar{t}u}{z\dfrac{\overline{x}yzu \qquad y\overline{z}}{\overline{x}\dfrac{\overline{x}yu \qquad xu}{u\dfrac{yu \qquad y\overline{u}}{y}}}}}} \quad\rightsquigarrow\quad \sqsubseteq\dfrac{yzu \quad = xu \otimes_x \overline{x}yz}{t\dfrac{yztu \qquad \bar{t}u \quad = xu \otimes_x \overline{x}\bar{t}u}{z\dfrac{yzu \qquad y\overline{z}}{u\dfrac{yu \qquad y\overline{u}}{y}}}}$$

Here we could just apply the rules from Figure 3 to the purifier $\overline{x}yu \otimes_{\overline{x}} xu = yu$, but then the issues exposed at the beginning of Section 5 would arise. Instead, we use pseudo-distributivity: the subsumption-resolution chain $\mathcal{D}\big(xu,\, x,\, \rho^{(2)}\big)$ derives the intermediate clause yu, so we replace $\rho^{(3)}$ with $\mathcal{D}\big(xu,\, x,\, \rho^{(2)}\big)$. The clause xu is still impure, so we could then move on to compute the subsumption-merge chains $\mathcal{D}(xu,\, x,\, \rho')$, where ρ' is the chains deriving $\overline{x}yz$ and $\overline{x}\bar{t}u$. This would in turn require computing new subsumption-merge chains through the pseudo-distributive operator.

More complex cases can occur, e.g. some intermediate clauses A_i could be impure. We classify the chains as follows:

Theorem 3. *Let us consider a subsumption-merge chain ρ as in Figure 1c. Then, exactly one of the following holds:*

1. *ρ is a pure chain: for all $0 \leq i \leq n$ the clause E_i is pure.*
2. *ρ is a left-semipure chain: there exist $0 \leq i < j \leq n$ such that E_i is impure and $k_j = l$.*

3. *ρ is a right-semipure chain: there exists an $0 < i \leq n$ such that E_i is impure and $k_i = \bar{l}$.*

4. *ρ is an impure chain: for some $0 \leq i \leq n$ the clause E_i is impure, and furthermore $k_j \notin \{l, \bar{l}\}$ for each $0 < j \leq n$.*

Furthermore, the derived clause A_n is impure if and only if ρ is impure.

Proof. We first show that at least one of the alternatives holds. Let us assume that ρ is neither pure, nor left-semipure, nor right-semipure; we show it is then impure. Since ρ is not pure, there must be an $0 \leq i \leq n$ such that E_i is impure. We distinguish two cases:

- If $i = 0$, then since E_i is impure we know that $l \in E_0 \subseteq A_0$. Since A_0 is a proper clause, we conclude $\bar{l} \notin A_0$, so Lemma 2 implies that $k_j \neq \bar{l}$ for all $0 < j \leq n$. Furthermore, because ρ is not left-semipure, we conclude that $k_j \neq l$ for all $0 < j \leq n$. This shows that ρ is impure.

- If $i > 0$, then observe that $A_{i-1} \parallel E_i = \{k_i\}$ because ρ is a subsumption-merge chain, and $l \in E_i$ because E_i is impure. Now, ρ is not right-semipure, so $k_i \neq \bar{l}$, which means that $l \in A_i = A_{i-1} \otimes_{k_i} E_i$. Lemma 2 then implies that for all $0 < j \leq i$, the literal k_j is neither l nor $\bar{l}$. Furthermore, since ρ is not left-impure, we know that $k_j \neq l$ for all $i < j \leq n$. Finally, let us assume that $k_j = \bar{l}$ for some $i < j \leq n$; then Lemma 2 shows that $\bar{l} \in A_i$, but this leads to a contradiction because A_i is a proper clause. We have thus shown that ρ is impure in this case too.

Now let us show that no pair of alternatives may simultaneously hold. The only non-trivial case is where ρ is both left-semipure and right-semipure. But in this case there exist $0 \leq i < j \leq n$ such that $k_i = \bar{l}$ and $k_j = l$, which contradicts Lemma 2.

We now show the final claim. We show that A_n is impure if ρ is impure, and that A_n is pure in each other case.

- If ρ is pure, then ρ is a resolution-subsumption derivation where each premise is pure, and so A_n is pure.

- If ρ is left-semipure, then there are $0 \leq i < j \leq n$ such that E_i is impure and $k_j = l$. Now, from Lemma 2 we know that $l = k_j \notin A_n$, so in particular A_n cannot be impure.

- If ρ is right-semipure, then there is an $0 < i \leq n$ such that E_i is impure and $k_i = \bar{l}$. Because ρ is a subsumption-merge chain, the clause A_{i-1} contains $k_i = \bar{l}$, and so it does not contain l. Finally, $A_n \subseteq A_{i-1}$, so A_n is pure because $l \notin A_n$.

- If ρ is impure, there is some $0 \leq i \leq n$ such that E_i is impure, and additionally k_j is neither l nor $\bar{l}$ for each $0 < j \leq n$. If $i = 0$, then since $E_0 \sqsubseteq A_0$, we conclude that A_0 is impure; if $i > 0$, then since $k_i \notin \{l, \bar{l}\}$, we deduce that $A_i = A_{i-1} \otimes_{k_i} E_i$ is impure. In both cases we have shown that A_i is impure. Since $k_j \notin \{l, \bar{l}\}$ for each $i < j \leq n$, a simple induction argument shows that the A_j are all impure; in particular A_n is.

□

Observe that Theorem 3 depends exclusively on the premises E_i and the pivots k_i of each subsumption-merge chain, so we do not need to reconstruct the chain to decide whether clauses in a RUP proof are pure or impure.

Example 9. Consider the DRAT proof from Example 1. If we consider the RAT xu as an impure clause, then the chains for the clauses $x\overline{y}$ and x are impure, the ones for $\overline{y}$, u and $\square$ are right-semipure, and the one for $z\overline{u}w$ is pure.

Let us reconsider the method for clause isolation on resolution-subsumption refutations as discussed in Example 8. Intuitively, for each left-purifier $E \otimes_l E'$, where E is impure, what our method does is finding a different derivation of the pure resolvent by reordering the proof. In particular, we find a way to derive $E \otimes_l E'$ without deriving the impure clause E. In order to do so, we may need to derive new clauses, but those can be derived by a purifier as well, i.e. by the resolvent of an impure clause with a pure clause upon l. We exploit this to design an algorithm for clause isolation on RUP refutations.

Algorithm overview Our algorithm proceeds bottom-up in the RUP refutation π of $F \cup \{C\}$. Throughout the process, a *to-do list* T containing pairs $[L, R]$ is kept, where R is a pure clause and $L \parallel R = \{l\}$; observe that the resolvent $L \otimes_l R$ is then always pure. Each pair $[L, R]$ in the to-do list represents a resolvent $L \otimes_l R$ for which a derivation needs to be found. At every stage of the algorithm, we have an *unprocessed RUP derivation* π of a CNF formula G from $F \cup \{C\}$. According to π, some of the clauses in G are pure; we call this subformula G_{pure}. We also have a *generated RUP refutation* σ of the pure clauses in G together with the resolvents $L \otimes_l R$ originating in the to-do list. Initially, both σ and T are empty. In every step, the algorithm removes the last RUP inference from π, and prepends some RUP inferences σ' to σ in such a way that Invariant 1 is preserved.

Invariant 1. *The following hold at every step in the algorithm:*

- *π is a RUP derivation of a CNF formula G from $F \cup \{C\}$.*
- *For every pair $[L, R] \in T$, we have $L \in G$, $R \in G_{pure}$, and $L \parallel R = \{l\}$.*
- *σ is a RUP refutation of $G_{pure} \cup \{L \otimes_l R \mid [L, R] \in T\}$.*

Every time a RUP inference from π is dropped, so is a clause from G; for each pair $[L, R]$ in the to-do list containing that clause, a derivation must be found to maintain Invariant 1. By the end of the algorithm, σ holds a RUP refutation of $(F \cup \{C\})_{\text{pure}} \cup \{L \otimes_l R \mid [L, R] \in T\}$. The first member of this union is just F. Furthermore, the clauses $L \otimes_l R$ are either resolvents of two clauses in F, which are RUPs in F anyway [18]; or $L = C$ and $R \in F$ hold, and then the resolvents $C \otimes_l R$ are allowed as premises of an isolated proof. Hence, C will be isolated upon l in the output proof.

Processing a single RUP We now describe how to construct the prepended RUP inferences σ'. We will use the pseudo-distributive operator defined in Section 5.

Given a clause D, a literal l and a subsumption-merge chain ρ given by $A\colon E_0 \otimes_{k_1} \ldots \otimes_{k_n} E_n$ where $D \parallel A = \{l\}$, the subsumption-merge chain $\mathcal{D}(D, l, \rho)$ is given by Figure 6d. Some of the premises E_i are used in $\mathcal{D}(D, l, \rho)$ in the form $D \otimes_l E_i$; we refer to these premises as the *external premises* of $\mathcal{D}(D, l, \rho)$; similarly, we call the premises E_i which are used in $\mathcal{D}(D, l, \rho)$ as themselves its *internal premises*. Let us show first a few auxiliary results that will be helpful in arguing the correctness of our algorithm.

Lemma 7. *Let D be a clause, k any literal, and ρ a subsumption-merge chain as in Figure 1c such that $D \parallel A_n = \{k\}$. Then, the following hold:*

1. $k_j \notin \{k, \overline{k}\}$ *for each* $0 < j \leq n$.

2. *For each internal premise* E_i *of* $\mathcal{D}(D, k, \rho)$, *we have* $\overline{k} \notin E_i$.

Proof. 1. The condition $D \parallel A_n = \{k\}$ implies that $\overline{k} \in A_n$, and since A_n is a proper clause, we conclude that $k \notin A_n$. Since $\overline{k} \in A_n$, we have $\overline{k} \in A_j$ for all $0 \leq j \leq n$. Because $A_j = A_{j-1} \setminus \{k_j\}$ for each $0 < j \leq n$, we conclude that $k_j \neq \overline{k}$. On the other hand, this also implies that $k \notin A_j$ for each $0 \leq j \leq n$, and since ρ is a subsumption-merge chain we conclude that $k_j \neq k$ for all $0 < j \leq n$.

2. Straightforward from the conditions for $E_i^\star = E_i$ in the definition of $\mathcal{D}(D, k, \rho)$ in Section 5. □

Let ρ be the last chain in the RUP derivation π of G from $F \cup \{C\}$. The subproof π' up to ρ derives the formula G'. Then, ρ is given by $A \colon E_0 \otimes_{k_1} \ldots \otimes_{k_n} E_n$, and furthermore $G = G' \cup \{A\}$ holds. Let us also assume that we have a to-do list T and a RUP refutation σ such that Invariant 1 holds for π, T and σ. We define a new to-do list T' and a new sequence of RUPs σ' such that Invariant 1 holds for π', T' and $\sigma'\sigma$.

The order of the RUPs in σ' is irrelevant, so we merely explain which RUPs must be derived and how to obtain the corresponding subsumption-merge chains. In order to enforce Invariant 1, we need to make sure that the premises of these chains either occur in G'_{pure}, or are the resolvent $L \otimes_l R$ of clauses $L, R \in G'$ such that R is pure; in the latter case we also need to add the pair $[L, R]$ to the to-do list T. In any case T' is obtained from T by removing all pairs containing the derived clause A, and adding the necessary pairs as above.

We may need to obtain two kinds of subsumption-merge chains. The first kind derives the clause A itself, provided it is pure; if A is impure, then it is not a premise of σ, so there is no need to derive it. The second kind derives the resolvent corresponding to each pair $[A, R]$ or $[L, A]$ in the to-do list. These cases are summarized in Algorithm 1.

Deriving the same conclusion If A is pure, we need to provide a subsumption-merge chain deriving A. Theorem 3 restricts this to three cases:

- If ρ is a pure chain, then ρ itself satisfies the aforementioned purity requirements for Invariant 1.

- If ρ is a left-semipure chain, then there is some $0 < i \leq n$ with $k_i = l$. Lemma 2 makes j unique. Since the subchain $\rho^{(i-1)}$ derives A_{i-1} and we have $A_i = A_{i-1} \otimes_l E_i = E_i \otimes_{\overline{l}} A_{i-1}$, then we can derive A through the subsumption chain:

$$A \colon \mathcal{D}\Big(E_i, \overline{k_i}, \rho^{(i-1)}\Big) \otimes_{k_{i+1}} E_{k_{i+1}} \otimes_{k_{i+2}} \ldots \otimes_{k_n} E_n \tag{3}$$

  We now show that the purity requirements above hold. First, every premise E_j with $i < j \leq n$ is pure because Lemma 2 forces $l \notin E_j$. Second, Lemma 7 ensures that for every internal premise E_j in $\mathcal{D}\big(E_i, \bar{l}, \rho_{i-1}\big)$ we have $l \notin E_j$, and so they are pure too. Finally, for every external premise E_j in $\mathcal{D}\big(E_i, \bar{l}, \rho_{i-1}\big)$, its corresponding resolvent is $E_j \otimes_l E_i$, and since $\bar{l} \in E_i$ we know that E_i is pure, so the pair $[E_j, E_i]$ satisfies the purity requirements.

- If ρ is a right-semipure chain, then there is an $0 < i \leq n$ such that E_i is impure and $k_i = \bar{l}$. In this case, we use again the chain (3). This time, since the pairs introduced in

the to-do list are of the form $[E_i, E_j]$ for some $0 \leq j < i$, we need to show that E_j is pure for all $j \neq i$. For $i < j \leq n$, this is granted by Lemma 2. For $0 < j < i$, Lemma 2 shows that $\overline{l} \in A_j$ (so in particular $l \notin A_j$) and $l \neq k_j \neq \overline{l}$. Since ρ is a subsumption-merge chain, this means that $E_j \setminus \{\overline{k}_j\} \subseteq A_j$. We thus conclude $l \notin E_j$, so E_j is pure. Finally, for the case $j = 0$, we know by the same argument as above that $l \notin A_0$, so $l \notin E_0$, hence E_0 is pure too.

Deriving resolvents for pairs in the to-do list For every pair in the to-do list containing A, we need to provide a subsumption-merge chain deriving the corresponding resolvent. Let us distinguish three cases:

- If a pair $[A, R]$ occurs in T, then the subsumption-merge chain $\mathcal{D}\big(R, \overline{l}, \rho\big)$ derives the resolvent $A \otimes_l R$. For every external premise E_i of this pseudo-distribution, the premise $E_i \otimes_l R$ is used; but this is unproblematic because the pair $[E_i, R]$ satisfies the conditions from Invariant 1. Furthermore, we need to make sure that every internal premise E_i is pure; Lemma 7 says that $l \notin E_i$, which precludes that they are impure.
- If a pair $[L, A]$ occurs in T and L is pure, then Invariant 1 forces A to be pure as well. Since A will be derived anyway because it is pure, we can simply prepend the subsumption-merge chain $(L \otimes_l A)\colon L \otimes_l A$.
- If a pair $[L, A]$ occurs in T and L is impure, then as above A must be pure. Now, Lemma 7 together with Theorem 3 forces ρ to be a pure chain, so all E_i are pure and therefore the subsumption-merge chain $\mathcal{D}(L,\, l,\, A)$ derives $L \otimes_l A$ and satisfies the purity requirements.

Example 10. We showcase our clause isolation procedure for RUP proofs by transforming the DRAT proof from Example 1 into a RUP proof. The proof contains only one RAT, namely xu upon x; our goal is to eliminate it by isolating this clause upon x. We proceed backwards in the proof, starting with an empty generated RUP refutation σ, and an empty to-do list T.

1. We process the right-semipure chain $\square\colon \overline{z}\overline{u}\overline{v} \otimes_{\overline{v}} \overline{u}v\overline{w} \otimes_{\overline{w}} \overline{x}yw \otimes_{\overline{z}} \overline{x}y z \otimes_{\overline{x}} x \otimes_y \overline{y} \otimes_{\overline{u}} u$. The to-do list is empty, so as explained above we must derive $\square$ through the chain $\mathcal{D}\big(x,\, x,\, \rho^{(3)}\big) \otimes_y \overline{y} \otimes_{\overline{u}} u$. This is given by

$$\square\colon \overline{z}\overline{u}\overline{v} \otimes_{\overline{v}} \overline{u}v\overline{w} \otimes_{\overline{w}} [x, \overline{x}yw] \otimes_{\overline{z}} [x, \overline{x}yz] \otimes_y \overline{y} \otimes_{\overline{u}} u$$

   where pairs $[L, R]$ denote the resolvent $L \otimes_x R$. By the end of this step the to-do list is $T = \{[x, \overline{x}yw], [x, \overline{x}yz]\}$.

2. Similarly to the previous step, the right-semipure chain $u\colon \overline{z}uv \otimes_v u\overline{v}\overline{w} \otimes_{\overline{w}} \overline{x}yw \otimes_{\overline{z}} \overline{x}yz \otimes_{\overline{x}} x \otimes_y \overline{y}$ must be replaced by the chain $\mathcal{D}\big(x,\, x,\, \rho^{(3)}\big) \otimes_y \overline{y}$ given by:

$$u\colon \overline{z}uv \otimes_v u\overline{v}\overline{w} \otimes_{\overline{w}} [x, \overline{x}yw] \otimes_{\overline{z}} [x, \overline{x}yz] \otimes_y \overline{y}$$

   The to-do list remains the same, since we only add pairs that already exist in T.

3. The right-semipure chain $\overline{y}\colon uvw \otimes_v z\overline{u}\overline{v} \otimes_u z\overline{u}w \otimes_w \overline{x}y\overline{w} \otimes_z \overline{x}y\overline{z} \otimes_{\overline{x}} x\overline{y}$ is replaced by $\mathcal{D}\big(x\overline{y},\, x,\, \rho^{(4)}\big)$:

$$\overline{y}\colon uvw \otimes_v z\overline{u}\overline{v} \otimes_u z\overline{u}w \otimes_w [x\overline{y}, \overline{x}y\overline{w}] \otimes_z [x\overline{y}, \overline{x}y\overline{z}]$$

   The two new pairs are added to the to-do list, which now contains the pairs $[x, \overline{x}yw]$, $[x, \overline{x}yz]$, $[x\overline{y}, \overline{x}y\overline{w}]$, and $[x\overline{y}, \overline{x}y\overline{z}]$.

Algorithm 1: Single RAT elimination

Input: F, a CNF formula
Input: l, a literal
Input: C, a RAT clause in F upon l
Input: θ_D, a subsumption-merge chain deriving $C \otimes_l D$ from F for each $D \in F$
Input: π, a RUP refutation of $F \cup \{C\}$
Let $\sigma = \varepsilon$
Let $T = \emptyset$
while π *is of the form* π', ρ *where* ρ *is* $A\colon E_0 \otimes_{k_1} \ldots \otimes_{k_n} E_{k_n}$ **do**
  if ρ *is a pure chain* **then**
    $\sigma \coloneqq \rho, \sigma$
  else if ρ *is left-semipure* **then**
    Let $0 < i \leq n$ such that $k_i = l$
    Let ρ' be $A\colon \mathcal{D}\big(E_i, \bar{l}, \rho^{(i-1)}\big) \otimes_{k_{i+1}} E_{k_{i+1}} \otimes_{k_{i+2}} \ldots \otimes_{k_n} E_n$
    $\sigma \coloneqq \rho', \sigma$
    $T \coloneqq T \cup \{[E_j, E_i]$ such that E_j is an external premise in $\rho'\}$
  else if ρ *is right-semipure* **then**
    Let $0 < i \leq n$ such that $k_i = \bar{l}$
    Let ρ' be $A\colon \mathcal{D}\big(E_i, l, \rho^{(i-1)}\big) \otimes_{k_{i+1}} E_{k_{i+1}} \otimes_{k_{i+2}} \ldots \otimes_{k_n} E_n$
    $\sigma \coloneqq \rho', \sigma$
    $T \coloneqq T \cup \{[E_i, E_j]$ such that E_j is an external premise in $\rho'\}$
  end
  for $[L, R] \in T$ **do**
    if $L = A$ **then**
      Let ρ' be $A \otimes_l R\colon \mathcal{D}\big(R, \bar{l}, \rho\big)$
      $\sigma \coloneqq \rho', \sigma$
      $T \coloneqq T \setminus \{[A, R]\} \cup \{[E_j, R]$ such that E_j is an external premise in $\rho'\}$
    else if $R = A$ **then**
      Let ρ' be $L \otimes_l A\colon \mathcal{D}(L, l, \rho)$
      $\sigma \coloneqq \rho', \sigma$
      $T \coloneqq T \setminus \{[L, A]\} \cup \{[L, E_j]$ such that E_j is an external premise in $\rho'\}$
    end
  end
  $\pi \coloneqq \pi'$
end
for $[L, R] \in T$ **do**
  if $L = C$ **then**
    $\sigma \coloneqq \theta_R, \sigma$
  else
    Let ρ be $L \otimes_l R\colon L \otimes_l R$
    $\sigma \coloneqq \rho, \sigma$
  end
end
Output: σ, a RUP refutation of F

4. We do not need to derive the clause x because its chain $x\colon z\overline{u}w \otimes_w xy\overline{w} \otimes_{\overline{u}} xu \otimes_z xy\overline{z} \otimes_y x\overline{y}$ is impure. However, x occurs in two pairs from T, so we need to compute the chains for the corresponding resolvents. These are $\mathcal{D}(\overline{x}yw, \overline{x}, \rho)$ and $\mathcal{D}(\overline{x}yz, \overline{x}, \rho)$, which are given by:

$$\begin{aligned} yw\colon z\overline{u}w \otimes_{\overline{u}} [xu, \overline{x}yw] \otimes_z [xy\overline{z}, \overline{x}yw] \\ yz\colon z\overline{u}w \otimes_w [xy\overline{w}, \overline{x}yz] \otimes_{\overline{u}} [xu, \overline{x}yz] \end{aligned}$$

   The to-do list then contains the pairs:

$$\begin{array}{lll} [x\overline{y}, \overline{xyw}] & [xu, \overline{x}yw] & [xy\overline{z}, \overline{x}yw] \\ [x\overline{y}, \overline{xyz}] & [xu, \overline{x}yz] & [xy\overline{w}, \overline{x}yz] \end{array}$$

5. The chain $z\overline{u}w\colon \overline{uv}w \otimes_{\overline{v}} z\overline{u}w$ is pure, so we can simply reuse it. Furthermore, the clause $z\overline{u}w$ does not occur in T, so no additional chains must be derived.

6. The chain $x\overline{y}\colon \overline{u}v\overline{w} \otimes_v \overline{zuv} \otimes_{\overline{u}} xu \otimes_{\overline{w}} x\overline{y}w \otimes_{\overline{z}} x\overline{y}z$ is again impure, so we only need to derive the chains for the corresponding pairs in T, namely $\mathcal{D}(\overline{xyw}, \overline{x}, \rho)$ and $\mathcal{D}(\overline{xyz}, \overline{x}, \rho)$. These are given by:

$$\begin{aligned} \overline{yw}\colon \overline{u}v\overline{w} \otimes_v \overline{zuv} \otimes_{\overline{u}} [xu, \overline{xyw}] \otimes_{\overline{z}} [x\overline{y}z, \overline{xyw}] \\ \overline{yz}\colon \overline{u}v\overline{w} \otimes_v \overline{zuv} \otimes_{\overline{u}} [xu, \overline{xyz}] \otimes_{\overline{w}} [x\overline{y}w, \overline{xyz}] \end{aligned}$$

   The final to-do list then contains the pairs:

$$\begin{array}{llll} [xu, \overline{x}yw] & [xu, \overline{x}yz] & [xu, \overline{xyw}] & [xu, \overline{xyz}] \\ [xy\overline{z}, \overline{x}yw] & [xy\overline{w}, \overline{x}yz] & [x\overline{y}z, \overline{xyw}] & [x\overline{y}w, \overline{xyz}] \end{array}$$

Observe that the resolvents for pairs in the first line are RUPs because xu is a RAT upon x, and their chains are given in the DRAT proof from Example 1; the resolvents for pairs in the second line are RUPs, for both clauses are in the original CNF formula. The generated RUP refutation is then:

$$\begin{aligned} \sigma' = {} & yuw, yzu, \overline{y}u\overline{w}, \overline{yz}u, && \text{(RAT resolvents)} \\ & y\overline{z}w, yz\overline{w}, \overline{y}z\overline{w}, \overline{yz}w, && \text{(pure resolvents)} \\ & \overline{yw}, \overline{yz}, z\overline{u}w, yz, yw, \overline{y}, u, \square && \text{(generated RUPs)} \end{aligned}$$

Complexity considerations Each application of the clause isolation procedure may add to a refutation of length n at most n^2 clauses, since each pair $[L, R]$ spawns a new chain. Given a DRAT proof of length n containing r RAT inferences, the output refutation length is bounded by $2^r n^{2^r}$.

7 Conclusion

We have presented a general algorithm to transform DRAT proofs into RUP proofs, parameterized by a clause isolation procedure. Coupled with any method to generate interpolants from

RUP proofs, this solves the problem of interpolant generation from DRAT proofs, enabling satisfiability-preserving techniques for SAT solving in model checking.

We have introduced two methods for clause isolation. The first method regards RUP proofs as subsumption-resolution proofs. Relaxing the usual conditions of the resolution rule enables distributivity laws over clauses and inferences. The iterative application of these laws yields clause-isolated proofs.

Proofs generated through distributivity contain redundant and useless fragments. To palliate this, we lift the distributivity rules to a pseudo-distributive operator. A clause isolation method which works directly on the premises of the subsumption-merge chains that make up RUP inferences is attained. Part of the inherent clause sharing in RUP proofs is preserved, and useless tautological inferences are avoided.

This has a cummulative saving effect, for clause isolation is iteratively applied when performing RAT elimination. The complexity properties of our method are exponential, yet it is unclear that this can be avoided, due to complexity constraints on the size of interpolants generated from DRAT proofs.

Related work After conversion to RUP, interpolants can be extracted by extending interpolation rules to chains [20] or by splitting chains into hyper-resolution inferences [35]. Interpolation techniques that do not require proofs exist [5, 12] but rely on incremental solving, which clashes with the non-monotonicity of RAT inferences [31]; similar challenges arise in IC3 [11]. Moreover, proofless interpolation can be exponentially more expensive than proof-based techniques [20].

Proof transformations where resolution inferences are reordered have been proposed to change interpolant strength [16, 35], proof size [4, 9], or structure [15]. Unlike our transformation, these transformations convert chains into single resolution inferences rather than processing them directly.

Future work The methods presented here are a theoretical development. The obvious next step is an implementation of our RAT elimination method, as well as research into further optimizations in the clause isolation procedure. From a more practical perspective, we aim to compare the efficiency of these methods in a model checking setting. Given that interpolant size does not seem to significantly impact interpolant quality, the relevant question is whether the gains from using satisfiability-preserving inprocessing techniques for SAT solving outweigh the overhead introduced by RAT elimination in terms of total time spent in model checking. Given the connection between DRAT, extended resolution and circuits [34], the methods developed in this paper might also be applicable to circuit-aware simplifications for bounded model checking [37].

Acknowledgments We would like to thank our anonymous reviewers for their detailed and helpful comments. This work was supported by the Austrian Science Fund (FWF) under project W1255-N23, by the Vienna Science and Technology Fund (WWTF) under projects VRG11-005 and ICT15-103, and by Microsoft Research through its PhD Scholarship Programme.

References

[1] F. A. Aloul, A. Ramani, I. L. Markov, and K. A. Sakallah. Solving difficult instances of boolean satisfiability in the presence of symmetry. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 22(9):1117–1137, 2003.

[2] P. B. Andrews. Resolution with merging. *J. ACM*, 15(3):367–381, 1968.

[3] F. Baader and T. Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.

[4] O. Bar-Ilan, O. Fuhrmann, S. Hoory, O. Shacham, and O. Strichman. Reducing the size of resolution proofs in linear time. *Software Tools for Technology Transfer (STTT)*, 13(3):263–272, 2011.

[5] S. Bayless, C. G. Val, T. Ball, H. H. Hoos, and A. J. Hu. Efficient modular SAT solving for IC3. In *Formal Methods in Computer-Aided Design (FMCAD)*, pages 149–156. IEEE, 2013.

[6] A. Biere. Preprocessing and inprocessing techniques in SAT. In *Haifa Verification Conference (HVC)*, volume 7261 of *LNCS*, page 1. Springer, 2011.

[7] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1579 of *LNCS*, pages 193–207. Springer, 1999.

[8] A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, February 2009.

[9] R. Bloem, S. Malik, M. Schlaipfer, and G. Weissenbacher. Reduction of resolution refutations and interpolants via subsumption. In *Haifa Verification Conference (HVC)*, volume 8855 of *LNCS*, pages 188–203. Springer, 2014.

[10] M. L. Bonet, T. Pitassi, and R. Raz. No feasible interpolation for TC0-Frege proofs. In *Foundations of Computer Science (FOCS)*, pages 254–263. IEEE, 1997.

[11] A. R. Bradley. SAT-based model checking without unrolling. In *Verification, Model Checking and Abstract Interpretation (VMCAI)*, volume 6538 of *LNCS*, pages 70–87. Springer, 2011.

[12] H. Chockler, A. Ivrii, and A. Matsliah. Computing interpolants without proofs. In *Haifa Verification Conference (HVC)*, volume 7857 of *LNCS*, pages 72–85. Springer, 2012.

[13] W. Craig. Linear reasoning. A new form of the Herbrand-Gentzen theorem. *J. Symb. Log.*, 22(3):250–268, 1957.

[14] L. Cruz-Filipe, M. J. H. Heule, W. A. H. Jr., M. Kaufmann, and P. Schneider-Kamp. Efficient certified RAT verification. In *Conference on Automated Deduction (CADE)*, volume 10395 of *LNCS*, pages 220–236. Springer, 2017.

[15] V. D'Silva, D. Kroening, M. Purandare, and G. Weissenbacher. Restructuring resolution refutations for interpolation. Technical report, Oxford University, 2008.

[16] V. D'Silva, D. Kroening, M. Purandare, and G. Weissenbacher. Interpolant strength. In *Verification, Model Checking and Abstract Interpretation (VMCAI)*, volume 5944 of *LNCS*, pages 129–145. Springer, 2010.

[17] N. Eén and A. Biere. Effective preprocessing in SAT through variable and clause elimination. In *Theory and Applications of Satisfiability Testing (SAT)*, volume 3569 of *LNCS*, pages 102–104. Springer, 2005.

[18] A. V. Gelder. Producing and verifying extremely large propositional refutations - have your cake and eat it too. *Ann. Math. Artif. Intell.*, 65(4):329–372, 2012.

[19] E. I. Goldberg and Y. Novikov. Verification of proofs of unsatisfiability for CNF formulas. In *DATE*, pages 10886–10891. IEEE Computer Society, 2003.

[20] A. Gurfinkel and Y. Vizel. DRUPing for interpolants. In *Formal Methods in Computer-Aided Design (FMCAD)*. IEEE, 2014.

[21] M. Heule, M. Järvisalo, F. Lonsing, M. Seidl, and A. Biere. Clause elimination for SAT and QSAT. *J. Artif. Intell. Res.*, 53:127–168, 2015.

[22] M. Heule, W. A. H. Jr., and N. Wetzler. Trimming while checking clausal proofs. In *Formal Methods in Computer-Aided Design (FMCAD)*, pages 181–188. IEEE, 2013.

[23] M. Heule, W. A. H. Jr., and N. Wetzler. Verifying refutations with extended resolution. In *Conference on Automated Deduction (CADE)*, volume 7898 of *LNCS*, pages 345–359. Springer, 2013.

[24] M. J. H. Heule and O. Kullmann. The science of brute force. *Communications of the ACM*,

60(8):70–79, 2017.

[25] M. Järvisalo, M. Heule, and A. Biere. Inprocessing rules. In *International Joint Conference on Automated Reasoning (IJCAR)*, volume 7364 of *LNCS*, pages 355–370. Springer, 2012.

[26] B. Kiesl, A. Rebola-Pardo, and M. J. H. Heule. Extended resolution simulates DRAT. In *International Joint Conference on Automated Reasoning (IJCAR)*, volume 10900 of *LNCS*, pages 516–531. Springer, 2018.

[27] L. Kovács and A. Voronkov. First-order interpolation and interpolating proof systems. In *LPAR*, volume 46 of *EPiC Series in Computing*, pages 49–64. EasyChair, 2017.

[28] J. Krajícek and P. Pudlák. Some consequences of cryptographical conjectures for s_2^1 and EF. *Information and Computation*, 140(1):82–94, 1998.

[29] N. Manthey, M. Heule, and A. Biere. Automated reencoding of boolean formulas. In *Haifa Verification Conference (HVC)*, volume 7857 of *LNCS*, pages 102–117. Springer, 2012.

[30] K. L. McMillan. Interpolation and SAT-based model checking. In *Computer Aided Verification (CAV)*, volume 2725 of *LNCS*, pages 1–13. Springer, 2003.

[31] T. Philipp and A. Rebola-Pardo. Towards a semantics of unsatisfiability proofs with inprocessing. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 46 of *EPiC Series in Computing*, pages 65–84. EasyChair, 2017.

[32] P. Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *Journal of Symbolic Logic*, 62(3):981–998, 1997.

[33] A. Rebola-Pardo and L. Cruz-Filipe. Complete and efficient DRAT proof checking. In *Formal Methods in Computer-Aided Design (FMCAD)*, pages 1–9. IEEE, 2018.

[34] A. Rebola-Pardo and M. Suda. A theory of satisfiability-preserving proofs in SAT solving. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*, volume 57 of *EPiC Series in Computing*, pages 583–603. EasyChair, 2018.

[35] M. Schlaipfer and G. Weissenbacher. Labelled interpolation systems for hyper-resolution, clausal, and local proofs. *Journal of Automated Reasoning*, 57(1):3–36, 2016.

[36] G. Tseitin. On the complexity of proofs in propositional logics. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning: Classical Papers in Computational Logic 1967–1970*, volume 2. Springer, 1983. Originally published 1970.

[37] Y. Vizel, A. Gurfinkel, and S. Malik. Fast interpolating BMC. In *Computer Aided Verification (CAV)*, volume 9206 of *LNCS*, pages 641–657. Springer, 2015.

[38] Y. Vizel, G. Weissenbacher, and S. Malik. Boolean satisfiability solvers and their applications in model checking. *Proceedings of the IEEE*, 103(11):2021–2035, 2015.

[39] N. Wetzler, M. Heule, and W. A. H. Jr. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *Theory and Applications of Satisfiability Testing (SAT)*, volume 8561 of *LNCS*, pages 422–429. Springer, 2014.