



Enhancing Superposition Reasoning in Linear Real Arithmetic through Selection and Simplification (Extended Version)

Johannes Schoisswohl, Laura Kovács, Konstantin Korovin and
Andrei Voronkov

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

September 27, 2026

Enhancing Superposition Reasoning in Linear Real Arithmetic through Selection and Simplification (Extended Version)

Johannes Schoisswohl¹, Laura Kovács¹, Konstantin Korovin², and Andrei Voronkov^{2,3}

¹ TU Wien

² University of Manchester

³ EasyChair

Abstract

The calculus ALASCA extends superposition-based saturation proving with reasoning in Uninterpreted Functions and Linear Real Arithmetic (UFLRA). As efficiency of saturation algorithms depend on literal selection, simplification, and redundancy elimination, this paper improves reasoning in UFLRA by introducing theory-specific notions and methods for all of these techniques. The implementation of these methods in the first-order theorem prover VAMPIRE shows substantial gains in every tested benchmark category, solving a number of problems previously unsolved by any system.

This paper is the extended version of a paper with the same title which will be published at LPAR 2026.

1 Introduction

The increasing complexity of modern computing systems requires automated approaches to ensure functional correctness, reliability and security of such systems [34, 36]. Most of existing formal methods supporting such automation make use of automated reasoning engines under the hood. A powerful class of automated reasoners are ATPs, which are used to check validity of logical formulas in extensions of first-order logical theories. Despite significant advances [5, 10, 29], reasoning about the combination of quantifiers ($\forall, \exists$), arithmetic and uninterpreted functions still poses challenges. This paper addresses this problem and introduces techniques that vastly improve efficiency of automated reasoning for Uninterpreted Functions and Linear Real Arithmetic (UFLRA). Key to our work is saturation-based first-order theorem proving using the Abstracting Linear Arithmetic Superposition Calculus (ALASCA) [29], an extension of the superposition calculus [37, 2, 46] used in modern ATPs for first-order logic [7, 47, 43, 18, 13].

ALASCA extends the superposition calculus by adding dedicated inference rules for arithmetic. These rules replace the need of axiomatizing theory symbols, such as addition $+$ and order $>$, as done by early saturation-based approaches to arithmetic reasoning [40]. The calculus was shown to outcompete state-of-the-art SMT-solvers and first-order theorem provers [29]. Let us explain the main challenges one faces when applying ALASCA, and the saturation algorithm implementing it, to UFLRA. Solving these challenges is the main motivation for our work.

Saturation & ALASCA. Given an input problem F , saturation theorem provers first negate F and clausify the negated input $\neg F$, obtaining a set of clauses; this set of clauses forms the initial search space. After that, one repeatedly applies a set of inference rules to clauses in the search space, adding conclusions of these inferences to the search space. The goal of the proving

process is to derive the empty clause, denoted as $\square$ which corresponds to a contradiction. Thus once $\square$ is derived, unsatisfiability of the input formula F is established and proving terminates.

Saturation algorithms may build very large search spaces; in practice, often containing millions of clauses. Thus, methods for making the search space smaller, can vastly improve performance. For handling this issue a number of concepts have been developed for superposition. Important for our work are *literal selection*, *redundancy* and *simplification*. Literal selection restricts the number of applicable inferences, and thus limits growth of the search space, redundancy allows for removing unnecessary clauses from the search space, and using simplification one can replace formulas by simpler ones instead of adding new clauses to the search space.

Even though these techniques have been well-studied in the context of superposition [2, 37, 33, 25, 22, 46], they cannot be straightforwardly used in ALASCA. ALASCA uses, among other rules, extensive normalizations, in particular rewriting arithmetic literals to a normal form $t \diamond 0$, where t is an arithmetic term and $\diamond \in \{\approx, \neq, >, \geq\}$. These normal forms are required for ALASCA to work in a sound and complete manner, but they interfere with (i) literal selection, (ii) equality reasoning and rewriting, and (iii) the detection of tautological clauses. Steps (i)-(iii) are however indispensable to keep proof search tractable. Our paper proposes new solutions for (i)-(iii), as detailed next.

Enhancing ALASCA. *Literal Selection* guides proof search, by selecting a subset of the literals of a clause that needs to be considered for making inferences with. In superposition, given a clause $C = L_1 \vee \dots \vee L_n$, a *selection function* sel determines a subset $\text{sel}(C)$ of the *negative* literals of C which needs to be considered for inferences. Inference with literals not in $\text{sel}(C)$ can be skipped, thus limiting the growth of the search space without compromising the solver's completeness [2]. Due to normalizations like $\neg(t > 0) \Rightarrow -t \geq 0$ negative inequality literals do not exist in ALASCA, which prevents the use of standard literal selection. We address this by extending literal selection to a wider class of so-called *selectable* literals (Sect. 5). A peek at our illustrating example (Ex. 1, Sect. 2) shows how this can tremendously improve proof search: Given two clauses $C = \bigvee_{i=1}^n -f(x_i, a_i) + x_i > 0$ and $D = f(x, x) - x \geq 0$, in the standard ALASCA approach of [29] there are n potential inferences resolving D with any of the n different literals of C . In our proposed advancement of ALASCA with literal selection, only one of these inferences needs to be performed, leading to an exponentially smaller search space.

Demodulation is one of the most widely used simplification rules in superposition [33]. Given an equality clause $l \approx r$ where r is simpler than l ($r \prec l$), it allows to *replace* any clause $C[l]$ with $C[r]$, discarding $C[l]$ from the search space without considering it for any further inferences. In the presence of arithmetic however, demodulation is less effective because it requires the term l in $C[l]$ to coincide with l in $l \approx r$. This, for example, would prevent $p(a + b)$ to be rewritten by $b + a \approx c$ into $p(c)$, despite the fact that $a + b$ and $b + a$ are semantically equal. One can also give many other similar examples, for example, rewriting $p(a + b)$ by $a + 1 \approx c$ into $p(b + c - 1)$, provided that $b + c - 1$ is smaller than $a + b$. Even worse, real-valued equalities are normalized to the form $t \approx 0$ in ALASCA, for which applying demodulation mostly fails. We address these shortcomings by formulating *ALASCA Demodulation (Dmd)*, a new variant of demodulation that works modulo linear arithmetic. Ex. 1 shows how Dmd improves proof search. Given a unit equality $g(x, y) + x - y \approx 0$, the standard demodulation rule would try to rewrite $g(x, y) + x - y$ into 0, which cannot be done in a clause $\bigvee_{i=1}^n -f(x_i, a_i) + g(y, x_i) > 0 \vee y > 0$, as the left-hand side of the equality does not match any subterm of the clause. Dmd in contrast rewrites $g(x, y)$ into $y - x$, and is thus more widely applicable, resulting in more simplifications and an exponential reduction of the search space size.

Inequality Merging is a new inference rule that removes arithmetical tautologies and simplifies clauses. In the pure propositional case, tautologies (e.g. $L \vee \neg L \vee C$) can be detected and deleted

easily. When dealing with arithmetic, we might face more complex tautologies like $a \geq 0 \vee -a \not\approx 0$, which are harder to discover, and might pollute our search space with unnecessary clauses. ALASCA’s normalizations introduce such tautologies, for example by turning $t > 0 \vee \neg(t > 0)$ into $t > 0 \vee -t \geq 0$. To address such and similar issues, we introduce the *new inference rule Inequality Merging (IM)*, which generalizes the idea of detecting tautologies and allows to simultaneously perform simplifications like $t > 0 \vee -t \not\approx 0 \Rightarrow -t \not\approx 0$. This greatly improves proof search as it can detect and remove redundant clauses, and also allows to reason about multiple literals at once (e.g. by processing $-t \not\approx 0$), instead of applying inferences to each individual literal (e.g. $t > 0$, and $-t \not\approx 0$). Ex. 1 shows how IM can drastically shrink proof length.

Our contributions are summarized as follows:

- We show how *Literal Selection* can be used in ALASCA by extending selection to so-called *selectable*, and not just negative, literals (Sect. 5). This results in fewer inferences and hence reduces the size of the search space.
- We introduce *ALASCA Demodulation (Dmd)*, which allows one to rewrite clauses by equalities modulo Linear Real Arithmetic (LRA) (Sect. 4). This drastically improves proof search by eliminating many inferences that would add new clauses to the search space.
- We introduce a new inference rule, called *Inequality Merging (IM)*, for simplifying clauses with multiple inequalities containing the same term and removing tautological clauses (Sect. 4). This speeds up proof search by removing tautologies and by avoiding repetitive work.
- We introduce $\text{ALASCA}^{\text{sel}}$, an enhanced variant of ALASCA, which integrates all of these techniques (Sect. 3.2).
- We implemented our work in the theorem prover VAMPIRE [33] and evaluated it against existing methods for automating reasoning in UFLRA (Sect. 6). Our experiments show the practical benefits of $\text{ALASCA}^{\text{sel}}$, solving 84 more problems than state-of-the-art systems in UFLRA reasoning.

Outline. This paper is structured as follows. Sect. 2 gives an example that illustrates the main new techniques introduced in this paper. Sect. 3 introduces a variant of the original ALASCA [29] which integrates with the methods introduced in the following sections. Sect. 4 introduces our new simplification and redundancy elimination rules, while Sect. 5 shows how literal selection can be done in ALASCA. The implementation of our work is evaluated in Sect. 6, showing that, our work improves automated reasoning in UFLRA. We show that, with the new extensions, ALASCA-based reasoning solves problems that could not be solved before by other state-of-the-art provers. In Sect. 7 the paper is positioned within relevant related work in automated reasoning in UFLRA, and finally Sect. 8 concludes the paper.

2 Illustrative Example

We showcase our reasoning enhancements upon ALASCA via Ex. 1.

$C_1:$	$g(x, y) + x - y \approx 0$	input
$C_2:$	$\bigvee_{i=1}^n -f(x_i, a_i) + g(y, x_i) > 0 \vee y > 0$	input
$C_3:$	$-f(x, y) + y \approx 0 \vee f(x, y) - y > 0 \vee h(x) - h(y) \not\approx 0$	input
$C_4^1:$	$-f(x_1, a_1) + x_1 - y > 0 \vee \bigvee_{i=2}^n -f(x_i, a_i) + g(y, x_i) > 0 \vee y > 0$	Dmd using C_1, C_2
$C_4^k:$	$\bigvee_{i=1}^k -f(x_i, a_i) + x_i - y > 0 \vee \bigvee_{i=k+1}^n -f(x_i, a_i) + g(y, x_i) > 0 \vee y > 0$	Dmd using C_1, C_4^{k-1}
$C_4^n:$	$\bigvee_{i=1}^n -f(x_i, a_i) + x_i - y > 0 \vee y > 0$	Dmd using C_1, C_4^{n-1}
$C_5:$	$\bigvee_{i=1}^n -f(x_i, a_i) + x_i > 0$	VE using C_4^n
$C_6:$	$f(x, y) - y \geq 0 \vee h(x) - h(y) \not\approx 0$	IM using C_3
$C_7:$	$f(x, x) - x \geq 0$	TF using C_6
$C_8^1:$	$\bigvee_{i=2}^n -f(x_i, a_i) + x_i > 0$	FM using C_5, C_7
$C_8^k:$	$\bigvee_{i=k+1}^n -f(x_i, a_i) + x_i > 0$	FM using C_8^{k-1}, C_7
$C_8^n:$	$\square$	FM using C_8^{n-1}, C_7

Figure 1: Proof of unsatisfiability of C from Ex. 1. Inference rules indicated on the side can be found in Figs. 4 and 5. Normalizations using inference rules $\text{NF} \cup \text{Ctd}$ are omitted.

Example 1 (Illustrative Example). Consider the set of clauses $C = \{C_1, C_2, C_3\}$

$$\begin{aligned}
& \forall x, y. g(x, y) + x - y \approx 0 & C_1 \\
& \forall x_1, \dots, x_n, y. \bigvee_{i=1}^n -f(x_i, a_i) + g(y, x_i) > 0 \vee y > 0 & C_2 \\
& \forall x, y. -f(x, y) + y \approx 0 \vee f(x, y) - y > 0 \vee h(x) - h(y) \not\approx 0 & C_3
\end{aligned}$$

where a_i are constants, and f, g, h are uninterpreted functions over $\mathbb{R}$.

Our enhanced ALASCA easily proves unsatisfiability of C for instances up to $n = 50$ within 10 seconds, generating only 208 clauses. Yet, ALASCA without our extensions times out for $n = 3$ after 10 seconds, while generating 49508 intermediate clauses. Fig. 1 shows the unsatisfiability proof of C , using our enhanced ALASCA approach. In the sequel, we detail the key steps of this proof and explain why disabling either of our new enhancements causes ALASCA to fail (dis)proving Ex. 1 for large n .

Enhanced ALASCA proof. In the first steps of the proof of Fig. 1, we use *our new rule ALASCA Demodulation* (Dmd) with the clause C_1 to rewrite $g(y, x_1) \Rightarrow x_1 - y$ in C_2 deriving C_4^1 , and then rewrite the result again by C_1 until we arrive at the clause $C_4^n : \bigvee_{i=1}^n -f(x_i, a_i) + x_i - y > 0 \vee y > 0$. Dmd is a simplification rule, which means whenever we use it to rewrite a clause, the original clause becomes redundant, that is, the original clause can be discarded from the search space. C_4^n contains so-called unshielded variable y ; a variable that does not occur under any uninterpreted function. This unshielded variable will be processed by the rule Variable Elimination (VE)¹, turning C_4^n into the equivalent clause $C_5 : \bigvee_{i=1}^n -f(x_i, a_i) + x_i > 0$ by substituting 0 for y .

For the input clause C_3 , we use *our new rule Inequality Merging* (IM) to simplify C_3 into the equivalent clause $C_6 : f(x, y) - y \geq 0 \vee h(x) - h(y) \not\approx 0$, and then use the rule Term Factoring (TF) to derive clause $C_7 : f(x, x) - x \geq 0$.

¹ Inference rules can be found in Fig. 4 (Page 9) and Fig. 5 (Page 12).

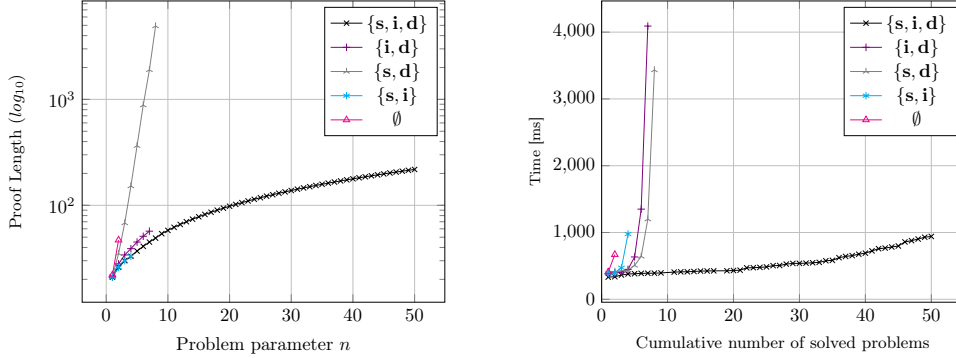


Figure 2: Results of running our enhanced ALASCA implementation (see Sect. 6) on Ex. 1 up to $n = 50$, with a timeout of 10 seconds. Configurations are denoted by the set of enabled reasoning features, where **s** means Literal Selection; **d** means ALASCA Demodulation (Dmd), and **i** means Inequality Merging (IM).

All literals $-f(x_i, a_i) + x_i > 0$ of C_5 are selectable² (see Sect. 5). This means we can select an arbitrary single literal from C_5 to apply inferences with, and do not need to make inferences with any other literal of C_5 . We apply *Literal Selection* choosing the first literal ($\text{sel}(C_5) = \{-f(x_1, a_1) + x_1 > 0\}$) and use the rule Fourier-Motzkin (FM) to resolve C_5 and C_7 deriving $C_8^1 : \bigvee_{i=2}^n -f(x_i, a_i) + x_i > 0$. Of the result we select the first literal again and resolve it with C_7 , until we finally arrive at C_8^n which is the empty clause $\square$. This step represents a contradiction and concludes our proof of unsatisfiability of C . We note that choosing different literals for selection would lead to deriving $\square$ in essentially the same way.

Need for ALASCA enhancements. Fig. 2 shows the results of running our enhanced ALASCA on Ex. 1. Our evaluation is conducted using our implementation in the VAMPIRE prover (see Sect. 6). As shown in Fig. 2, we evaluate the impact of Literal Selection (**s**), Dmd (**d**), and IM (**i**). We evaluate ALASCA with all our enhancements enabled ($\{\mathbf{s}, \mathbf{i}, \mathbf{d}\}$), as well as with either of the techniques disabled ($\{\mathbf{s}, \mathbf{d}\}$, $\{\mathbf{s}, \mathbf{i}\}$, $\{\mathbf{i}, \mathbf{d}\}$). We also evaluate ALASCA with all of our improvements disabled ($\emptyset$). The plot on the left hand side of Fig. 2 indicates the length of found proofs for each n . The plot on the right hand side of Fig. 2 gives the number of problems solved by each ALASCA enhancement and configuration. Let us now explain why all three of our new reasoning techniques are necessary to solve Ex. 1 in reasonable time.

ALASCA Demodulation (Dmd). If Dmd is disabled (conf. $\{\mathbf{s}, \mathbf{i}\}$), the rewrites obtained using C_1 in order to obtain C_4^k are performed using the standard superposition (**Sup**), instead of Dmd. **Sup** is however a generating, not simplifying, inference rule: it generates and adds clauses to the search spaces, without checking whether premises become redundant (and could be removed). For example, when rewriting a clause $-f(x_1, a_1) + g(y, x_1) > 0 \vee -f(x_2, a_2) + g(y, x_2) > 0 \vee y > 0$ to $-f(x_1, a_1) + x_1 - y > 0 \vee -f(x_2, a_2) + g(y, x_2) > 0 \vee y > 0$, the former clause will not be removed from the search space. Superposition is therefore applied to its second literal and derives $-f(x_1, a_1) + g(y, x_1) > 0 > 0 \vee -f(x_2, a_2) + x_2 - y > 0 \vee y > 0$; both of these clauses can however be rewritten to the same clause C_4^2 . This means, we also obtain the clause C_4^2 , but derive exponentially many unnecessary consequences on the way: for arbitrary n we will derive

²as their only $\prec$ -maximal term $f(x_i, a_i)$ occurs with a negative sign

$\bigvee_{i \in I} -f(x_i, a_i) + g(y, x_i) > 0 \vee \bigvee_{j \in \bar{I}} -f(x_j, a_j) + g(y, x_j) > 0 \vee y > 0$ for any $I \subseteq \{1 \dots n\}$. Fig. 2 shows that proof lengths with this configuration are the same as for $\{\mathbf{s}, \mathbf{i}, \mathbf{d}\}$ since the same proofs can be found, but the solver times out for $n = 5$.

Inequality Merging (IM). If we disable IM (conf. $\{\mathbf{s}, \mathbf{d}\}$), the proof of Fig. 1 can be performed only up to step C_5 . However, we cannot use IM on C_3 to further derive C_6 , from which we would infer C_7 . Rather, we directly obtain the clause $D_7 : f(x, x) - x > 0 \vee -f(x, x) + x \approx 0$ from C_3 . When combining D_7 with C_5 in the same way as with C_7 in the proof of Fig. 1, we get $D_8^k : \bigvee_{i=k}^n -f(x_i, a_i) + x_i > 0 \vee \bigvee_{i=1}^{k-1} -f(a_i, a_i) + a_i \approx 0$, arriving at $D_8^n : \bigvee_{i=1}^n -f(a_i, a_i) + a_i \approx 0$ instead of the empty clause C_8^n . Derivation of the empty clause from here comes at a massive cost in proof length, yielding a timeout after 10 seconds for $n = 9$ (see Fig. 2). Looking at the length of proofs (Fig. 2), we also observe an approximately exponential growth of the proof length in terms of n .

Literal Selection. Disabling literal selection (conf. $\{\mathbf{i}, \mathbf{d}\}$) implies that FM needs to be applied for every literal of C_5 instead of one single selected literal. This gives an exponential blow-up in search space as we can derive all clauses $\bigvee_{i \in I} -f(x_i, a_i) + x_i > 0$ for $I \subseteq \{1 \dots n\}$ from C_5, C_7 and their successors, which yields a timeout for $n = 8$ (Fig. 2).

In summary, we conclude that the combination of Literal Selection, ALASCA Demodulation, and Inequality Merging is essential for solving Ex. 1.

3 ALASCA Revised

In what follows, we use ALASCA_0 to refer to the calculus ALASCA from [29]. This section introduces $\text{ALASCA}_{\prec}^{\text{sel}}$, a variant of ALASCA_0 which allows integrating simplification and selection techniques we will introduce in the following sections. Changes to the old calculus were made in order to make the newly introduced simplification rules Inequality Merging (IM), and ALASCA Demodulation (Dmd) compatible with $\text{ALASCA}_{\prec}^{\text{sel}}$ ³.

We introduce necessary background and fix notation (Sect. 3.1), before defining the calculus $\text{ALASCA}_{\prec}^{\text{sel}}$ and introducing conditions required for its parameters $\prec$, and sel (Sect. 3.2).

3.1 Notation and Background Knowledge

We consider a first-order signature of UFLRA, with a single sort⁴. This sort contains the function symbol $+$, the constant 1, the unary predicate symbols $(\approx 0), (> 0), (\geq 0)$, and an infinite set of unary functions $\mathbb{Q}$ called the numeral multiplications, representing multiplications by a constant factor.

Clauses are multisets of literals $L_1 \vee \dots \vee L_n$. We call a clause unit if $n = 1$. The empty clause $n = 0$ is denoted by $\square$. By $\mathbf{F}, \mathbf{V}, \text{Trm}, \text{Lit}$, and Cls we denote the set of all first-order function symbols, variables, terms, literals, and clauses respectively. We use $\mathbf{gTrm}, \mathbf{gLit}$ for the respective ground (variable-free) subsets.

We say f is *uninterpreted* if $f \notin \{+\} \cup \mathbb{Q}$ and a term t is *atomic* if it is a variable or its top-level symbol $\mathbf{sym}(t)$ is uninterpreted. In the sequel, let $x \in \mathbf{V}, f \in \mathbf{F}, k \in \mathbb{Q}, \diamond \in \{>, \geq, \approx, \not\approx\}, s, t \in \text{Trm}, L \in \text{Lit}, C \in \text{Cls}$ and $\prec$ be a preorder.

For the sake of readability we sometimes abbreviate numeral terms, e.g. 2 for 2(1). Literals are considered to be normalized to $s \diamond 0$ implicitly. **AC** are the axioms of associativity and commutativity of $+$. We define the function $\circ^-$ which negates all the coefficients of a term (e.g., $(a + -2b - c)^- = -a + 2b + c$), and we use $\circ^+$ for the identity function ($t^+ = t$). By $\equiv_T$

³For a detailed comparison between ALASCA_0 and $\text{ALASCA}_{\prec}^{\text{sel}}$, we refer to Appendix A.

⁴Our work can be easily extended to additional uninterpreted sorts.

we denote equivalence modulo T . We write $s \trianglelefteq_T t$ ($s \triangleleft_T t$) for s being a (strict) subterm of t modulo T .

We define $\langle \text{coeff}(kt), \text{trm}(kt) \rangle = \langle k, t \rangle$, and $\langle \text{coeff}(t), \text{trm}(t) \rangle = \langle 1, t \rangle$ if $\text{sym}(t) \notin \mathbb{Q}$. We call the multiset $\{t_i \mid i \in I\} = \underline{\text{smnds}}(\sum_{i \in I} t_i)$ the *summands* of $\sum_i t_i$.

We call $\underline{\text{smnds}}^\prec(t)$ the weakly maximal summands of t , defined as $\underline{\text{smnds}}^\prec(t) = \{t_0 \in \underline{\text{smnds}}(t) \mid \forall t_1 \in \underline{\text{smnds}}(t) \setminus \{t_0\}. \text{trm}(t_0) \not\prec \text{trm}(t_1)\}$, and define the strictly maximal summands $\underline{\underline{\text{smnds}}}$ (up to $\equiv_{\mathbf{AC}}$) analogously, but replacing $\not\prec$ by $\not\leq$. We use double underlining to mark strictly maximal summands of a term ($k \underline{\underline{\text{smnds}}} + t$), single underlining for weakly maximal summands ($k \underline{\text{smnds}} + t$), and dots for arbitrary summands ($k \dot{\text{smnds}} + t$). That is $k \underline{\underline{\text{smnds}}} + t$ represents some term containing a strictly maximal summand s' with $\text{coeff}(s') = k$, and $\text{trm}(s') = s$. We drop the relation $\prec$ from the notation where there is no ambiguity. We define $\underline{\text{trms}}(s) = \{\text{trm}(t) \mid \text{trm}(\underline{\text{smnds}}(s))\}$ and define $\underline{\underline{\text{trms}}}$, $\underline{\text{trms}}^\prec$ analogously.

We call a literal $t \diamond 0$ *stably positive* ($t \diamond 0 \in \text{Lit}^+$) iff it is an equality, or it is an inequality and $\forall t_i \in \underline{\text{smnds}}(t). \text{coeff}(t_i) > 0 \ \& \ \text{trm}(t_i) \notin \mathbf{V}$. We call it *stably negative* ($t \diamond 0 \in \text{Lit}^-$) if it is a disequality, or it is an inequality and $\forall t_i \in \underline{\text{smnds}}(t). \text{coeff}(t_i) < 0 \ \& \ \text{trm}(t_i) \notin \mathbf{V}$. We call it *undecided* ($t \diamond 0 \in \text{Lit}^?$) otherwise.

A selection function sel is a function that maps a clause to a subset of its literals. We define the *leading literals* $\text{leadLit}(C)$, and *leading terms* $\text{leadLit}^{\text{sel}}(C)$ of C to be $\text{leadLit}^{\text{sel}}(C) = \text{sel}(C)$ if $\text{sel}(C) \neq \emptyset$, and otherwise $\text{leadLit}^{\text{sel}}(C) = (\underline{\text{lit}}(C) \cap \text{Lit}^+) \cup (\underline{\text{lit}}(C) \cap (\text{Lit} \setminus \text{Lit}^+))$. Further $\text{lead}^{\text{sel}}(C) = \{\sum \underline{\text{smnds}}(L) \mid L \in \text{leadLit}^{\text{sel}}(C)\}$, where $\underline{\text{lit}}(C)$ and $\underline{\underline{\text{lit}}}(C)$ are the weakly and strictly maximal literals in C . We leave away $\prec$ and sel if there is no ambiguity.

We say x is a *top level variable* of any $k \underline{\underline{\text{smnds}}} + t$. x is *shielded* in t iff there is a t' such that $x \triangleleft t' \trianglelefteq t$ where $\text{sym}(t')$ is uninterpreted, or unshielded otherwise.

By $s[\circ]$ we denote a term (literal, clause) with a single occurrence of a designated variable $\circ \notin \mathbf{V}$. By $s[t]$ we denote the term obtained from replacing t for $\circ$. We call $s[\circ]$ a *term (literal, clause) context*. We call a context $C[\circ]$ *uninterpreted*, iff $f(\dots, \circ, \dots) \trianglelefteq C[\circ] \implies f$ is an uninterpreted function.

We say $\prec$ is T -compatible iff $s \equiv_T s' \prec t' \equiv_T t \implies s \prec t$. A simplification ordering modulo T is a binary relation $\prec$ on Trm that is closed under ground substitutions and, for $\mathbf{gTrm}$, it is a T -compatible well-founded monotonic strict partial ordering that is total modulo T and fulfils the subterm property.

As shown in [32] there is no complete calculus of UFLRA wrt. the standard model $\mathbb{R}$. Therefore in [29] completeness of the original ALASCA is shown wrt. to a wider class of models which are defined by the axiomatisation $\mathcal{A}$ given in Fig. 3.

3.2 The Calculus $\text{ALASCA}_{\prec}^{\text{sel}}$

The base inference system of $\text{ALASCA}_{\prec}^{\text{sel}}$ is given in Fig. 4. It lists all its inferences except for the new simplification rules ALASCA Demodulation (Dmd) and Inequality Merging (IM), which are presented in Fig. 5 and discussed in Sect. 4.

We point out that inferences make use of an ordering $\prec$, Unification With Abstraction (UWA), and a selection function sel . The selection function will be discussed in Sect. 5; thus for this section it is sufficient to consider a trivial empty selection function $\forall C. \text{sel}(C) = \emptyset$.

UWA, as used in side conditions like $\langle \sigma, \mathcal{E} \rangle = \text{UWA}(s, t)$, is an alternative to Unification modulo a background theory – in our case modulo $\mathbf{AC}$ – which is defined in terms of an abstraction oracle ao [8]. It returns $\langle \sigma, \mathcal{E} \rangle$, a set of literals $\mathcal{E}$ and a substitution σ , such that $\neg \mathcal{E} \implies s\sigma \equiv_{\mathbf{AC}} t\sigma$, or it returns $\perp$ if s, t are not unifiable modulo $\mathbf{AC}$. An example could be

$$\begin{array}{ll}
\mathcal{A} = \mathcal{A}_{\text{eq}} \cup \mathcal{A}_{\text{ineq}} & \mathbf{AC} = \{x + y \approx y + x\} \\
\mathcal{A}_{\text{ineq}} = \{x < y \wedge y < z \rightarrow x < z\} & \cup \{x + (y + z) \approx (x + y) + z\} \\
\cup \{x < y \rightarrow x + z < y + z\} & \mathcal{A}_{\text{eq}} = \mathbf{AC} \\
\cup \{x < y \vee x > y \vee x \approx y\} & \cup \{j\mathbf{x} + kx \approx (j + k)x \mid j, k \in \mathbb{Q}\} \\
\cup \{\neg(x < x)\} & \cup \{j(k(x)) \approx (jk)x \mid j, k \in \mathbb{Q}\} \\
\cup \{x \leq y \leftrightarrow (x < y \vee x \approx y)\} & \cup \{1(x) \approx x\} \\
\cup \{x > 0 \rightarrow kx > 0 \mid k \in \mathbb{Q}, k > 0\} & \cup \{k(x + y) \approx kx + ky \mid k \in \mathbb{Q}\} \\
\cup \{1 > 0\} & \cup \{x + 0 \approx x, 0x \approx 0\}
\end{array}$$

Figure 3: The axiom set $\mathcal{A}$ of the ALASCA calculus from [29]. All axioms are implicitly universally quantified. We note that this axiomatisation has both $\mathbb{R}$ and $\mathbb{Q}$ as models.

applying FM to $-g(x, y + z) + c > 0$ and $g(a, b + c) > 0$, resulting in $\sigma = \{x \mapsto a\}$, $\mathcal{E} = \{y + z \not\approx b + c\}$, and thus in the conclusion $c > 0 \vee y + z \not\approx b + c$. We can read this as “if $y + z$ and $b + c$ are equal (unify), then $c > 0$ ”.⁵

Similarly to literal selection, orderings serve to guide proof search. They help limiting the number of applicable inferences, by restricting the literals and terms to use for inferences. For example, the rule FM only resolves maximal terms s (as defined by the syntax $k\underline{s} + t$). Thus, if the ordering asserts that $a \succ b$, then a FM inference can be performed with $a + b > 0$ and $-a > 0$, but not with $a + b > 0$ and $-b > 0$. We require the term ordering to fulfil Def. 1 and 2⁶, ensuring behaviour analogous to the completeness proof in [29], while making our new simplification rules compatible with the calculus (see Lem. 1). The concrete orderings used in our implementation are given in Sect. 3.3.

Definition 1 (ALASCA Term Ordering). *We say a relation $\prec$ is an ALASCA term ordering iff*

1. *it is a simplification ordering modulo $\mathbf{AC}$,*
2. *for any $k\underline{s} + t \in \mathbf{gTrm}$, where s is atomic, we have $s \succ -\frac{1}{k}t$, and*
3. *for any $s \approx t \in \mathcal{A}_{\text{eq}}^{\overline{\mathbf{AC}}}$, and ground substitution θ we have $s\theta \succ t\theta$, where $\mathcal{A}_{\text{eq}}^{\overline{\mathbf{AC}}} = \mathcal{A}_{\text{eq}} \setminus \mathbf{AC}$.*

Definition 2 (ALASCA Ordering). *We say $\prec$ is an ALASCA ordering iff for any $s, s_i, t, t_i \in \mathbf{gTrm}$*

1. *$\prec$ is an ALASCA term ordering*
2. *$s \prec t \implies s \diamond 0 \prec t \diamond 0$*
3. *$\prec$ is closed under ground substitutions, and for $\mathbf{gLit}$ it is an $\mathbf{AC}$ -compatible well-founded strict partial ordering that is total modulo $\mathbf{AC}$.*
4. *$s_1 \prec s_2 \implies j\underline{s_1} + t \diamond_1 0 \prec k\underline{s_2} + u \diamond_2 0$ where $\diamond_i \in \{>, \geq, \approx, \not\approx\}$*
5. *$\forall i. t_i \prec s \implies k_1 t_1 + \dots k_n t_n \diamond_1 0 \prec k\underline{s} + t \diamond_2 0$ where $\diamond_i \in \{>, \geq, \approx, \not\approx\}$*

⁵Appendix B gives the concrete **ao** we use in our implementation which fulfils the properties required for abstraction oracles from [8].

⁶Proof in Appendix C

Fourier-Motzkin

$$\frac{C_1 \vee j \underline{s} + t_1 \gtrsim_1 0 \quad C_2 \vee -k \underline{s}' + t_2 \gtrsim_2 0}{(C_1 \vee C_2 \vee kt_1 + jt_2 > 0) \sigma \vee \mathcal{E}} \text{ (FM)}$$

where

- $\langle \sigma, \mathcal{E} \rangle = \text{UWA}(s, s')$
- $js + t_1 \gtrsim_1 0 \not\vdash_{\text{sel}}^\sigma C_1$
- $-ks' + t_2 \gtrsim_2 0 \not\vdash_{\text{sel}}^\sigma C_2$
- $\{>\} \subseteq \{\gtrsim_1, \gtrsim_2\} \subseteq \{>, \geq\}$
- $j, k > 0$

Superposition

$$\frac{C_1 \vee k \underline{s} + t \approx 0 \quad C_2 \vee L[s']}{(C_1 \vee C_2 \vee L[-\frac{1}{k}t]) \sigma \vee \mathcal{E}} \text{ (Sup)}$$

where

- $\langle \sigma, \mathcal{E} \rangle = \text{UWA}(s, s')$
- $k \underline{s} + t \approx 0 \not\vdash_{\text{sel}}^\sigma C_1$
- $(s') \sigma \triangleleft u \in \text{lead}((L[s'] \vee C_2) \sigma)$
- $(k \underline{s} + t \approx 0) \sigma \not\vdash L[s'] \sigma$
- s is atomic

Equality Factoring

$$\frac{C \vee j \underline{s} + t_1 \approx 0 \vee k \underline{s}' + t_2 \approx 0}{(C \vee kt_1 - jt_2 \approx 0 \vee js + t_1 \approx 0) \sigma \vee \mathcal{E}} \text{ (EF)}$$

where

- $\langle \sigma, \mathcal{E} \rangle = \text{UWA}(s, s')$
- $ks' + t_2 \approx 0 \not\vdash_{\text{sel}}^\sigma C \vee js + t_1 \approx 0$

Normalization

$$\frac{C \vee L[t_1]}{C \vee L[t_2]} \text{ (NF)}$$

where

- $t_1 \triangleleft u \in \text{lead}(L[t_1] \vee C)$
- $t_1 \approx t_2 \in \left\{ \begin{array}{l} 0u \approx 0, \quad 0 \cdot 1 + u \approx u, \\ j(ku) \approx (jk)u, \quad 1(u) \approx u, \\ k(u+v) \approx ku + kv, \\ j \cdot u + k \cdot u + v \approx (j+k)u + v, \end{array} \middle| \begin{array}{l} u, v \in \text{Trm} \\ j, k \in \mathbb{Q} \end{array} \right\}$

Abstraction

$$\frac{C[k \underline{x} + t]}{C[y] \vee kx + t \not\approx y} \text{ (Abs)}$$

where

- $x \in \mathbf{V}$
- y is a fresh variable
- $C[o]$ is an uninterpreted clause context

Tight Fourier-Motzkin

$$\frac{C_1 \vee j \underline{s} + t_1 \geq 0 \quad C_2 \vee -k \underline{s}' + t_2 \geq 0}{(C_1 \vee C_2 \vee kt_1 + jt_2 > 0 \vee -ks' + t_2 \approx 0) \sigma \vee \mathcal{E}} \text{ (TFM)}$$

where

- $\langle \sigma, \mathcal{E} \rangle = \text{UWA}(s, s')$
- $js + t_1 \geq 0 \not\vdash_{\text{sel}}^\sigma C_1$
- $-ks' + t_2 \geq 0 \not\vdash_{\text{sel}}^\sigma C_2$
- $j, k > 0$

Inequality Factoring

$$\frac{C \vee j \underline{s} + t_1 \gtrsim_1 0 \vee k \underline{s}' + t_2 \gtrsim_2 0}{(C \vee kt_1 - jt_2 \gtrsim_3 0 \vee k \underline{s}' + t_2 \gtrsim_2 0) \sigma \vee \mathcal{E}} \text{ (IF)}$$

where

- $\langle \sigma, \mathcal{E} \rangle = \text{UWA}(s, s')$
- $ks' + t_2 \gtrsim_2 0 \not\vdash_{\text{sel}}^\sigma C \vee js + t_1 \gtrsim_1 0$
- or $js + t_1 \gtrsim_1 0 \not\vdash_{\text{sel}}^\sigma C \vee ks' + t_2 \gtrsim_2 0$
- $\gtrsim_1, \gtrsim_2 \in \{>, \geq\}$
- $\gtrsim_3 = \begin{cases} \geq & \text{if } \gtrsim_1 = \geq, \text{ and } \gtrsim_2 = > \\ > & \text{else} \end{cases}$
- $j, k > 0$

Term Factoring

$$\frac{L[j \underline{s} + k \underline{s}' + t] \vee C}{(L[(j+k)s + t] \vee C) \sigma \vee \mathcal{E}} \text{ (TF)}$$

where

- $\langle \sigma, \mathcal{E} \rangle = \text{UWA}(s, s')$
- $(j \underline{s} + k \underline{s}' + t) \sigma \triangleleft u \in \text{lead}((L[j \underline{s} + k \underline{s}' + t] \vee C) \sigma)$

Variable Elimination

$$\frac{C \vee \bigvee_{i \in I} (k_{i \cdot \underline{x}} + t_i \diamond_i 0)}{\bigwedge_{j \in J} (C \vee C_j)} \text{ (VE)}$$

where

- $\forall i. \diamond_i \in \{\approx, \not\approx, >, \geq\}$
- $\{C_j \mid j \in J\} = \text{QE}_x(\bigvee_{i \in I} (k_{i \cdot \underline{x}} + t_i \diamond_i 0))$
- QE_x is the universal quantifier elimination of the variable x , for details of this rule we refer to [29]
- $x \in \mathbf{V}$ is unshielded in the premise

Contradiction

$$\frac{C \vee q \diamond 0}{C} \text{ (Ctd)}$$

where

- $q \in \mathbb{Q}$
- $\diamond \in \{>, \geq, \approx, \not\approx\}$
- $\mathbb{Q} \not\models q \diamond 0$

Figure 4: Inference rules of our revised calculus $\text{ALASCA}_{\gtrsim}^{\text{sel}}$ parametrized by an ALASCA-ordering $\prec$ (Def. 2) and an ALASCA-compatible selection function sel (Def. 6 in Sect. 5). Notation $L \diamond_{\text{sel}}^\sigma C$ stands for “Either $\forall L' \in C. L \sigma \diamond L' \sigma$ and $\emptyset = \text{sel}(L \sigma \vee C \sigma)$, or $L \sigma \in \text{sel}(L \sigma \vee C \sigma)$ ”. We consider arbitrary $C, C_i \in \text{Cls}, L \in \text{Lit}, j, k, q \in \mathbb{Q}, s, s', s_i \in \text{Trm} \setminus \mathbf{V}, t, t_i \in \text{Trm}, \diamond, \diamond_i \in \{>, \geq, \approx, \not\approx\}, \gtrsim, \gtrsim_i \in \{>, \geq\}$.

6. $k\underline{s} + t \approx 0 \prec j\underline{s} + u \diamond 0$ where $\diamond \in \{\neq, >, \geq\}$
7. $|k|\underline{s} + t \gtrsim_1 0 \prec -|j|\underline{s} + u \gtrsim_2 0$ where $\gtrsim_i \in \{>, \geq\}$
8. $k\underline{s} + t \not\approx 0 \prec j\underline{s} + u \diamond 0$ where $\diamond \in \{>, \geq\}$
9. $t \geq 0 \prec t > 0$

3.3 The Skeleton Ordering

Our term ordering is inspired by QKBO from [29], an extension of the Knuth-Bendix Ordering (**KBO**) [28]. In contrast to QKBO, our new $\text{ALASCA}_{\prec}^{\text{sel}}$ ordering can not only be parametrized by a KBO, but by an *arbitrary* simplification ordering, (including e.g., LPO [27]).

We define our new ordering as a comparison function cmp that maps two terms to *less*, *greater*, *equivalent*, or *incomparable*. For combining the results of comparison functions, we introduce the operator $\oplus$ defined as (*equivalent* $\oplus y$) = *y*, and $(x \oplus y) = x$ if $x \neq \text{equivalent}$. For any comparison function cmp of a relation $\prec$, we introduce its multiset extension cmp^{mul} which corresponds to the comparison function of the multiset extension $\prec^{\text{mul}}$ wrt equality up to **AC** [37], and cmp^{lex} corresponding to the lexicographical extension respectively. We compare natural numbers $\text{cmp}_{\mathbb{N}}$ by the $<$ relation, and signs $\pm \in \{0, +, -\}$ by the comparison function cmp_{sgn} of the total ordering $0 \prec + \prec -$. Rationals j, k are compared by the comparison function

$$\text{cmp}_{\mathbb{Q}}(j, k) = \text{cmp}_{\text{sgn}}(\text{sgn}(j), \text{sgn}(k)) \oplus \text{cmp}_{\mathbb{N}}(\text{den}(j), \text{den}(k)) \oplus \text{cmp}_{\mathbb{N}}(\text{num}(j), \text{num}(k)),$$

where sgn is the sign, den is the denominator, and num is the enumerator of a number.

Our new $\text{ALASCA}_{\prec}^{\text{sel}}$ ordering (Def. 4 and 5) extends a simplification ordering $\prec$ for uninterpreted terms to terms, and literals containing interpreted symbols. In the first steps, the two terms s, t which should be compared are transformed into terms that do not contain interpreted symbols using the function skel . Then, $\text{skel}(s)$ and $\text{skel}(t)$ are compared using the simplification ordering, and only if that comparison returns equal, we take into account the full terms s, t with their interpreted content.

Definition 3 (Skeleton). *Let $t_i, t, s \in \text{Trm}, k \in \mathbb{Q}$, $\prec$ some preorder, and f an arbitrary uninterpreted function. We call $\text{skel}(t)$ the (uninterpreted) skeleton of t wrt $\prec$:*

$$\begin{aligned} \text{skel}(f(t_1 \dots t_n)) &= f(\text{skel}(t_1) \dots \text{skel}(t_n)) \\ \text{skel}(kt) &= \text{skel}(t) \\ f(\dots \text{undefined} \dots) &= \text{undefined} \end{aligned} \quad \text{skel}(s + t) = \begin{cases} \text{skel}(t) & \text{if } \text{skel}(s) \prec \text{skel}(t) \\ \text{skel}(s) & \text{if } \text{skel}(t) \prec \text{skel}(s) \\ \text{undefined} & \text{else} \end{cases}$$

Definition 4 (Skeleton Term Ordering). *Let $\prec_0$ be some simplification ordering and s, t be terms. We define the skeleton extension $\prec_0^{\text{sk}}$ of $\prec_0$ for terms to be $s \prec_0^{\text{sk}} t$ iff $\text{cmp}_{\text{sk}}(s, t) = \text{less}$. Here:*

$$\text{cmp}_{\text{sk}}(s, t) = \begin{cases} \text{cmp}_{\prec_0}(\text{skel}(s), \text{skel}(t)) \oplus \text{cmp}_{\text{seq}}(s, t) & \text{if } \text{skel}(s) \neq \text{undefined} \neq \text{skel}(t) \\ \text{incomparable} & \text{else} \end{cases}$$

where

$$\begin{aligned}
\text{cmp}_{\text{seq}}(s, s) &= \text{equivalent} && (\text{trm}_1) \\
\text{cmp}_{\text{seq}}(s, t) &= \text{incomparable} && \text{if } \{v \in \text{smnds}(s) \mid v \in \mathbf{V}\} \neq \{v \in \text{smnds}(t) \mid v \in \mathbf{V}\} \quad (\text{trm}_2) \\
\text{cmp}_{\text{seq}}(1, t) &= \text{less} && \text{else } (\text{trm}_3) \\
\text{cmp}_{\text{seq}}(t, 1) &= \text{greater} && \text{else } (\text{trm}_4) \\
\text{cmp}_{\text{seq}}(\underbrace{s_1 + s_2}_s, \underbrace{t_1 + t_2}_t) &= \text{cmp}_{\text{seq}}(s, t_1 + t_2) = \text{cmp}_{\text{sk}}^{\text{mul}}(\text{trms}(s), \text{trms}(t)) && \text{else } (\text{trm}_5) \\
&&& \oplus \text{cmp}_{\text{sk}}^{\text{mul}}(\text{smnds}(s), \text{smnds}(t)) \\
\text{cmp}_{\text{seq}}(js, kt) &= \text{cmp}_{\text{seq}}(s, t) \oplus \text{cmp}_{\mathbb{Q}}(j, k) && \text{else } (\text{trm}_6) \\
\text{cmp}_{\text{seq}}(js, t) &= \text{cmp}_{\text{seq}}(s, t) \oplus \text{greater} && \text{else } (\text{trm}_7) \\
\text{cmp}_{\text{seq}}(s, kt) &= \text{cmp}_{\text{seq}}(s, t) \oplus \text{less} && \text{else } (\text{trm}_8) \\
\text{cmp}_{\text{seq}}(f(s_1 \dots s_n), f(t_1 \dots t_n)) &= \text{cmp}_{\text{seq}}(s_1, t_1) \oplus \dots \oplus \text{cmp}_{\text{seq}}(s_n, t_n) && \text{else } (\text{trm}_9)
\end{aligned}$$

Definition 5 (Literal Extension). *Let $\prec_0$ be an ALASCA term ordering, and $L, M \in \text{Lit}$. We define the literal extension $\prec_0^{\text{lit}}$ of $\prec_0$ for literals to be $L \prec_0^{\text{lit}} M$ iff $\text{cmp}_{\text{lit}}(L, M) = \text{less}$:*

$$\begin{aligned}
\text{cmp}_{\text{lit}}(s \diamond_s 0, t \diamond_t 0) &= \text{incomparable} && \text{if } \text{smnds}(s) \cap \mathbf{V} \neq \text{smnds}(t) \cap \mathbf{V} \quad (\text{lit}_0) \\
\text{cmp}_{\text{lit}}(s \diamond_s 0, t \diamond_t 0) &= \text{cmp}_{\text{at}}^{\text{mul}}(\text{atms}_{\text{lit}}(s \diamond_s 0), \text{atms}_{\text{lit}}(t \diamond_t 0)) && (\text{lit}_1) \\
&\oplus \text{cmp}_{\prec_0}(s, t) && (\text{lit}_2) \\
\text{atms}_{\text{lit}}(l + r \diamond 0) &= \text{atms}_{\text{lit}}(l \diamond 0) \cup \text{atms}_{\text{lit}}(r \diamond 0) \\
\text{atms}_{\text{lit}}(0t \diamond 0) &= \emptyset \\
\text{atms}_{\text{lit}}(kt \sim 0) &= \{\langle t, (\sim) \rangle\} && \text{if } \sim \in \{\approx, \not\approx\} \\
\text{atms}_{\text{lit}}(kt \gtrsim 0) &= \{\langle t, (\gtrsim) \rangle\} && \text{if } k > 0 \ \& \gtrsim \in \{>, \geq\} \\
\text{atms}_{\text{lit}}(kt \lesssim 0) &= \{\langle t, (\lesssim) \rangle\} && \text{if } k < 0 \ \& \lesssim \in \{>, \geq\} \\
\text{atms}_{\text{lit}}(t \diamond 0) &= \{\langle t, (\diamond) \rangle\} \\
\text{cmp}_{\text{at}}(\langle s, l \rangle, \langle t, m \rangle) &= \text{cmp}_{\prec_0}(s, t) \oplus \text{cmp}_{\text{sym}}(l, m) \\
(\approx) \prec_{\text{sym}} (\not\approx) \prec_{\text{sym}} (\geq) \prec_{\text{sym}} (>) \prec_{\text{sym}} (\leq) \prec_{\text{sym}} (<)
\end{aligned}$$

Example 2. *To illustrate Def. 4 and 5 we give some examples of how literals and terms are being oriented using the literal extension of the skeleton extension of a KBO ordering with weight function $w(s) = 1$ for every function symbol and every variable using an arbitrary precedence relation. In what follows f is a function, a is a constant, x, y are variables and $\gtrsim, \gtrsim' \in \{>, \geq\}$.*

$$\begin{aligned}
-f(x) \gtrsim 0 &\succ 5f(x) + a \gtrsim' 0 && 3f(x) \succ f(x) \\
-f(x) \approx 0 &\prec 5f(x) + a \gtrsim 0 && 3f(x) \prec f(x) + 3 \\
f(x) \gtrsim 0 &\succ f(x) + x \approx 0 && 3f(x) + 1 \succ f(x) + 3 \\
f(x) \approx 0 &\prec f(x) + x \approx 0 && x + y \prec x + y + y \\
-f(x) \approx 0 &\prec f(x) + x \approx 0 && x + 2y \prec x + y + y \\
f(x) \not\approx 0 &\succ -f(x) + x \approx 0 && x + y \text{ incomparable } x + a \\
-f(x) > 0 &\succ -7f(x) \geq 0 && f(x) \text{ incomparable } f(y) \\
f(x) > 0 &\succ 7f(x) \geq 0 && f(x) + f(y) \text{ incomparable to any term}
\end{aligned}$$

4 Simplification and Redundancy in ALASCA

Simplification and redundancy elimination are essential techniques for improving reasoning in saturation-based theorem provers. Redundancy elimination is used to delete redundant clauses from the search space, while simplification rules are inferences that replace some clause C in the search space by a set of “simpler” clauses D : by clauses D that make C redundant. The aim of both of these techniques is to keep the search space small and tractable in order to speed up proof search. In this section, we introduce our new inferences rules ALASCA Demodulation (Dmd) and Inequality Merging (IM) (Fig. 5), handling redundancy elimination and simplification in $ALASCA_{\prec}^{sel}$. Dmd is a variant of the demodulation inference rule [33], adjusted for UFLRA. IM is a combination of a simplification rule and a redundancy elimination technique that extends classical tautology deletion to work when reasoning in UFLRA.

Simplification rules must be compatible with the calculus, meaning that newly derived clauses and side premises make the main premise redundant. Speaking more formally we say a simplification rule $\langle C_1 \dots \cancel{C_n}, D \rangle$ is *compatible with $ALASCA_{\prec}^{sel}$* iff for any ground substitution θ it is the case that $\forall C' \in Cs. C'\theta \prec C_n\theta$ and $Cs\theta \models C_n\theta$ where $Cs = \{C_1, \dots, C_{n-1}, D\} \setminus \{\top\}$.

ALASCA Demodulation (Dmd). In the standard superposition calculus, demodulation is an inference rule that rewrites clauses by unit equalities $s \approx t$, such that the rewritten clause becomes redundant [33]. For example, the unit equality $f(x) \approx x$, where $x \in \mathbf{V}$, can be used to rewrite $g(f(a)) > 0$ to $g(a) > 0$. Variants of demodulation are heavily used in proof search, as evidenced by our experiments: when running VAMPIRE without ALASCA enabled on arithmetic benchmarks *forward* demodulation (rewriting newly derived clauses by previously derived unit equalities) is used in 21.16% percent of all proofs, whereas *backward* demodulation (rewriting old clauses with newly derived equalities) is used in 5.39%; notably, standard superposition inferences only occur in 18.55% of all proofs found. Thus proof search in $ALASCA_{\prec}^{sel}$ should clearly benefit from making use of demodulation.

In $ALASCA_{\prec}^{sel}$ unit equalities $s \approx t$ that would usually be used by demodulation, are normalized to $s - t \approx 0$, which in general will prevent the application of classic demodulation. Take

Inequality Merging $\frac{C \vee s \diamond_1 0 \vee s' \diamond_2 0}{C \vee s \diamond_3 0} \text{ (IM)}$ where • $s \equiv_{\mathbf{AC}} (s')^{\pm}$ • $\diamond_2 \in \{>, \geq\} \implies \diamond_1 \in \{>, \geq\}$ • $\diamond_3 = \begin{cases} \diamond_1 \sqcup \lesssim & \text{if } \pm = -, \diamond_2 = \gtrsim \in \{>, \geq\} \\ \diamond_1 \sqcup \diamond_2 & \text{else} \end{cases}$ • $L \sqcup R = set^{-1}(set(L) \cup set(R))$, • $set(\top) = \mathbb{Q}$ • $set(\diamond) = \{q \in \mathbb{Q} \mid q \diamond 0\}$ for $\diamond \in \{>, \geq, <, \leq, \approx, \not\approx\}$ • the conclusion is $\top$ if $\diamond_3 = \top$	ALASCA Demodulation $\frac{k\underline{s} + t \approx 0 \quad C[s\sigma]}{C[-\frac{1}{k}t\sigma]} \text{ (Dmd)}$ where • $(k\underline{s} + t \approx 0)\sigma \prec C[s\sigma]$ • $\forall t_0 \in \mathbf{trms}(t). t_0\sigma \prec s\sigma$ • s is atomic • $s \notin \mathbf{V}$
--	--

Figure 5: Simplification rules of $ALASCA_{\prec}^{sel}$. $\cancel{C}$ denotes C is deleted from the search space.

for example the unit equality $f(x) \approx x$ from above. In our signature this literal will only be present as $f(x) - x \approx 0$, which will block the application of demodulation to rewrite the clause $g(f(a)) > 0$, as none of the subterms can be matched with $f(x) - x$.

As a remedy, we introduce our inference rule ALASCA Demodulation (Dmd) (Fig. 5). Dmd differs from standard demodulation: it uses unit equalities $k\underline{s} + t \approx 0$ to perform rewrites $s \Rightarrow -\frac{1}{k}t$ whenever s is the $\prec$ -maximal summand of $ks + t$, instead of performing rewrites $ks + t \Rightarrow 0$. This also allows not only for rewriting atomic terms, but automatically also rewriting sums of multiple terms. Consider for example $a + b \approx c$, with $a \succ b \succ c$, and the clause $f(a + b) > 0$. The equality will be normalized to $a + b - c \approx 0$ and then Dmd will rewrite $a \Rightarrow c - b$ in $f(a + b) > 0$, deriving $f((c - b) + b) > 0$ which is next normalized to $f(c) > 0$.

To save computation time, Dmd can also be used in a *preordered* mode: replacing the rule's condition $\forall t_0 \in \text{trms}(t). t_0 \sigma \prec s \sigma$ by $\forall t_0 \in \text{trms}(t). t_0 \prec s$. The advantage is that this condition does only depend on the left premise of the rule, thus it can be evaluated once per clause, and does not have to be evaluated for every pair of clauses the rule could be applied to. Nevertheless, this change makes Dmd less applicable. Our experiments allow for both of these Dmd variants (Sect. 6).

Inequality Merging (IM). The other simplification rule we introduce is Inequality Merging (IM), as introduced in Fig. 5. IM is inspired by tautology deletions to delete trivially redundant clauses of the form $C \vee L \vee \neg L$. However, negative inequality literals, like $\neg(t > 0)$, are normalized in ALASCA_0 to positive ones $-t \geq 0$, and hence tautology deletions cannot be applied. To overcome this limitation, IM combines any two literals $t_1 \diamond_1 0$ and $t_2 \diamond_2 0$ that define a bound on the same term $t = t_1 = t_2^\pm$, by merging them into one equivalent literal $t \diamond_3 0$. For combining the inequality symbols $\diamond_1, \diamond_2$ to get $\diamond_3$, we use a *join operation* $\sqcup$ defined by semantic set union of the respective relations.

For example, $-a + b > 0 \vee a - b \approx 0$ is rewritten to $-a + b \geq 0$. This merging of inequalities can detect tautologies, like $-t \geq 0 \vee t \not\approx 0 \Rightarrow \top$, in which case the clause will be deleted from the search space.

In order to turn IM into an actual simplification rule, we need to enforce that $\{t_1 \diamond_1 0, t_2 \diamond_2 0\} \succ^{\text{mul}} \{t_1 \diamond_3 0\}$. This is ensured by ALASCA Ordering (Def. 2). In order to see this for the sake of simplicity consider cases where $t_1 = a$ is a constant. Cases where $\diamond_3 = \top$ do not need to be considered, as the rule will just delete a tautology without introducing a new clause then. Cases where $\diamond_3 \in \{\diamond_1, \diamond_2\}$ obviously hold as these cases amount to a literal deletion. Otherwise, if $\diamond_3 = \neq$, then it must be that $t_2 = \pm a$, thus $\{a > 0, \pm a > 0\} \succ \{a > 0\} \succ^{\text{mul}} \{a \neq 0\}$ by Def. 2.8. Further, if $\diamond_3 = \geq$, then $t_2 = \pm a$, thus $\{a > 0, \pm a \approx 0\} \succ \{a > 0\} \succ^{\text{mul}} \{a \geq 0\}$ by Def. 2.9.

Lemma 1. IM and Dmd are compatible with $\text{ALASCA}_{\prec}^{\text{sel}}$.⁷

5 Literal Selection in ALASCA

Literal selection is a mechanism to guide proof search: it selects subsets of the literals of a clause and restricts the application of inferences to only selected literals. This refines the idea of standard superposition to restrict inferences to only $\prec$ -maximal literals.

Example 3 (Superposition and Maximal Literals). *Consider the superposition inference rule $C \vee s \approx t, D \vee L[s] \vdash C \vee D \vee L[t]$. Take the set of clauses $c \not\approx b \vee d \approx c \vee c \approx a$ and $d \not\approx c$, with term ordering $d \succ c \succ b \succ a$. A side condition of superposition ensures that the rule is applied*

⁷Proof in Appendix D

only when $L[s]$ is maximal wrt $\prec$, that is $L[s] \succ D$. Applied to our clause set, this means we only apply rewriting $d \rightarrow c$, but not rewriting $c \rightarrow a$, in $d \not\approx c$, deriving only the conclusion $c \not\approx b \vee c \approx a \vee c \not\approx c$.

The completeness proof of superposition [2] shows that whenever a clause contains at least one negative literal, we can *select some of these negative literals* and only make inferences with these selected literals, skipping all other inferences involving this clause. Based on this observation any superposition inference that has a side condition of the form “ L is maximal in C ” is then replaced by the condition “either $L \in \text{sel}(C)$, or $\text{sel}(C) = \emptyset$ and L is maximal in C ”.

Example 4 (Superposition and Selection). *In Ex. 3, we can select the negative literal $c \not\approx b$ in the first clause; that is, $\text{sel}(c \not\approx b \vee d \approx c \vee c \approx a) = \{c \not\approx b\}$. No inferences can be performed with the selected literal $c \not\approx b$ and the other clause $d \not\approx c$, thus the set of clauses is saturated without computing any new inferences.*

Selection of negative literals cannot be however easily used in $\text{ALASCA}_{\prec}^{\text{sel}}$, as negative inequalities $\neg(s > 0)$ are normalized to positive ones $-s \geq 0$. As such, selecting arbitrary negative literals in $\text{ALASCA}_{\prec}^{\text{sel}}$ would not work with inequality literals. Our work therefore introduces the concept of *selectable* literals (Def. 6), allowing to extend selection functions also to non-negative literals.

Selection in Superposition. We first explain selection principles in superposition, after which we detail which inequality literals are selectable in $\text{ALASCA}_{\prec}^{\text{sel}}$. We illustrate this by outlining the refutational completeness proof of superposition with selection [2]. Completeness of (ground) superposition is usually shown by first constructing a model functor I_N , which is a function that maps sets of ground clauses N to candidate models I_N . Next, superposition is proven to be a counterexample-reducing inference system wrt. I_N . This means that, for every $\prec$ -minimal $C \in N$ such that $I_N \not\models C$, there is some superposition inference $\iota = \langle C_1, \dots, C_n, C, D \rangle$, with $D \prec C$ and $\forall i. C_i \prec C$, such that $I_N \not\models D$; we say that ι reduces the counterexample C to D . Any inference system that satisfies this property for a well-founded ordering $\prec$ is *refutationally complete* [4].

Consider now some clause C that contains a negative equality $C = C' \vee s \not\approx t$. If $I_N \not\models C$, then $I_N \not\models s \not\approx t$, which is the case iff $I_N(s) = I_N(t)$. A candidate model I_N for superposition is constructed to be a convergent rewrite system. This means that either $s = t$, or there must be some $(u \rightarrow v) \in I_N$ such that s or t can be rewritten by $u \rightarrow v$. In the former case, C can be reduced to C' , while in the latter case, by construction of I_N , there must be some clause $C_2 = C'_2 \vee u \approx v$ such that C can be reduced by a superposition inference to $C' \vee C'_2 \vee (s \not\approx t)[u/v]$. Thus, for any such negative literal $s \not\approx t$, the clause can be reduced by some inference rule, thus we can select this negative literal to be the only one used within an inference. The same argument is however not applicable for positive literals $s \approx t$: they might not be reducible by any inference when they are so-called *productive* literals, which are always true ($I_N \models s \approx t$) by construction of I_N and thus cannot be a counterexample.

Selection in $\text{ALASCA}_{\prec}^{\text{sel}}$. Even though no completeness proof for $\text{ALASCA}_{\prec}^{\text{sel}}$ is given in this paper, we motivate the definition of selectable literals by investigating the completeness proof of ALASCA_0 from [29]. In ALASCA_0 equalities are treated almost the same as in superposition, thus we only pay attention to inequalities here. Inequality literals with a maximal negative term $-k\underline{s} + t > 0$ can never be productive. Consider some clause $C = C' \vee -k\underline{s} + t > 0$. If $I_N \not\models C$, then one of the following cases occurs: ; (i) if s is strictly maximal and $I_N(s) \neq s$, then

s can be reduced by superposition, NF, or TF; (ii) if s is strictly maximal and $I_N(s) = s$, then $I_N \models ks - t \geq 0$, which in turn means that there is some $s > l \in I_N$ such that $I_N \models kl - t \geq 0$, and $C_2 = +js - jl \gtrsim 0 \vee C'_2 \in N$. In case (ii), we use FM, which reduces C to $C' \vee -k(jl) + jt > 0$ using C_2 . In summary, if we select some inequality literal with a negative maximal term, we do not need to perform any inferences with any other literal. The same idea applies to non-strict inequalities $\geq$, and disequalities $\neq$ can be selected for the same reason as in superposition.

Non-Ground Selection in $\text{ALASCA}_{\text{sel}}^{\text{sel}}$. As selection needs to be lifted to non-ground clauses, the following two issues need to be considered.

First, there might be multiple maximal terms in an ALASCA sum. We can only select non-ground literals where every ground instance is selectable. Consider for example the literal $-f(x) + g(y) > 0$. Such a literal cannot be selected since, depending on the choice of x and y , either the first or the second summand could be maximal, thus in some instances the maximal term has a negative summand, while in other instances the maximal term has a positive sign.

The second issue comes from the treatment of top-level variables in ALASCA. As inferences with variables as maximal terms cannot be lifted easily, they will be processed by VE during proof search. Since inferences with these variables cannot be lifted directly, we cannot select literals with maximal variables either.

Both of these conditions are fulfilled by stably negative literals Lit^- . We thus introduce selectable literals and ALASCA-compatible selection functions to be used in Fig. 4.

Definition 6 (Selectable Literal). *A literal L is selectable iff it is stably negative $L \in \text{Lit}^-$. A selection function sel is ALASCA-compatible if, for any clause C , every $L \in \text{sel}(C)$ is selectable.*

Selection Strategies. The notion of selectable literals does not yield a unique selection function. Thus as there is no single best selection strategy for how many and/or which selectable literals should be selected, we implemented the following selection mechanisms:

- *total*: selecting all literals (including both selectable and non-selectable ones);
- *maximal*: selecting no literals, thus always using maximal literals only for rule applications;
- *random*: choosing a random selectable literal;
- *best*: choosing the “best” selectable literal, where best means it is maximal wrt some measure $\ll$, as defined in Def. 7;
- *lookahead*: choosing the selectable literal which unifies with the least number of terms that are currently present in the search space, adjusting results of [25];
- *incomplete*: for every strategy which chooses from the set of selectable literals, there is also an incomplete version that chooses from the set of all literals of the clause instead.

Def. 7 gives the ordering $\ll$ for the selection strategy *best*. Using $\ll$, we select literals to yield as few rule applications as possible, but, in contrast to *lookahead* selection, without looking at the entire search space. The measure $\ll$ of the selection strategy *best* is defined lexicographically in Def. 7, by comparing tuples of multiple values and using the following functions:

- **prob**: is a measure that reflects how likely it is for a term to unify with a ground term, assuming that the probability distribution of terms is $P(X = f(t_1 \dots t_n)) = \frac{1}{|\mathbb{F}|} (P(X = t_1) \dots P(X = t_n))$. As a greater **prob** will potentially yield more unifiers, we prefer terms that minimize **prob**.

- cnt_+ : is the number of $+$ symbols occurring in a literal. Due to **AC**, more occurrences of $+$ will potentially yield more unifiers, thus we prefer terms that minimize cnt_+ .
- negEq : is 1, if the literal is a negative equality, and 0 otherwise. The idea of this measure is that negative equalities are mostly introduced by UWA, and thus represent unifiability constraints. It is unclear whether terms with negEq being 0 or 1 should be preferred in general. Thus our portfolio approach (Sect. 6) uses both these options.

Finally, in every literal $t \diamond 0$, only the terms $\text{trms}(t)$ are used to compute inferences, and the rest of the terms of $\text{smnds}(t)$ are used only after the maximal ones have been resolved. We thus first compute the measures $m \in \{\text{cnt}_+, \text{prob}\}$ for the maximal terms m^{max} , and then for all terms m^{all} .

Definition 7 (Literal Selection Measure). *Our measure $\ll$ used within the (best) literal selection mechanism is parametrized by the option $\pm$ and defined below:*

$$\begin{aligned}
 L_1 \ll L_2 &\iff \text{measure}(L_1) <^{lex} \text{measure}(L_2) \\
 \text{measure}(L) &= \langle \pm \text{negEq}(L), -\text{prob}^{\text{max}}(L), -\text{prob}^{\text{all}}(L), -\text{cnt}_+^{\text{max}}(L), -\text{cnt}_+^{\text{all}}(L) \rangle \\
 \text{cnt}_+(t_1 + \dots + t_n) &= 1 + \sum_{i=1}^n \text{cnt}_+(t_i) & \text{prob}(t) &= \left(\frac{1}{|\mathbf{F}|}\right)^{\text{sz}(t) - |\{v \in \mathbf{V} \mid v \triangleleft t\}|} \\
 \text{cnt}_+(f(t_1 \dots t_n)) &= \sum_{i=1}^n \text{cnt}_+(t_i) & \text{negEq}(t \diamond 0) &= \begin{cases} 1 & \text{if } (\diamond) = (\neq) \\ 0 & \text{else} \end{cases} \\
 \text{cnt}_+(x) &= 0 \\
 m^{\text{max}}(t \diamond 0) &= \sum \{m(t) \mid t \in \text{trms}(t)\} & \text{sz}(f(t_1 \dots t_n)) &= 1 + \sum_{i=1}^n \text{sz}(t_i) \\
 m^{\text{all}}(t \diamond 0) &= \sum \{m(t) \mid t \in \text{trms}(t)\} & \text{sz}(t_1 + t_2) &= 1 + \text{sz}(t_1) + \text{sz}(t_2) \\
 & & \text{sz}(v) &= 1
 \end{aligned}$$

Delayed VE. The rule Variable Elimination (VE) removes literals $kx + t \diamond 0$, since inferences involving unshielded variables x cannot be lifted easily. If a clause contains a selected literal, though there are no inferences involving x , that would need to be lifted. Thus the rule VE needs to only be applied if there are no selected literals in a clause, and can be skipped if there are selected ones. We call this skipping of VE inferences using *delayed* variable elimination, and implemented an option to enable or disable this behaviour.

6 Implementation and Experiments

Implementation. We implemented⁸ $\text{ALASCA}_{\prec}^{\text{sel}}$ with all additions described in the paper in the theorem prover VAMPIRE [7]. Our implementation handles a vast option space of various literal selection strategies (Sect. 5), enabling/disabling IM (Sect. 4), Dmd (Sect. 4), delayed variable elimination (Sect. 5), and running Dmd in preordered mode.

To accommodate the many options and strategies in $\text{ALASCA}_{\prec}^{\text{sel}}$, we created a portfolio of randomized option combinations. For this we chose a randomized subset B of 50% of our full benchmark set. Then, the following was repeated: (i) select a timeout within $[1, 10]$ seconds; (ii) generate 80 random option combinations; (iii) run options on the current set B . We chose the strategy that solved most problems of B and added it to the portfolio, after which we removed all solved problems from B . We repeated this process for 12 hours. In our experiments, $\text{ALASCA}_{\prec}^{\text{sel}}$ refers to the resulting portfolio⁹ setting.

⁸ Available at <https://github.com/vprover/vampire/tree/alasca-lpar-2026>.

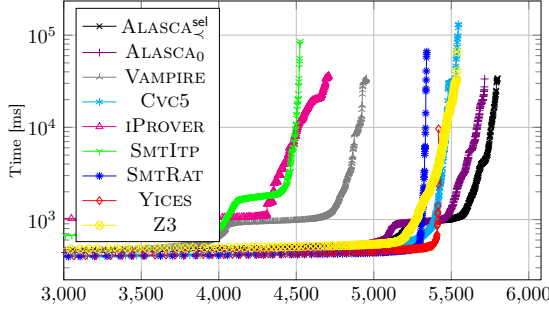


Figure 6: The cumulative number of benchmarks solved by each solver.

Benchmarks	ALASCA ^{sel}	CVC5	Z3	YICES	SMT-RAT	VAMPIRE	iPROVER	SMTITP
all	42	30	2	9	-	-	1	1
LRA	-	2	1	9	-	-	-	-
NRA	-	-	-	-	-	-	-	-
UFLRA	-	-	-	-	-	-	-	-
TRIANGULAR	13	-	-	-	-	-	-	-
LIMIT	24	-	-	-	-	-	-	-
SH	5	28	1	-	-	-	1	1

Figure 7: Number of problems solved uniquely by each solver.

Benchmarks (#)	ALASCA ^{sel}	ALASCA ₀	CVC5	Z3	YICES	SMT-RAT	VAMPIRE	iPROVER	SMTITP	solved
all (6291)	5799 (19)	5715 (1)	5547 (29)	5537 (2)	5418 (8)	5342 (0)	4953 (0)	4709 (1)	4525 (1)	5882
LRA (1629)	1594 (0)	1571 (0)	1346 (2)	1412 (1)	1611 (8)	1530 (0)	856 (0)	624 (0)	619 (0)	1616
NRA (3814)	3801 (0)	3797 (0)	3803 (0)	3812 (0)	3807 (0)	3812 (0)	3802 (0)	3782 (0)	3718 (0)	3812
UFLRA (13)	10 (0)	10 (0)	10 (0)	10 (0)	0 (0)	0 (0)	10 (0)	10 (0)	10 (0)	10
TRIANGULAR (34)	25 (0)	24 (0)	7 (0)	1 (0)	0 (0)	0 (0)	8 (0)	5 (0)	0 (0)	25
LIMIT (287)	116 (14)	102 (0)	91 (0)	81 (0)	0 (0)	0 (0)	82 (0)	82 (0)	0 (0)	116
SH (514)	253 (5)	211 (1)	290 (27)	221 (1)	0 (0)	0 (0)	195 (0)	206 (1)	178 (1)	303

Figure 8: Number of problems solved by each solver in each problem group. Column 1 lists the names and the number of problems of each group. In columns 2–8, a cell “ s (u)” for a solver and a benchmark group denotes that the solver solved s benchmarks, of which u benchmarks were uniquely solved by that solver.

Benchmarks. We use the state-of-the-art benchmark set of [29]⁹. We only consider unsatisfiable problems, as these are the only relevant ones for ALASCA^{sel}. Our benchmark set is split into multiple groups, as summarized in Fig. 8 and detailed next: NRA (non-linear real arithmetic), LRA and UFLRA represent the respective SMT-LIB logics [6]; TRIANGULAR and LIMIT contain mathematical properties from [29]; SH contains Sledgehammer proving problems from [15].

Solvers. We compare ALASCA^{sel} against all solvers that entered the **Arith** division of the SMT-COMP 2025 system competition¹⁰. We also compare ALASCA^{sel} to the original calculus ALASCA₀ from [29], and VAMPIRE without ALASCA enabled.

Benchmarking Setup. Our experiments were conducted on an AMD EPYC 7252 8-Core Processor with 16 cores and 2 threads per core and a RAM of 125Gi. Benchmarks were run for 120 seconds on 4 cores in parallel.

Results. Experimental results are summarized in Fig. 6–Fig. 7. As evidenced in Fig. 6, our implementation ALASCA^{sel} vastly outperforms all other solvers: already after about 1 second, ALASCA^{sel} solves substantially more benchmarks than all other solvers at that time. Looking at the raw data reveals that ALASCA^{sel} solves 5716 problems within only 4330 milliseconds, which is more than any other prover solves in the full timeout of 120 seconds. Fig. 8 shows

⁹Available at https://github.com/vprover/vampire_benchmarks/tree/master/alasca-lpar-2026.

¹⁰See <https://smt-comp.github.io/2025>

that $\text{ALASCA}_{\succ}^{\text{sel}}$ outperforms ALASCA_0 by 84 (1.46 %) more solved problems and dominates ALASCA_0 in every problem category. $\text{ALASCA}_{\succ}^{\text{sel}}$ also outperforms VAMPIRE in every problem category. Fig. 7 lists the number of uniquely solved problems by $\text{ALASCA}_{\succ}^{\text{sel}}$, demonstrating ALASCA-based reasoning solves problems no other solver is able to prove.

7 Related Work

The most prominent approaches for (restrictions of) UFLRA are SMT solving [14, 5, 10] and first-order theorem proving [19, 3, 45, 32, 29].

SMT-based reasoning for UFLRA. SMT solvers naturally support reasoning in quantifier-free UFLRA using dedicated decision procedures in linear real arithmetic. As quantified UFLRA is undecidable [32], incomplete quantifier instantiations are used to instantiate universally quantified variables with pragmatically chosen terms, trying to reduce the problem of solving quantified UFLRA problems to quantifier-free UFLRA instances. Various quantifier instantiation heuristics emerged, notably E-matching [16] and model-based quantifier instantiation [21]. Further, [42, 26] introduce strategies for the order of enumerating all potentially useful instances. These approaches are restricted to heuristic-driven instantiations, limiting their applicability to quantified UFLRA.

When only considering quantified linear real arithmetic (LRA), quantifier elimination methods via Fourier–Motzkin elimination [20, 35], conflict resolution [31], Cylindrical Algebraic Decomposition (CAD) [1, 11], and virtual substitutions [44] have emerged to decide LRA. The work of [30] combines virtual substitutions with Cooper’s method [12] to create a quantifier elimination procedure for linear mixed integer-real arithmetic, and hence LRA as well. A related, model-driven approach for solving LRA problems using quantifier elimination is introduced in [9]. Recently, the approach of [24] introduces the first singly-exponential quantifier elimination for linear integer arithmetic, improving doubly-exponential complexities known before; yet, practical use of this computational improvement is not yet evidenced. ALASCA [29] implements a form of quantifier elimination via its Variable Elimination (VE) rule (Fig. 4). Our enhancements upon ALASCA inherit quantifier elimination for LRA, yet extends them with dedicated selection and simplification rules for quantified UFLRA. As such and in contrast to the aforementioned techniques, our work enables efficient reasoning with quantifiers, uninterpreted functions and linear real arithmetic (see Sect. 6).

First-order theorem proving for UFLRA. Saturation-based theorem proving using the superposition calculus [37, 2, 46] is the leading and refutationally complete technology for automating reasoning in first-order logic with equality. Superposition proving is however notoriously bad when faced with interpreted symbols, that is functions or predicates with a fixed-semantic in a background theory, such as transitive relations or associative and commutative (AC) functions. This is due to the fact that the naive approach of adding AC or transitivity axioms to the input problem explicitly leads to an intractable blow-up of the search space [17]. In order to circumvent this, various approaches have been introduced: In [19] the concept of ground joinability was proposed to skip inferences involving AC axioms. Using the AVATAR architecture [38] and theory instantiation [41], SMT solving can be used within saturation in order to reason about subsets of formulas from the search space. SMT solving helps to detect AC functions in input problems [18], enabling specialized reasoning techniques like ground joinability for these functions. Various incomplete simplification rules [39] and guiding proof search

by clause selection [23] have also been shown to greatly improve practical performance for interpreted problems in first-order theories. Unlike these methods, our work supports arithmetic reasoning via dedicated inference rules in superposition.

By improving the underlining inference systems of saturation-based provers, the chaining calculi [3] was proposed for arbitrary transitive relations that bypass the need for a transitivity axiom. Chaining with **AC** reasoning is combined in [45], creating a calculus for totally ordered divisible Abelian groups. Going further, a ground extension of the superposition calculus was introduced in [32] for UFLRA. Fusing ideas of [3, 45, 32], the ALASCA calculus has been proposed in [29] in combination with unification with abstraction (UWA) [8]. Our work enhances [29] by improving selection, simplification and redundancy elimination during proof search, improving automated reasoning in quantified UFLRA (see Sect. 6).

8 Conclusions

We introduced the calculus $\text{ALASCA}_{\prec}^{\text{sel}}$ enhancing ALASCA_0 with Literal Selection and the new simplification rules ALASCA Demodulation and Inequality Merging. We implemented $\text{ALASCA}_{\prec}^{\text{sel}}$ in a standard saturation loop and demonstrated experimentally that our work outperforms the state-of-the-art. We aim to further improve literal selection by finding an accurate probability of a term unifying with other terms modulo **AC** which could be a promising heuristic for literal selection. We are also interested in instantiations and generalizations modulo **AC**, increasing applicability of demodulation.

Acknowledgements. This research was funded in whole or in part by the Austrian Science Fund – FWF Grant ID 10.55776/EFP2661725 and SpyCoDe Grant 10.55776/F85, the ERC Proof of Concept Grant LEARN 101213411, the TU Wien Doctoral College SecInt, and the WWTF grant [ForSmart Grant ID: 10.47379/ICT22007].

References

- [1] Dennis S. Arnon, George E. Collins, and Scott McCallum. Cylindrical algebraic decomposition I: the basic algorithm. *SIAM J. Comput.*, 13(4):865–877, 1984.
- [2] Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *J. Log. Comput.*, 4(3):217–247, 1994.
- [3] Leo Bachmair and Harald Ganzinger. Ordered chaining calculi for first-order theories of transitive relations. *J. ACM*, 45(6):1007–1049, 1998.
- [4] Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In *Handbook of Automated Reasoning*, volume 1, pages 19–99. Elsevier and MIT Press, 2001.
- [5] Haniel Barbosa, Clark W. Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. cvc5: A versatile and industrial-strength SMT solver. In *TACAS*, volume 13243 of *LNCS*, pages 415–442. Springer, 2022.
- [6] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The smt-lib standard: Version 2.7. Technical report, Department of Computer Science, The University of Iowa, 2025.
- [7] Filip Bárték, Ahmed Bhayat, Robin Couteil, Márton Hajdú, Matthias Hetzenberger, Petra Hozová, Laura Kovács, Jakob Rath, Michael Rawson, Giles Reger, Martin Suda, Johannes Schoisswohl, and Andrei Voronkov. The vampire diary. In *CAV*, volume 15933 of *LNCS*, pages 57–71. Springer, 2025.
- [8] Ahmed Bhayat, Konstantin Korovin, Laura Kovács, and Johannes Schoisswohl. Refining unification with abstraction. In *LPAR*, volume 94 of *EPiC*, pages 36–47. EasyChair, 2023.
- [9] Nikolaj S. Bjørner and Mikoláš Janota. Playing with quantified satisfaction. In *LPAR*, volume 35 of *EPiC*, pages 15–27. EasyChair, 2015.
- [10] Nikolaj S. Bjørner and Lev Nachmanson. Arithmetic solving in Z3. In *CAV*, volume 14681 of *LNCS*, pages 26–41. Springer, 2024.
- [11] George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition: a synopsis. *SIGSAM Bull.*, 10(1):10–12, 1976.
- [12] David C Cooper. Theorem proving in arithmetic without multiplication. *Machine intelligence*, 7(91-99):300, 1972.
- [13] Simon Cruanes. *Extending Superposition with Integer Arithmetic, Structural Induction, and Beyond. (Extensions de la Superposition pour l’Arithmétique Linéaire Entière, l’Induction Structurale, et bien plus encore)*. PhD thesis, École Polytechnique, Palaiseau, France, 2015.
- [14] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Z3: an efficient SMT solver. In *TACAS*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.
- [15] Martin Desharnais, Petar Vukmirovic, Jasmin Blanchette, and Makarius Wenzel. Seventeen provers under the hammer. In *ITP*, volume 237 of *LIPIcs*, pages 8:1–8:18, 2022.
- [16] David Detlefs, Greg Nelson, and James B. Saxe. Simplify: a theorem prover for program checking. *J. ACM*, 52(3):365–473, 2005.
- [17] Eric Dmenjoud. A technical note on ac-unification. the number of minimal unifiers of the equation $\alpha x_1 + \dots + \alpha x_i \dot{=}_{AC} \beta y_1 + \dots + \beta y_q$. *J. Autom. Reason.*, 8(1):39–44, 1992.
- [18] André Duarte and Konstantin Korovin. Implementing superposition in iprover (system description). In *IJCAR*, volume 12167 of *LNCS*, pages 388–397. Springer, 2020.
- [19] André Duarte and Konstantin Korovin. Ground joinability and connectedness in the superposition calculus. In *IJCAR*, volume 13385 of *LNCS*, pages 169–187. Springer, 2022.
- [20] Jean Baptiste Joseph Fourier. Solution d’une question particuliere du calcul des inégalités. *Nouveau Bulletin des Sciences par la Société philomatique de Paris*, 99:100, 1826.
- [21] Yeteng Ge and Leonardo Mendonça de Moura. Complete instantiation for quantified formulas in

- satisfiability modulo theories. In CAV, volume 5643 of LNCS, pages 306–320. Springer, 2009.
- [22] Bernhard Gleiss, Laura Kovács, and Jakob Rath. Subsumption demodulation in first-order theorem proving. In IJCAR, volume 12166 of LNCS, pages 297–315. Springer, 2020.
 - [23] Bernhard Gleiss and Martin Suda. Layered clause selection for theory reasoning. CoRR, abs/2001.09705, 2020.
 - [24] Christoph Haase, Shankara Narayanan Krishna, Khushraj Madnani, Om Swostik Mishra, and Georg Zetsche. An efficient quantifier elimination procedure for presburger arithmetic. In ICALP, volume 297 of LIPIcs, pages 142:1–142:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.
 - [25] Krystof Hoder, Giles Reger, Martin Suda, and Andrei Voronkov. Selecting the selection. In IJCAR, volume 9706 of LNCS, pages 313–329. Springer, 2016.
 - [26] Mikolás Janota, Haniel Barbosa, Pascal Fontaine, and Andrew Reynolds. Fair and adventurous enumeration of quantifier instantiations. In FMCAD, pages 256–260. IEEE, 2021.
 - [27] Sam Kamin and Jean-Jacques Lévy. Two generalizations of the recursive path ordering. Unpublished manuscript, 1980.
 - [28] Donald E Knuth and Peter B Bendix. Simple word problems in universal algebras. Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970, pages 342–376, 1983.
 - [29] Konstantin Korovin, Laura Kovács, Giles Reger, Johannes Schoisswohl, and Andrei Voronkov. ALASCA: reasoning in quantified linear arithmetic. In TACAS, volume 13993 of LNCS, pages 647–665. Springer, 2023.
 - [30] Konstantin Korovin, Laura Kovács, and Johannes Schoisswohl. VIRAS: conflict-driven quantifier elimination for integer-real arithmetic. In LPAR, volume 100 of EPiC, pages 147–164. EasyChair, 2024.
 - [31] Konstantin Korovin, Nestan Tsiskaridze, and Andrei Voronkov. Conflict resolution. In CP, volume 5732 of LNCS, pages 509–523. Springer, 2009.
 - [32] Konstantin Korovin and Andrei Voronkov. Integrating linear arithmetic into superposition calculus. In CSL, volume 4646 of LNCS, pages 223–237. Springer, 2007.
 - [33] Laura Kovács and Andrei Voronkov. First-order theorem proving and vampire. In CAV, volume 8044 of LNCS, pages 1–35. Springer, 2013.
 - [34] Amira T Mahmoud, Ahmad A Mohammed, Mahitap Ayman, Walaa Medhat, Sahar Selim, Hala Zayed, Ahmed H Yousef, and Nahla Elaraby. Formal verification of code conversion: A comprehensive survey. Technologies, 12(12):244, 2024.
 - [35] Theodore Samuel Motzkin. Beiträge zur Theorie der linearen Ungleichungen. Azriel, 1936.
 - [36] M. Saqib Nawaz, Moin Malik, Yi Li, Meng Sun, and Muhammad Ikram Ullah Lali. A survey on theorem provers in formal methods. CoRR, abs/1912.03028, 2019.
 - [37] Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In Handbook of Automated Reasoning, volume 1, pages 371–443. Elsevier and MIT Press, 2001.
 - [38] Giles Reger, Nikolaj S. Bjørner, Martin Suda, and Andrei Voronkov. AVATAR modulo theories. In GCAI, volume 41 of EPiC, pages 39–52. EasyChair, 2016.
 - [39] Giles Reger, Johannes Schoisswohl, and Andrei Voronkov. Making theory reasoning simpler. In TACAS, volume 12652 of LNCS, pages 164–180. Springer, 2021.
 - [40] Giles Reger and Martin Suda. Set of support for theory reasoning. In IWIL, volume 1 of Kalpa Publications in Computing, pages 124–134. EasyChair, 2017.
 - [41] Giles Reger, Martin Suda, and Andrei Voronkov. Unification with abstraction and theory instantiation in saturation-based reasoning. In TACAS, volume 10805 of LNCS, pages 3–22. Springer, 2018.
 - [42] Andrew Reynolds, Haniel Barbosa, and Pascal Fontaine. Revisiting enumerative instantiation. In TACAS, volume 10806 of LNCS, pages 112–131. Springer, 2018.
 - [43] Stephan Schulz, Simon Cruanes, and Petar Vukmirovic. Faster, higher, stronger: E 2.3. In CADE,

- volume 11716 of LNCS, pages 495–507. Springer, 2019.
- [44] Thomas Sturm. Thirty years of virtual substitution: Foundations, techniques, applications. In ISSAC, pages 11–16. ACM, 2018.
 - [45] Uwe Waldmann. Superposition and chaining for totally ordered divisible abelian groups. In IJCAR, volume 2083 of LNCS, pages 226–241. Springer, 2001.
 - [46] Uwe Waldmann, Sophie Turret, Simon Robillard, and Jasmin Blanchette. A comprehensive framework for saturation theorem proving. J. Autom. Reason., 66(4):499–539, 2022.
 - [47] Christoph Weidenbach, Dilyana Dimova, Arnaud Fietzke, Rohit Kumar, Martin Suda, and Patrick Wischniewski. SPASS version 3.5. In Renate A. Schmidt, editor, CADE, volume 5663 of LNCS, pages 140–145. Springer, 2009.

A Comparison to original ALASCA

For the most part the revised $\text{ALASCA}^{\text{sel}}$ matches the original ALASCA_0 from [29]. The main conceptual change in our updated calculus is the following: In ALASCA_0 reasoning modulo $\mathcal{A}_{\text{eq}}$ (see Fig. 3) is implicitly performed using UWA. This requires the ordering to be $\mathcal{A}_{\text{eq}}$ -compatible, which leads to breaking the subterm property, which we would like to have in our calculus. We will describe this issue in more details in Sect. A.1. To resolve this instead of using UWA for $\mathcal{A}_{\text{eq}}$, in our updated calculus $\text{ALASCA}^{\text{sel}}$ we use UWA for the subset **AC** only and introduce additional simplifications to cover the rest of the axioms in $\mathcal{A}_{\text{eq}} \setminus \mathbf{AC}$ (Sect. A.2). This conceptual change requires us to introduce (i) a different approach to normalization (Sect. A.2.1), which in turn requires generalizing the rule Term Factoring (TF) and adding a new generating inference rule Abstraction (Abs); (ii) a new term and literal ordering that is **AC**-compatible, and possesses the subterm property modulo **AC** (Sect. C); and (iii) a new approach to unification with abstraction (UWA) so that it works modulo **AC** (Sect. B).

Before explaining the issues around ALASCA_0 's ordering in detail, let us go through each of the rules of $\text{ALASCA}^{\text{sel}}$ (repeated on Page 24) and explain how they relate to ALASCA_0 . If not stated otherwise the rule is the same in both calculi.

- **Fourier-Motzkin (FM)** joins inequalities to simulate transitivity. For example it can use $-a + b > 0$ and $a + c \geq 0$ to infer $b + c > 0$.
- **Tight Fourier-Motzkin (TFM)** is a version of FM for non-strict inequalities, that allows for deriving necessary equalities from them. It can use $-a \geq 0$ and $a \geq 0$ to infer $-a \approx 0 \vee 0 > 0$.
- **Superposition (Sup)** rewrites clauses using equality literals, similarly to our simplification rule Dmd (Sect. 4) rule (Sect. 4), but generating additional clauses instead of simplifying existing ones. With this rule we can derive $f(x, y) - g(y, x) \approx 0, g(a, x) > 0 \vdash f(x, a) > 0$. This rule is essentially the same as in ALASCA_0 , but using the notation $u \in \text{lead}(\dots)$, which more concisely represents multiple side conditions and also makes sure that only selected literals are rewritten if any are selected. Further the rule does not use the notation $\approx$ from ALASCA_0 , which is not needed in this paper as we do not consider additional uninterpreted sorts.
- **Inequality Factoring (IF)** combines multiple inequality literals in one clause in such a way that they can properly be used with FM and TFM. For example, we use $f(a) + a > 0 \vee f(a) + b > 0$ to derive both $f(a) + b > 0 \vee a - b > 0$ and $f(a) + a > 0 \vee b - a > 0$.
- **Equality Factoring (EF)** is similar to IF, but instead of inequalities, it combines multiple equality literals such that they can properly be used by Sup. It derives, for example, $2f(a) + a \approx 0 \vee 3f(a) + b \approx 0 \vdash 3a \approx 2b \vee 2f(a) + a \approx 0$. This rule is essentially the same as in ALASCA_0 , but without using the notation $\approx$, which is not needed in this paper as we do not consider additional uninterpreted sorts.
- **Contradiction (Ctd)** is used to remove trivially unsatisfiable literals, deriving $a > 0 \vee -5 > 0 \vdash a > 0$.
- **Normalization (NF)** performs one step of simplifying a literal to a normal form, like $\frac{1}{2}(6a + 2f(y)) \approx b + a + f(x) \vdash \dots \vdash 2a + b + f(x) + f(y) \approx 0$. This rule does not exist in ALASCA_0 as in that calculus normalization is performed implicitly by unifying modulo $\mathcal{A}_{\text{eq}}$.
- **Term Factoring (TF)** add up unifiable summands in a clause, deriving for example $g(2f(x) + 3f(y)) > 0 \vdash g(5f(x)) > 0$. This rule is a generalization of the corresponding rule in ALASCA_0 .

Fourier-Motzkin

$$\frac{C_1 \vee j \underline{s} + t_1 \gtrsim_1 0 \quad C_2 \vee -k \underline{s}' + t_2 \gtrsim_2 0}{(C_1 \vee C_2 \vee kt_1 + jt_2 > 0) \sigma \vee \mathcal{E}} \text{ (FM)}$$

where

- $\langle \sigma, \mathcal{E} \rangle = \text{UWA}(s, s')$
- $js + t_1 \gtrsim_1 0 \not\vdash_{\text{sel}}^\sigma C_1$
- $-ks' + t_2 \gtrsim_2 0 \not\vdash_{\text{sel}}^\sigma C_2$
- $\{>\} \subseteq \{\gtrsim_1, \gtrsim_2\} \subseteq \{>, \geq\}$
- $j, k > 0$

Superposition

$$\frac{C_1 \vee k \underline{s} + t \approx 0 \quad C_2 \vee L[s']}{(C_1 \vee C_2 \vee L[-\frac{1}{k}t]) \sigma \vee \mathcal{E}} \text{ (Sup)}$$

where

- $\langle \sigma, \mathcal{E} \rangle = \text{UWA}(s, s')$
- $k \underline{s} + t \approx 0 \not\vdash_{\text{sel}}^\sigma C_1$
- $(s')\sigma \triangleleft u \in \text{lead}((L[s'] \vee C_2)\sigma)$
- $(k \underline{s} + t \approx 0)\sigma \not\vdash L[s']\sigma$
- s is atomic

Equality Factoring

$$\frac{C \vee j \underline{s} + t_1 \approx 0 \vee k \underline{s}' + t_2 \approx 0}{(C \vee kt_1 - jt_2 \approx 0 \vee js + t_1 \approx 0) \sigma \vee \mathcal{E}} \text{ (EF)}$$

where

- $\langle \sigma, \mathcal{E} \rangle = \text{UWA}(s, s')$
- $ks' + t_2 \approx 0 \not\vdash_{\text{sel}}^\sigma C \vee js + t_1 \approx 0$

Normalization

$$\frac{C \vee L[t_1]}{C \vee L[t_2]} \text{ (NF)}$$

where

- $t_1 \triangleleft u \in \text{lead}(L[t_1] \vee C)$
- $t_1 \approx t_2 \in \left\{ \begin{array}{l} 0u \approx 0, \quad 0 \cdot 1 + u \approx u, \\ j(ku) \approx (jk)u, \quad 1(u) \approx u, \\ k(u+v) \approx ku + kv, \\ j \cdot u + k \cdot u + v \approx (j+k)u + v, \end{array} \middle| \begin{array}{l} u, v \in \text{Trm} \\ j, k \in \mathbb{Q} \end{array} \right\}$

Abstraction

$$\frac{C[k \underline{x} + t]}{C[y] \vee kx + t \not\approx y} \text{ (Abs)}$$

where

- $x \in \mathbf{V}$
- y is a fresh variable
- $C[o]$ is an uninterpreted clause context

Tight Fourier-Motzkin

$$\frac{C_1 \vee j \underline{s} + t_1 \geq 0 \quad C_2 \vee -k \underline{s}' + t_2 \geq 0}{(C_1 \vee C_2 \vee kt_1 + jt_2 > 0 \vee -ks' + t_2 \approx 0) \sigma \vee \mathcal{E}} \text{ (TFM)}$$

where

- $\langle \sigma, \mathcal{E} \rangle = \text{UWA}(s, s')$
- $js + t_1 \geq 0 \not\vdash_{\text{sel}}^\sigma C_1$
- $-ks' + t_2 \geq 0 \not\vdash_{\text{sel}}^\sigma C_2$
- $j, k > 0$

Inequality Factoring

$$\frac{C \vee j \underline{s} + t_1 \gtrsim_1 0 \vee k \underline{s}' + t_2 \gtrsim_2 0}{(C \vee kt_1 - jt_2 \gtrsim_3 0 \vee k \underline{s}' + t_2 \gtrsim_2 0) \sigma \vee \mathcal{E}} \text{ (IF)}$$

where

- $\langle \sigma, \mathcal{E} \rangle = \text{UWA}(s, s')$
- $ks' + t_2 \gtrsim_2 0 \not\vdash_{\text{sel}}^\sigma C \vee js + t_1 \gtrsim_1 0$
- or $js + t_1 \gtrsim_1 0 \not\vdash_{\text{sel}}^\sigma C \vee ks' + t_2 \gtrsim_2 0$
- $\gtrsim_1, \gtrsim_2 \in \{>, \geq\}$
- $\gtrsim_3 = \begin{cases} \geq & \text{if } \gtrsim_1 = \geq, \text{ and } \gtrsim_2 = > \\ > & \text{else} \end{cases}$
- $j, k > 0$

Term Factoring

$$\frac{L[j \underline{s} + k \underline{s}' + t] \vee C}{(L[(j+k)s + t] \vee C) \sigma \vee \mathcal{E}} \text{ (TF)}$$

where

- $\langle \sigma, \mathcal{E} \rangle = \text{UWA}(s, s')$
- $(j \underline{s} + k \underline{s}' + t)\sigma \triangleleft u \in \text{lead}((L[j \underline{s} + k \underline{s}' + t] \vee C)\sigma)$

Variable Elimination

$$\frac{C \vee \bigvee_{i \in I} (k_i \underline{x}_i + t_i \diamond_i 0)}{\bigwedge_{j \in J} (C \vee C_j)} \text{ (VE)}$$

where

- $\forall i. \diamond_i \in \{\approx, \not\approx, >, \geq\}$
- $\{C_j \mid j \in J\} = \text{QE}_x(\bigvee_{i \in I} (k_i x + t_i \diamond_i 0))$
- QE_x is the universal quantifier elimination of the variable x , for details of this rule we refer to [29]
- $x \in \mathbf{V}$ is unshielded in the premise

Contradiction

$$\frac{C \vee q \diamond 0}{C} \text{ (Ctd)}$$

where

- $q \in \mathbb{Q}$
- $\diamond \in \{>, \geq, \approx, \not\approx\}$
- $\mathbb{Q} \not\models q \diamond 0$

Figure 9: (Repeated Fig. 4) Inference rules of our revised calculus $\text{ALASCA}_{\gtrsim}^{\text{sel}}$ parametrized by an ALASCA-ordering $\prec$ (Def. 2) and an ALASCA-compatible selection function sel (Def. 6 in Sect. 5). Notation $L \diamond_{\text{sel}}^\sigma C$ stands for “Either $\forall L' \in C. L\sigma \diamond L'\sigma$ and $\emptyset = \text{sel}(L\sigma \vee C\sigma)$, or $L\sigma \in \text{sel}(L\sigma \vee C\sigma)$ ”. We consider arbitrary $C, C_i \in \text{Cls}, L \in \text{Lit}, j, k, q \in \mathbb{Q}, s, s', s_i \in \text{Trm} \setminus \mathbf{V}, t, t_i \in \text{Trm}, \diamond, \diamond_i \in \{>, \geq, \approx, \not\approx\}, \gtrsim, \gtrsim_i \in \{>, \geq\}$.

as it is also used to add up terms occurring nested in subterms (e.g. $g(f(x) + f(y)) > 0 \vdash g(2f(x)) > 0$), and not only on the top level (e.g. $f(x) + f(y) > 0 \vdash 2f(x) > 0$) as in ALASCA_0 . This is needed as in this paper unification is performed modulo **AC** instead of modulo $\mathcal{A}_{\text{eq}}$.

- **Abstraction (Abs)** is needed for cases where normalization cannot be lifted to clauses with variables properly (see Sect. A.2.1). It derives for example $f(3x+a) > 0 \vdash f(y) > 0 \vee 3x+a \approx y$. This rule is not needed in ALASCA_0 as in that calculus normalization is performed implicitly by unifying modulo $\mathcal{A}_{\text{eq}}$.
- **Variable Elimination (VE)** is needed to process so-called unshielded variables, that is variables which occur only on the top level of a clause and are not shielded under some uninterpreted function. For this it uses a quantifier elimination procedure. An example is $f(y) + x > 0 \vee b - x > 0 \vee c > 0 \vdash f(y) + b > 0 \vee c > 0$.

A.1 Ordering Issues in ALASCA

The clause ordering $\prec$ of ALASCA_0 is defined in terms of a term ordering, also referred to as $\prec$. As the rules of ALASCA_0 are formulated modulo $\mathcal{A}_{\text{eq}}$, in order for the calculus to be complete, its term ordering $\prec$ needs to be formulated in such a way that any two ground terms s, t that are equal modulo $\mathcal{A}_{\text{eq}}$ are in the same $\prec$ -equivalence class. This leads to the fact that, in contrast to standard simplification orderings used in superposition [37], the simplification ordering (QKBO) used in ALASCA_0 does not have the subterm property – the property that $s \triangleleft t$ implies $s \prec t$ – but only a weaker form of it. The reason for this is the additional requirement that the ordering is stable under ground substitutions ($s \prec t$ implies $s\theta \prec t\theta$), as illustrated by the following example:

Example 5 (Orderings and Subterms). *Consider the terms x and $f(x+y)$. Even though x is a subterm of $f(x+y)$, it cannot be the case that $x \prec f(x+y)$, as explained next. Consider the substitution $\theta = \{x \mapsto f(f(0)), y \mapsto -f(f(0))\}$. We get $x\theta = f(f(0))$ and $f(x+y)\theta = f(f(f(0)) - f(f(0))) \equiv_{\mathcal{A}_{\text{eq}}} f(0)$. For these ground terms we have $x\theta = f(f(0)) \succ f(0) = f(x+y)\theta$, thus it cannot be the case that $x \prec f(x+y)$.*

This fact is unfortunate, as we would like to orient equalities $f(x+y) \approx x$ in order to use them for our simplification rule ALASCA Demodulation (Dmd) (Sect. 4). Therefore, we introduce changes in $\text{ALASCA}_{\prec}^{\text{sel}}$ such that its term ordering possesses the subterm property modulo **AC**.

A.2 Reasoning modulo AC

To achieve a term ordering that allows for simplifications modulo **AC**, we make the following *main conceptual change in $\text{ALASCA}_{\prec}^{\text{sel}}$* : instead of formulating inference rules modulo $\mathcal{A}_{\text{eq}}$ as in ALASCA_0 , in $\text{ALASCA}_{\prec}^{\text{sel}}$ we formulate them modulo **AC** only and implement the remaining axioms $\mathcal{A}_{\text{eq}}^{\overline{\text{AC}}} = \mathcal{A}_{\text{eq}} \setminus \text{AC}$ using dedicated inference rules (TF $\cup$ NF) instead of handling them by UWA.

We therefore introduce normalization rules, that is, inference rules capturing $\mathcal{A}_{\text{eq}}^{\overline{\text{AC}}}$. Further, as the inference rules $\text{ALASCA}_{\prec}^{\text{sel}}$ are formulated modulo **AC** instead of $\mathcal{A}_{\text{eq}}$, we adjust the term ordering $\prec$ as follows: (i) any two **AC**-equivalent terms need to be in the same equivalence class of the ordering; and (ii) the equations from $\mathcal{A}_{\text{eq}}^{\overline{\text{AC}}}$ need to be oriented in the right way for the newly introduced inference rules. Additionally, in order to make our simplifying inference

Inequality Merging (IM) (Sect. 4) work correctly, our literal ordering is adjusted. Finally, we also have to adjust UWA to unify modulo **AC**, instead of modulo $\mathcal{A}_{\text{eq}}$.

A.2.1 Normalization

Most of the axioms in $\mathcal{A}_{\text{eq}}^{\overline{\mathbf{AC}}}$ can be processed in a straight forward manner without the need for unification. These are all captured by the inference rule Normalization (NF). This rule can be seen as a specialized Superposition (Sup) rewrite using implicit side premises $s \approx t \in \mathcal{A}_{\text{eq}}^{\overline{\mathbf{AC}}}$. Therefore it inherits the side condition from Sup, that rewrites need to be only applied to subterms ($s \trianglelefteq_{\mathbf{AC}} u$) of leading terms of a clause ($u \in \text{lead}(C)$).

The only axioms that need special care are $\{\forall x. jx + kx \approx (j+k)x \mid j, k \in \mathbb{Q}\}$. For these the rule Term Factoring (TF) is needed in order to unify candidate terms for x . To illustrate this, consider the clauses $-g(2f(a)) > 0$ and $g(f(x) + f(y)) > 0$. TF will derive $g(f(y) + f(x)) > 0 \vdash g(2f(x)) > 0$, and then we use FM to derive $-g(2f(a)) > 0, g(2f(x)) > 0 \vdash 0 > 0$. This rule was not required in the original ALASCA calculus [29] as such cases would be handled by UWA instead: Using FM with UWA, we would derive $-g(2f(a)) > 0, g(f(x) + f(y)) > 0 \vdash 0 > 0 \vee f(x) + f(y) - 2f(a) \not\approx 0$, which would then be processed by the rule Term Factoring. The rule can be seen as a specialized superposition inference as well.

We point out that each of these rules is formulated modulo **AC** (using the underlining syntax like $k\underline{s} + t$), which means for example that $f(0+b) \Rightarrow f(b)$, even though $(b+0)$ does not syntactically occur in the term. When constructing our new ALASCA ordering, we will make sure that, for every ground substitution θ , and rule $s \approx t \in \mathcal{A}_{\text{eq}}^{\overline{\mathbf{AC}}}$ we have $s\theta \succ t\theta$ (see ALASCA Term Ordering (Def. 1)), in order to ensure that the calculus is complete.

For ground clauses, NF and TF cover all necessary rewrites. However, to lift it to non-ground clauses, corner-cases arise when clauses containing variables occur. For these cases, we introduce the new inference rule Abstraction (Abs), as given in Fig. 4.

Abstraction. The rule Abstraction (Abs) is needed for lifting rewrites of non-trivial sums $k_1 t_1 + \dots + k_n t_n$, where some t_i is a variable. To see why this is needed, take for example the clause $f(3x) + f(2x) > 0$. Assume we want to lift the rewrite rules $j(kt) \Rightarrow (jk)t$, that is, perform a non-ground inference that covers all ground instances of this rewrite rule. We would then have to in particular use the rewrite $3(kx) \Rightarrow (3k)x$, for any $k \in \mathbb{Q}$, resulting in an infinite set of successors $\{f((3q)x) + f(2(qx)) > 0 \mid q \in \mathbb{Q}\}$. Similar issues occur when lifting the other rewrite rules. In order to resolve such issues, we introduce a new inference rule Abs (see Fig. 4), similar to the one used in [45], which moves these subterms into new disequality literals where they can later be processed by the rule Variable Elimination (VE) as in ALASCA_0 . For our example, we would rewrite the clause to $f(y) + f(z) > 0 \vee 3x \not\approx y \vee 2x \not\approx z$. VE will then produce the clause to $f(y) + f(z) > 0 \vee \frac{2}{3}y \not\approx z$, and reasoning will continue with the literal $f(y) + f(y) > 0$ until either of the variables y, z becomes unshielded and can be processed by VE. Note, that Abs can be restricted to uninterpreted contexts $C[o]$. This helps avoiding infinite chains of applications in Abs, where a chain $f(3x) > 0 \Rightarrow f(x_1) > 0 \vee x_1 \not\approx 3x \Rightarrow f(x_1) > 0 \vee x_1 \not\approx x_2 \vee x_2 \not\approx 3x \Rightarrow \dots$ would be possible if we would not restrict applications to uninterpreted contexts.

B Unification With Abstraction

In contrast to the original ALASCA_0 , our updated $\text{ALASCA}_0^{\text{sel}}$ formulates rules modulo **AC** instead of modulo $\mathcal{A}_{\text{eq}}$. Thus, UWA needs to be updated in order to unify modulo **AC** as well. As described in [8], the UWA algorithm is parametrized by a so-called abstraction oracle: a partial

function $\mathbf{ao}$ that takes two terms s, t and either returns $\mathbf{ao}(s, t) = \perp$, or $\mathbf{ao}(s, t) = \langle \mathit{unif}, \mathit{constr} \rangle$. Both unif and constr are sets of pairs of terms $s' \sim t' \in \mathit{unif} \cup \mathit{constr}$ which signal the UWA algorithm that all terms s', t' need to unify in order to make $s \sim t$ equal. More specifically, our UWA algorithm will introduce a constraint $s' \not\approx t'$ for pairs in constr , and recurse on all pairs in unif , processing them in the main loop of the algorithm. If there are multiple different unifiers for a pair $s' \sim t'$, it must be added to constr , as UWA can only compute one abstracting unifier per pair of input terms s, t . If there is at most one unifier for $s' \sim t'$, the pair should be in constr as this ensures further processing and a more accurate unifier. If $\perp$ is returned by $\mathbf{ao}$ this means that the terms s, t are not unifiable, and if $\mathbf{ao}$ is undefined then the unification algorithm should progress unifying by traversing subterms as in normal unification.

Definition 8 (Abstraction Oracle). *We define the abstraction oracle ($\mathbf{ao}$) as follows:*

$$\mathbf{ao}(s, t) = \begin{cases} \text{undefined} & \text{if } \mathbf{sym}(s) \neq + \neq \mathbf{sym}(t) & (\mathbf{ao}_0) \\ \perp & \text{if } \mathbf{ao}(s^-, t^-) = \perp & (\mathbf{ao}_1) \\ \langle \emptyset, \{s \sim t\} \rangle & \text{if } \mathbf{ao}(s^-, t^-) = \{\mathbf{smnds}(s) \sim \mathbf{smnds}(t)\} & (\mathbf{ao}_2) \\ \langle \{\Sigma L \sim \Sigma R \mid L \sim R \in \mathbf{ao}(s^-, t^-)\}, \emptyset \rangle & \text{else} & (\mathbf{ao}_3) \end{cases}$$

where $u^- = \mathbf{smnds}(u) \setminus (\mathbf{smnds}(s) \cap \mathbf{smnds}(t))$

$$\begin{aligned} \mathbf{ao}(\emptyset, \emptyset) &= \emptyset & (\mathbf{ao}_1) \\ \mathbf{ao}(S, \emptyset) &= \perp & \text{else} & (\mathbf{ao}_2) \\ \mathbf{ao}(\{x\}, T) &= \perp & \text{else if } x \triangleleft t \in T & (\mathbf{ao}_3) \\ \mathbf{ao}(\dot{x}, T) &= \{x \sim T\} & \text{else} & (\mathbf{ao}_4) \\ \mathbf{ao}(S, T) &= \{S^{=f} \sim T^{=f} \mid f \in \mathbf{match}\} & \text{else if } S^{\mathbf{V}} = T^{\mathbf{V}} = \emptyset, \mathbf{miss} = \emptyset & (\mathbf{ao}_5) \\ \mathbf{ao}(S, T) &= \perp & \text{else if } S^{\mathbf{V}} = T^{\mathbf{V}} = \emptyset, \mathbf{miss} \neq \emptyset & (\mathbf{ao}_6) \\ \mathbf{ao}(S, T) &= \{S^{=f} \sim T^{=f} \mid f \in \mathbf{match}\} \cup \{S^{\mathbf{V}} \cup S^{\in \mathbf{miss}} \sim T^{\in \mathbf{miss}}\} & \text{else if } |S| > 1, |S^{\mathbf{V}}| \geq 1, T^{\mathbf{V}} = \emptyset, \\ & & \forall f. |S^{=f}| \leq |T^{=f}|, |S^{\mathbf{V}} \cup S^{\in \mathbf{miss}}| \leq |T^{\in \mathbf{miss}}| & (\mathbf{ao}_7) \\ \mathbf{ao}(S, T) &= \perp & \text{else if } |S| > 1, |S^{\mathbf{V}}| \geq 1, T^{\mathbf{V}} = \emptyset, \exists f. |S^{=f}| > |T^{=f}| & (\mathbf{ao}_8) \\ \mathbf{ao}(S, T) &= \{S \sim T\} & \text{else if } |S| > 1, |S^{\mathbf{V}}| \geq 1, |T| > 1, |T^{\mathbf{V}}| \geq 1 & (\mathbf{ao}_9) \\ \mathbf{ao}(S, T) &= \mathbf{ao}(T, S) \end{aligned}$$

$$\begin{aligned} T^{\mathbf{V}} &= \{x \mid x \in \mathbf{V} \cap T\} & \mathbf{fns}(T) &= \{f \mid f(\dots) \in T\} \\ T^{=f} &= \{t \in T \mid \mathbf{sym}(t) = f\} & \mathbf{match} &= \{f \in \mathbf{fns}(S \cup T) \mid |S^{=f}| = |T^{=f}|\} \\ T^{\in F} &= \{t \in T \mid \mathbf{sym}(t) \in F\} & \mathbf{miss} &= \{f \in \mathbf{fns}(S \cup T) \mid |S^{=f}| \neq |T^{=f}|\} \end{aligned}$$

Def. 8 introduces the abstraction oracle $\mathbf{ao}$ used by UWA in our revised $\text{ALASCA}_{\sim}^{\text{sel}}$ calculus, fulfilling the following properties. Before formally proving that it fulfils Lemma 2, let us give some intuition for the individual steps in the rather lengthy Definition 8.

As we unify modulo $\mathbf{AC}$ of $+$, we want to treat sums as multisets. Thus, we first use the function $\mathbf{smnds}$ to transform the terms s, t into their respective summands. As the terms $s_1 + \dots + u + \dots + s_n$, and $t_1 + \dots + u + \dots + t_m$ are equal modulo $\mathbf{AC}$ iff $s_1 + \dots + s_n$ and $t_1 + \dots + t_m$ are equal modulo $\mathbf{AC}$, we compute s^-, t^- cancelling out all common terms in $\mathbf{smnds}(s) \cap \mathbf{smnds}(T)$.

In $(\mathbf{ao}_2)$, the main function $\mathbf{ao}$ is used to implement the abstraction oracle, computing a set of pairs $L \sim R$ of multisets L, R of terms or $\perp$. The intended meaning of these pairs

$L \sim R$ is that when we have $\sum L \equiv_{\mathbf{AC}} \sum R$ for all such pairs $L \sim R$, then $s \equiv_{\mathbf{AC}} t$. Case (ao₃) is there for preventing infinite loops in the UWA algorithm by introducing a constraint if $\text{ao}(s^-, t^-) = \{\text{smnds}(s) \sim \text{smnds}(t)\}$.

The cases (ao₁) - (ao₄), are rather straightforward: Case (ao₁) reflects that two empty sums are always equal, while case (ao₂) asserts that an empty sum can never be equal to a non-empty one. In case (ao₃), we make sure that a variable x can never unify with a term t that contains the variable as a strict subterm ($x \triangleleft t$), and case (ao₄) returns the pair $\{x\} \sim T$ back to the UWA algorithm which will make sure that $x \rightarrow \sum T$ in the substitution of the final abstracting unifier.

In the remaining cases some auxiliary definitions are used: $X^{=f}$ is the multiset of all terms in X that have the top level symbol f . By $X^{\mathbf{V}}$ we denote the multiset of all variables in X . We partition the set of all top level functions into the sets **miss** and **match**, **match** containing all functions f where the number of terms with top level symbol f in both multisets S, T is equal, and **miss** containing all other functions.

The multisets S, T are the arguments of **ao** that represent the summands of s, t used for **ao** with duplicates cancelled out. If there are no variables in S, T , then every term S must unify with exactly one term T with the same top symbol. If this is not possible **miss** will be non-empty and the S, T are not unifiable (ao₆). If **miss** = $\emptyset$, then S, T can unify exactly when all the sub-sums with matching symbols unify (ao₅).

If the multiset S contains variables and T does not, then S and T can only unify if all the subsums with matching symbol counts unify $\{S^{=f} \sim T^{=f} \mid f \in \text{match}\}$, while all the summands with symbols in **miss** must unify with the variables $S^{\mathbf{V}} \cup S^{\in \text{miss}} \sim T^{\in \text{miss}}$ (ao₇). If any of these symbols $f \in \text{miss}$ occurs in S more often than in T , then two terms can obviously never unify as T does not contain variables which is asserted by (ao₈).

In (ao₉), where both S, T contain variables and other terms, we introduce a constraint $S \sim T$, as in this case multiple different ways of unifying these variables might be possible.

We will now give the intuition of the individual steps in the rather lengthy Definition 8, before formally proving that it fulfils Lemma 2.

As we unify modulo **AC** of $+$, we want to treat sums as multisets. Thus, we first use the function **smnds** to transform the terms s, t into their respective summands. As the terms $s_1 + \dots + u + \dots + s_n$, and $t_1 + \dots + u + \dots + t_m$ are equal modulo **AC** iff $s_1 + \dots + s_n$ and $t_1 + \dots + t_m$ are equal modulo **AC**, we compute s^-, t^- cancelling out all common terms in $\text{smnds}(s) \cap \text{smnds}(t)$.

In (ao₂), the main function **ao** is used to implement the abstraction oracle, computing a set of pairs $L \sim R$ of multisets L, R of terms or $\perp$. The intended meaning of these pairs $L \sim R$ is that when we have $\sum L \equiv_{\mathbf{AC}} \sum R$ for all such pairs $L \sim R$, then $s \equiv_{\mathbf{AC}} t$. Case (ao₃) is there for preventing infinite loops in the UWA algorithm by introducing a constraint if $\text{ao}(s^-, t^-) = \{\text{smnds}(s) \sim \text{smnds}(t)\}$.

The cases (ao₁) - (ao₄), are rather straightforward: Case (ao₁) reflects that two empty sums are always equal, while case (ao₂) asserts that an empty sum can never be equal to a non-empty one. In case (ao₃), we make sure that a variable x can never unify with a term t that contains the variable as a strict subterm ($x \triangleleft t$), and case (ao₄) returns the pair $\{x\} \sim T$ back to the UWA algorithm which will make sure that $x \rightarrow \sum T$ in the substitution of the final abstracting unifier.

In the remaining cases some auxiliary definitions are used: $X^{=f}$ is the multiset of all terms in X that have the top level symbol f . By $X^{\mathbf{V}}$ we denote the multiset of all variables in X . We partition the set of all top level functions into the sets **miss** and **match**, **match** containing all functions f where the number of terms with top level symbol f in both multisets S, T is equal,

and **miss** containing all other functions.

The multisets S, T are the arguments of $\mathbf{ao}$ that represent the summands of s, t used for $\mathbf{ao}$ with duplicates cancelled out. If there are no variables in S, T , then every term S must unify with exactly one term T with the same top symbol. If this is not possible **miss** will be non-empty and the S, T are not unifiable ($\mathbf{ao}_6$). If **miss** = $\emptyset$, then S, T can unify exactly when all the sub-sums with matching symbols unify ($\mathbf{ao}_5$).

If the multiset S contains variables and T does not, then S and T can only unify if all the subsums with matching symbol counts unify $\{S^{=f} \sim T^{=f} \mid f \in \mathbf{match}\}$, while all the summands with symbols in **miss** must unify with the variables $S^{\mathbf{V}} \cup S^{\in \mathbf{miss}} \sim T^{\in \mathbf{miss}}$ ($\mathbf{ao}_7$). If any of these symbols $f \in \mathbf{miss}$ occurs in S more often than in T , then two terms can obviously never unify as T does not contain variables which is asserted by ($\mathbf{ao}_8$).

In ($\mathbf{ao}_9$), where both S, T contain variables and other terms, we introduce a constraint $S \sim T$, as in this case multiple different ways of unifying these variables might be possible.

Lemma 2 (Abstraction Oracle). *$\mathbf{ao}$ is **AC**-sound, **AC**-minimal, $\prec$ -founded, terminating, and captures **AC** for any ALASCA-ordering $\prec$.*

Proof. Let us expand the definitions from [8] of the properties in the lemma. For termination we define the following ordering

$$s \sim t \ll s' \sim t' \iff s \prec s' \ \& \ t \preceq t' \text{ or } s \preceq s' \ \& \ t \prec t'$$

and consider θ to be a ground substitution.

$$\begin{aligned} (P_1) \quad \mathbf{AC}\text{-sound} \quad & \begin{cases} \mathbf{ao}(s, t) = \langle \mathbf{unif}, \mathbf{constr} \rangle & \Rightarrow \mathbf{AC}, (\mathbf{unif} \cup \mathbf{constr})^\approx \models s \approx t \\ \mathbf{ao}(s, t) = \perp & \Rightarrow \forall \theta. \mathbf{AC} \not\models (s \approx t)\theta \end{cases} \\ (P_2) \quad \mathbf{AC}\text{-minimal} \quad & \mathbf{ao}(s, t) = \langle \mathbf{unif}, \mathbf{constr} \rangle \Rightarrow \forall \theta (s\theta \equiv_{\mathbf{AC}} t\theta \Rightarrow \mathbf{AC} \models (\mathbf{unif} \cup \mathbf{constr})^\approx \theta) \\ (P_3) \quad \prec\text{-founded} \quad & \mathbf{ao}(s, t) = \langle \mathbf{unif}, \mathbf{constr} \rangle \Rightarrow \forall \theta (s\theta \equiv_{\mathbf{AC}} t\theta \Rightarrow (\mathbf{unif} \cup \mathbf{constr})^\approx \theta \preceq (s \not\approx t)\theta) \\ (P_4) \quad \text{captures } \mathbf{AC} \quad & \mathbf{ao}(s, t) = \mathbf{undef} \Rightarrow \nexists \theta (s\theta \equiv_{\mathbf{AC}} t\theta) \text{ or } \mathbf{sym}(s) = \mathbf{sym}(t) \text{ is uninterpreted wrt. } \mathbf{AC} \\ (P_5) \quad \text{terminating} \quad & \left(\begin{array}{l} \mathbf{ao}(s, t) = \langle \mathbf{unif}, \mathbf{constr} \rangle \\ \& s' \sim t' \in \mathbf{unif} \end{array} \right) \Rightarrow \left(\begin{array}{l} s' \sim t' \ll s \sim t \\ \text{or } \mathbf{ao}(s', t') = \mathbf{undef} \\ \text{or } s', t' \text{ are trivially unifiable} \end{array} \right) \end{aligned}$$

We first observe that

$$(P_6) \quad s^- \subseteq \mathbf{smnds}(s) \ \& \ t^- \subseteq \mathbf{smnds}(t)$$

$$(P_7) \quad \forall s' \sim t' \in \mathbf{ao}(s^-, t^-). s' \subseteq s^- \ \& \ t' \subseteq t^-$$

- P_5 terminating: By ($\mathbf{ao}_2$) if $\mathbf{ao}(s, t) = \langle \mathbf{unif}, \mathbf{constr} \rangle$ with $\mathbf{unif} \neq \emptyset$ we know that for every element $s' \sim t' \in \mathbf{unif}$ we have $s' \sim t' \neq \mathbf{smnds}(s) \sim \mathbf{smnds}(t)$. Thus by P_7 and as $\prec$ is **AC**-compatible and has the subterm property (Def. 2.1) we know that $s' \sim t' \ll s \sim t$.
- P_3 $\prec$ -founded: Follows from the fact that we only return **AC**-subterms of the original terms: For $s' \sim t' \in \mathbf{ao}(s, t)$ we have $s' \subseteq \mathbf{smnds}(s)$, $t' \subseteq \mathbf{smnds}(t)$ by P_6 and P_7 . Thus by P_7 and as $\prec$ is **AC**-compatible and has the subterm property (Def. 2.1) we know that $\sum s' \not\approx \sum t' \ll s \not\approx t$.
- P_4 captures **AC**: Obviously satisfied due to ($\mathbf{ao}_0$).
- P_2 **AC**-minimality, and P_1 **AC**-soundness: Obviously $\mathbf{AC} \models (\sum s^- \approx \sum t^- \iff s \approx t)$. Thus in the following it is sufficient to show all the equality properties for s^-, t^- .

We show this by going through the cases of the definition of $\mathfrak{a}\mathfrak{o}$. We sometimes will need counter-models to show soundness. For this we consider the model $I_{\mathbf{AC}}$

$$\begin{aligned} I_{\mathbf{AC}}(1) &= \llbracket 1 \rrbracket_{\mathbf{AC}} \\ I_{\mathbf{AC}}(k)(\llbracket t \rrbracket_{\mathbf{AC}}) &= \llbracket kt \rrbracket_{\mathbf{AC}} \\ I_{\mathbf{AC}}(+)(\llbracket s \rrbracket_{\mathbf{AC}}, \llbracket t \rrbracket_{\mathbf{AC}}) &= \llbracket s + t \rrbracket_{\mathbf{AC}} \\ I_{\mathbf{AC}}(f)(\llbracket t_1 \rrbracket_{\mathbf{AC}}, \dots, \llbracket t_n \rrbracket_{\mathbf{AC}}) &= \llbracket f(t_1, \dots, t_n) \rrbracket_{\mathbf{AC}} \\ I_{\mathbf{AC}}(> 0) &= I_{\mathbf{AC}}(\geq 0) = \emptyset \end{aligned}$$

Obviously $I_{\mathbf{AC}} \models \mathbf{AC}$.

case ($\mathfrak{a}\mathfrak{o}_1$): This can only be the case if $\mathfrak{s}\mathfrak{m}\mathfrak{n}\mathfrak{d}\mathfrak{s}(s) = \mathfrak{s}\mathfrak{m}\mathfrak{n}\mathfrak{d}\mathfrak{s}(t)$ which means that $\mathbf{AC} \models s \approx t$, hence $\mathbf{AC}$ -soundness P_1 holds, and as $\mathfrak{a}\mathfrak{o}(s, t) = \emptyset$ in this case $\mathbf{AC}$ -minimality P_2 also obviously holds. $\dashv$

case ($\mathfrak{a}\mathfrak{o}_2$): In this case $t \equiv_{\mathbf{AC}} s + t$, thus obviously $I_{\mathbf{AC}} \not\models s \approx s + t$, hence $\mathbf{AC}$ -soundness P_1 holds. $\dashv$

case ($\mathfrak{a}\mathfrak{o}_3$): In this case obviously $s^- \triangleleft_{\mathbf{AC}} t^-$, thus $I_{\mathbf{AC}}(s) \neq I_{\mathbf{AC}}(t)$, hence $\mathbf{AC}$ -soundness P_1 holds. $\dashv$

case ($\mathfrak{a}\mathfrak{o}_4$): In this case obviously both $\mathbf{AC}$ -soundness P_1 and $\mathbf{AC}$ -minimality P_2 hold, by just unfolding definitions. $\dashv$

case ($\mathfrak{a}\mathfrak{o}_5$): Obviously each of the $s' \sim t' \in \mathfrak{a}\mathfrak{o}(s, t)$ are sub-sums of $s \sim t$. And as $\mathfrak{miss} = \emptyset$ all sub-sums of s, t are covered, thus $\mathbf{AC}$ -soundness P_1 holds.

We show $\mathbf{AC}$ -minimality P_2 contra-positive. Consider an arbitrary ground substitution θ . Suppose there is some model I , and some $f \in \mathfrak{match}$ such that $I \not\models \sum S =^f \theta \approx \sum T =^f \theta$. This means that there must be some $s^* \in S =^f \theta \setminus T =^f \theta$ $t^* \in T =^f \theta \setminus S =^f \theta$ such that $s^* \not\equiv_{\mathbf{AC}} t^*$. Thus obviously also $I_{\mathbf{AC}} \not\models \sum S =^f \theta \approx \sum T =^f \theta$, thus obviously also $I_{\mathbf{AC}} \not\models s\theta \approx t\theta$, thus $s \not\equiv_{\mathbf{AC}} t$, and thus $\mathbf{AC}$ -minimality P_2 holds. $\dashv$

case ($\mathfrak{a}\mathfrak{o}_6$): Let $f \in \mathfrak{miss}$. This means that

$$\begin{aligned} s^- &\equiv_{\mathbf{AC}} \underbrace{f(\dots) + \dots + f(\dots)}_{n \text{ times}} + s_0 \\ t^- &\equiv_{\mathbf{AC}} \underbrace{f(\dots) + \dots + f(\dots)}_{m \text{ times}} + t_0 \end{aligned}$$

with $n \neq m$. And all summands of s_0, t_0 are non-variables, do not have f as symbol. Thus obviously $I_{\mathbf{AC}} \not\models \sum s^- \approx \sum t^-$ for any θ , thus $\mathbf{AC}$ -soundness P_1 holds. $\dashv$

case ($\mathfrak{a}\mathfrak{o}_7$): Obviously each of the $s' \sim t' \in \mathfrak{a}\mathfrak{o}(s, t)$ are sub-sums of $s \sim t$, and all sub-sums of s, t are covered, thus $\mathbf{AC}$ -soundness P_1 holds.

Consider an arbitrary ground substitution θ .

case $\exists f, t \in \text{smnds}(\sum S^V \theta). f = \text{sym}(t) \in \text{match}$: Then

$$\begin{aligned} s^- \theta &\equiv_{\mathbf{AC}} \underbrace{f(\dots) + \dots + f(\dots)}_{n \text{ times}} + s_0 \\ t^- \theta &\equiv_{\mathbf{AC}} \underbrace{f(\dots) + \dots + f(\dots)}_{m \text{ times}} + t_0 \end{aligned}$$

with $n > m$, where s_0, t_0 do not have terms with f as function as symbol. Thus obviously $I_{\mathbf{AC}} \not\models \sum s^- \theta \approx \sum t^- \theta$, hence in this case $s\theta \not\equiv_{\mathbf{AC}} t\theta$. $\dashv$

case $\forall t \in \text{smnds}(\sum S^V \theta). \text{sym}(t) \notin \text{match}$: Again we show $\mathbf{AC}$ -minimality P_2 contrapositive. Consider some model I such that $I \not\models \text{ao}(s, t) \approx$.

case $I \not\models \sum S^{=f} \theta \approx \sum T^{=f} \theta$ for some $f \in \text{match}$: As in the case before this means that there must be some $s^* \in S^{=f} \theta \setminus T^{=f} \theta, t^* \in T^{=f} \theta \setminus S^{=f} \theta$ such that $s^* \not\equiv_{\mathbf{AC}} t^*$. Thus obviously also $I_{\mathbf{AC}} \not\models \sum S^{=f} \theta \approx \sum T^{=f} \theta$. By the case condition $\forall t \in \text{smnds}(\sum S^V \theta). \text{sym}(t) \notin \text{match}$ we also know that $I_{\mathbf{AC}} \not\models s\theta \approx t\theta$, thus $s \not\equiv_{\mathbf{AC}} t$, and thus $\mathbf{AC}$ -minimality P_2 holds. $\dashv$

case $I \not\models (\sum S^V + \sum S^{\in \text{miss}}) \theta \approx \sum T^{\in \text{miss}}$: As in the case before this means that

there must be some $s^* \in \text{smnds}(l) \setminus \text{smnds}(r), t^* \in \text{smnds}(r) \setminus \text{smnds}(l)$ such that $s^* \not\equiv_{\mathbf{AC}} t^*$. Thus obviously also $I_{\mathbf{AC}} \not\models l \approx r$. Further by the case condition $\forall t \in \text{smnds}(\sum S^V \theta). \text{sym}(t) \notin \text{match}$ we also know that $I_{\mathbf{AC}} \not\models s\theta \approx t\theta$, thus $s \not\equiv_{\mathbf{AC}} t$, and thus $\mathbf{AC}$ -minimality P_2 holds. $\dashv$

case (ao_8) : Consider any θ . Thus we know that

$$\begin{aligned} s^- \theta &\equiv_{\mathbf{AC}} \underbrace{f(\dots) + \dots + f(\dots)}_{n \text{ times}} + s_0 \\ t^- \theta &\equiv_{\mathbf{AC}} \underbrace{f(\dots) + \dots + f(\dots)}_{m \text{ times}} + t_0 \end{aligned}$$

with $n > m$, where s_0, t_0 do not have terms with f as function as symbol. Thus $I_{\mathbf{AC}} \not\models s^- \theta \approx t^- \theta$, hence $s\theta \not\equiv_{\mathbf{AC}} t\theta$. $\dashv$

case (ao_9) : In this case obviously both $\mathbf{AC}$ -soundness P_1 and $\mathbf{AC}$ -minimality P_2 hold, by just unfolding definitions. $\dashv$

□

C Ordering

In Sect. 3 we defined properties (Def. 1, and Def. 2) we require from the ordering $\prec$ for $\text{ALASCA}_{\prec}^{\text{sel}}$. We introduced a concrete orderings used in our implementation in Sect. 3.3. The main ALASCA Ordering used in the calculus is parametrized by an arbitrary ALASCA Term Ordering. It is called the Literal Extension (Def. 5). The ALASCA Term Ordering we use as the parameter for Def. 5 in our implementation is the Skeleton Term Ordering (Def. 4) introduced in Sect. 3.3. It is in turn parametrized by an arbitrary simplification ordering for which we use VAMPIRE's default ordering, determined by the options used to run VAMPIRE.

Before showing that these orderings (Def. 4 and 5) satisfy Def. 2, we give an intuition why these properties are required at all.

C.1 ALASCA-Ordering Intuition

In $\text{ALASCA}_{\prec}^{\text{sel}}$, rewrites $t[s] \rightarrow t[s']$ are performed by Superposition-like rewrite rules (Sup, NF, and TF). These rewrites should result in smaller terms $t[s] \succ t[s']$. At the same time, $\text{ALASCA}_{\prec}^{\text{sel}}$ must be compatible with **AC**. These properties are reflected in Def. 1

Definition 1 (ALASCA Term Ordering; repeated). *We say a relation $\prec$ is an ALASCA term ordering iff*

1. *it is a simplification ordering modulo **AC**,*
2. *for any $k\underline{s} + t \in \text{gTrm}$, where s is atomic, we have $s \succ -\frac{1}{k}t$, and*
3. *for any $s \approx t \in \mathcal{A}_{\text{eq}}^{\overline{\text{AC}}}$, and ground substitution θ we have $s\theta \succ t\theta$, where $\mathcal{A}_{\text{eq}}^{\overline{\text{AC}}} = \mathcal{A}_{\text{eq}} \setminus \text{AC}$.*

Requirements for the literal ordering mirror properties needed from the ordering of ALASCA_0 in order to make that calculus complete. Even though we do not proof completeness of $\text{ALASCA}_{\prec}^{\text{sel}}$, we believe that these properties are essential for making the calculus work properly. Thus we require that literals are ordered by their maximal summands (Def. 2.4, Def. 2.5), equalities are smaller than inequalities and disequalities (Def. 2.6), lower bounds are smaller than upper bounds (Def. 2.7), and disequalities are smaller than inequalities (Def. 2.8).

In addition to these requirements arising from the completeness proof of ALASCA_0 , they also ensure that the new simplification rule Inequality Merging (IM) does indeed simplify its premise; i.e.: that it's premise is smaller than the conclusion (Def. 2.8).

Definition 2 (ALASCA Ordering; repeated). *We say $\prec$ is an ALASCA ordering iff for any $s, s_i, t, t_i \in \text{gTrm}$*

1. *$\prec$ is an ALASCA term ordering*
2. *$s \prec t \implies s \diamond 0 \prec t \diamond 0$*
3. *$\prec$ is closed under ground substitutions, and for gLit it is an **AC**-compatible well-founded strict partial ordering that is total modulo **AC**.*
4. *$s_1 \prec s_2 \implies j\underline{s}_1 + t \diamond_1 0 \prec k\underline{s}_2 + u \diamond_2 0$ where $\diamond_i \in \{>, \geq, \approx, \not\approx\}$*
5. *$\forall i. t_i \prec s \implies k_1 t_1 + \dots k_n t_n \diamond_1 0 \prec k\underline{s} + t \diamond_2 0$ where $\diamond_i \in \{>, \geq, \approx, \not\approx\}$*
6. *$k\underline{s} + t \approx 0 \prec j\underline{s} + u \diamond 0$ where $\diamond \in \{\not\approx, >, \geq\}$*

$$7. |k|\underline{s} + t \gtrsim_1 0 \prec -|j|\underline{s} + u \gtrsim_2 0 \text{ where } \gtrsim_i \in \{>, \geq\}$$

$$8. k\underline{s} + t \not\approx 0 \prec j\underline{s} + u \diamond 0 \text{ where } \diamond \in \{>, \geq\}$$

$$9. t \geq 0 \prec t > 0$$

C.2 ALASCA-Ordering Proofs

To formally show that Skeleton Term Ordering (Def. 4) indeed fulfils ALASCA Term Ordering (Def. 1), we first introduce Def. 9, and Lem. 3.

Definition 9 (Ordering Helper). *We define*

$$\begin{aligned} \text{trm}^*(t) &= \begin{cases} \text{trm}(t) & \text{if } \text{trm}(t) \neq t \\ t & \text{else} \end{cases} \\ \text{coeff}^*(t) &= \begin{cases} \langle \rangle & \text{if } \text{trm}(t) = t \\ \langle \text{coeff}(t), k_1, \dots, k_n \rangle & \text{else } \text{coeff}^*(t) = \langle k_1, \dots, k_n \rangle \end{cases} \end{aligned}$$

$$\begin{aligned} \text{cmp}^{\text{lex}}(\langle s_1 \dots s_m \rangle, \langle t_1 \dots t_n \rangle) &= \text{cmp}_{\mathbb{N}}(m, n) \oplus \text{cmp}(s_1, t_1) \oplus \dots \oplus \text{cmp}(s_n, t_n) \\ (\text{cmp} \oplus \text{cmp}')(\langle s, t \rangle, \langle s', t' \rangle) &= \text{cmp}(s, s') \oplus \text{cmp}(t, t') \end{aligned}$$

Lemma 3 (Ordering Helper). *Let $s, t \in \text{Trm}$ such that $\underline{\text{smnds}}(s) \cap \mathbf{V} = \underline{\text{smnds}}(t) \cap \mathbf{V}$. Then*

$$\begin{aligned} \text{cmp}_{\text{sk}}(s, t) &= \text{cmp}_{\text{sk}}^{\text{mul}}(\underline{\text{trms}}(s), \underline{\text{trms}}(t)) \oplus \text{cmp}_{\text{sk}}^{\text{mul}}(\underline{\text{smnds}}(s), \underline{\text{smnds}}(t)) \\ \text{cmp}_{\text{sk}}(s, t) &= (\text{cmp}_{\text{sk}} \oplus \text{cmp}_{\text{sk}}^{\text{lex}})(\langle \text{trm}^*(s), \text{coeff}^*(s) \rangle, \langle \text{trm}^*(t), \text{coeff}^*(t) \rangle) \end{aligned}$$

Proof.

- The first equation can be easily seen by unfolding definitions.
- The second equation can be shown by induction along the subterm relation on s, t .

□

Thus now we can show that Skeleton Term Ordering (Def. 4) is indeed an ALASCA term ordering.

Lemma 4 (Skeleton Ordering). *The skeleton ordering $\prec^{\text{sk}}$ is an ALASCA term ordering.*

Proof. We use $\prec_0$ for $\prec$.

- | | |
|---|---|
| 1. Let $s, t \in \text{Trm}$. We need to show that | |
| (P_8) AC -compatible | $s' \equiv_{\text{AC}} s \prec^{\text{sk}} t \equiv_{\text{AC}} t' \implies s' \prec^{\text{sk}} t'$ |
| (P_9) well-founded | $T \subseteq \text{Trm} \implies \exists t \in T. t \text{ is } \prec^{\text{sk}}\text{-minimal in } T$ |
| (P_{10}) monotonic | $s \prec^{\text{sk}} t \implies u[s] \prec^{\text{sk}} u[t]$ |
| (P_{11}) irreflexive | $s \not\prec^{\text{sk}} s$ |
| (P_{12}) transitive | $s \prec^{\text{sk}} t \prec^{\text{sk}} u \implies s \prec^{\text{sk}} u$ |
| (P_{13}) total | $s \prec^{\text{sk}} t \text{ or } t \prec^{\text{sk}} s \text{ or } s \equiv_{\text{AC}} t$ |
| (P_{14}) subterm property | $s \trianglelefteq t \implies s \prec^{\text{sk}} t$ |
| (P_{15}) closure | $s \prec^{\text{sk}} t \implies s\sigma \prec^{\text{sk}} t\sigma$ |
- } $s, t, s', t', u[s] \in \text{gTrm}$

We proof all these properties by induction on the multiset extension of the subterm relation modulo **AC**. We define the function trm^* as the least fixed point of trm . We need to show the following

- P_8 **AC**-compatible: We proof this by induction on the subterm ordering of s, s', t, t' . Suppose $s' \equiv_{\mathbf{AC}} s \prec^{\text{sk}} t \equiv_{\mathbf{AC}} t'$. Obviously $\text{skel}(s) = \text{skel}(s')$, and $\text{skel}(t) = \text{skel}(t')$.
 $\text{case skel}(s) \prec_0 \text{skel}(t)$: Then obviously $\text{skel}(s') \prec_0 \text{skel}(t')$, thus if $s \prec^{\text{sk}} t$. $\dashv$
- $\text{case skel}(s) = \text{skel}(t)$: Then obviously $\text{skel}(t') = \text{skel}(s')$.
 $\text{case } 1 = s \ \& \ 1 \neq t$: Then obviously $s' = 1 \prec^{\text{sk}} t' \neq 1$. $\dashv$
- $\text{case } + \in \{\text{sym}(s), \text{sym}(t)\} = \{\text{sym}(s'), \text{sym}(t')\}$: Then we can use the induction hypothesis on $\text{smnds}, \text{trms}$, of s, s', t, t' . From $s \prec^{\text{sk}} t$ we know that there can be two cases:
 $\text{case } \text{trms}(s) \prec^{\text{sk}} \text{trms}(t)$: In which case by induction hypothesis $\text{trms}(s') \prec^{\text{sk}} \text{trms}(t')$. $\dashv$
- $\text{case } \text{trms}(s) \equiv_{\mathbf{AC}} \text{trms}(t) \ \& \ \text{smnds}(s) \prec^{\text{sk}} \text{smnds}(t)$: In which case again by induction hypothesis $\text{smnds}(s') \prec^{\text{sk}} \text{smnds}(t')$. $\dashv$
- $\text{case } \emptyset \neq \mathbb{Q} \cap \{\text{sym}(s), \text{sym}(t)\}$: Then we can use the induction hypothesis on trm^* , of s, s', t, t' . By Ordering Helper (Lem. 3) we know that there can be two cases
 $\text{case } \text{trm}^*(s) \prec^{\text{sk}} \text{trm}^*(t)$: In which case by induction hypothesis we know that $s' \prec^{\text{sk}} t'$. $\dashv$
- $\text{case } \text{trm}^*(s) \equiv_{\mathbf{AC}} \text{trm}^*(t) \ \& \ \text{coeff}^*(s) \prec^{\text{sk}^{\text{lex}}} \text{coeff}^*(t)$: Thus as $\text{coeff}^*(s') = \text{coeff}^*(s)$, and $\text{coeff}^*(t') = \text{coeff}^*(t)$ we know that $s' \prec^{\text{sk}} t'$. $\dashv$
- P_{13} total modulo **AC**: We proof this property by induction on the subterm relation of s, t . Consider any two terms s, t such that $s \not\equiv_{\mathbf{AC}} t$.
 $\text{case skel}(s) \neq \text{skel}(t)$: By totality of $\prec_0$ we know that $s \prec t$ or $t \prec s$. $\dashv$
- $\text{case skel}(s) = \text{skel}(t)$:
 $\text{case } s = 1 \text{ or } t = 1$: Then the other term cannot be 1 thus $s \prec t$ or $t \prec s$ must hold. $\dashv$
- $\text{case } + \in \{\text{sym}(s), \text{sym}(t)\}$: Then we can use the induction hypothesis for terms in trms , and smnds of s, t .

case $\text{trms}(s) \not\equiv_{\mathbf{AC}} \text{trms}(t)$: Thus by induction hypothesis, and the definition of the multiset extension we know that $\text{trms}(s) \prec \text{trms}(t)$, or $\text{trms}(s) \succ \text{trms}(t)$.
 $\neg$

case $\text{trms}(s) \equiv_{\mathbf{AC}} \text{trms}(t)$: Then as $\sum \text{smnds}(s) \equiv_{\mathbf{AC}} s \not\equiv_{\mathbf{AC}} t \sum \text{smnds}(t)$. Thus by induction hypothesis, and the definition of the multiset extension we know that $\text{smnds}(s) \prec \text{smnds}(t)$, or $\text{smnds}(s) \succ \text{smnds}(t)$.
 $\neg$

case $\emptyset \neq \mathbb{Q} \cap \{\text{sym}(s), \text{sym}(t)\}$: Then we can use the induction hypothesis for $\text{trm}^*(s), \text{trm}^*(t)$.

case $\text{trm}^*(s) \not\equiv_{\mathbf{AC}} \text{trm}^*(t)$: Then by induction hypothesis and Ordering Helper (Def. 9) we know that $s \prec t$ or $t \prec s$.
 $\neg$

case $\text{trm}^*(s) \equiv_{\mathbf{AC}} \text{trm}^*(t)$: Then Ordering Helper (Def. 9) we know that $\text{coeff}^*(s) \neq \text{coeff}^*(t)$. By definition of $\circ^{\text{lex}}$ we know that it must be the case that $s \prec t$, or $t \prec s$.
 $\neg$

case $\text{sym}(s) = \text{sym}(t) = f$ is uninterpreted : Then we can use the induction hypothesis for the arguments and compare lexicographically, thus obviously either $s \prec t$, or $t \prec s$.
 $\neg$

- P_9 well-founded:

We proof this by induction on the multiset extension of the subterm relation on T . Consider its subset T_1 of all terms in T with $\prec_0$ -minimal skel . By well-foundedness of $\prec_0$ such a set exists, and by totality of $\prec_0$, the skeleton of all these terms must be equal to a term skel .

case $1 \in T_1$: Then obviously 1 is a minimal element of $T_1 \subseteq T$.
 $\neg$

case $\exists t \in T_1. \text{sym}(t) = +$: Thus we can use the induction hypothesis on $\{\text{trms}(t), \text{smnds}(t) \mid t \in T_1\}$.

Thus by induction hypothesis there is some set $T_2 \subseteq T_1$ where for $t' \in T_2$, $\text{trms}(t')$ is minimal in T_1 . Again by induction hypothesis there is some set $T_3 \subseteq T_2$ where for $t' \in T_3$, $\text{smnds}(t')$ is minimal in T_2 . By Ordering Helper (Lem. 3) and totality modulo $\mathbf{AC}$ P_{13} we know that any element in $T_3 \subset T$, is $\prec^{\text{sk}}$ minimal in T .
 $\neg$

case $\exists t \in T_1. \text{sym}(t) \in \mathbb{Q}$: Thus we can use the induction hypothesis on $\{\text{trm}^*(t) \mid t \in T_1\}$.

Let thus by induction hypothesis T_2 be the subset of T_1 that where trm^* is $\prec$ -minimal. Further let T_3 be the subset of T_2 that where coeff^* is $\prec^{\text{lex}}$ -minimal.

Thus we know that all of the elements in T_3 must be **AC**-equivalent. Therefore by Ordering Helper (Lem. 3) and totality modulo **AC** P_{13} we know that any element in $T_3 \subset T$, is $\prec^{\text{sk}}$ minimal in T . $\dashv$

case $\forall t \in T_1. \text{sym}(t) \notin \mathbb{Q} \cup +$: Then by Skeleton (Def. 3) all terms in T_1 must have the same uninterpreted function f as top symbol. Thus all subterms are compared lexicographically. We can use the induction hypothesis on the arguments, and as the lexicographic extension of a well-founded ordering is well-founded we know there is a minimal subset. $\dashv$

- P_{10} monotonic: We show this by induction on the subterm relation of $u[\circ]$. We need to show that $s \prec t \implies u[s] \prec u[t]$. Firstly, it is easy to see that if $\text{skel}(s) \prec \text{skel}(t)$, then $\text{skel}(u[s]) \prec \text{skel}(u[t])$. Thus suppose $\text{skel}(s) = \text{skel}(t) = \text{skel}$. We case split on the context $u[\circ]$.
case $u[\circ] = \circ$: Then the property trivially holds. $\dashv$

case $\text{sym}(u[\circ]) = +$: Then by induction hypothesis we have $\text{trms}(u[s]) \prec^{\text{mul}} \text{trms}(u[t])$, thus by (trm_5) $u[s] \prec u[t]$. $\dashv$

case $u[\circ] = k(u'[\circ])$: Then by induction hypothesis we know that $u'[s] \prec u'[t]$, thus by (trm_6) $u[s] \prec u[t]$. $\dashv$

case $u[\circ] = f(u_1, \dots, u_n)$: Then by induction hypothesis we know that $u_i[\circ/s] \prec u_i[\circ/t]$, where u_i contains $\circ$, and equal otherwise. Thus by (trm_9) $u[s] \prec u[t]$. $\dashv$

- P_{11} irreflexive: This follows from irreflexivity of $\prec_0$.
- P_{12} transitive: We proof this by induction on the subterm relation of s, t, u . Suppose $s < t < u$.

case $\text{skel}(s) = \text{skel}(t) \neq \text{skel}(u)$ or $\text{skel}(s) \neq \text{skel}(t) = \text{skel}(u)$: Then obviously $s \prec u$, by the skeleton comparison case. $\dashv$

case $\text{skel}(s) \neq \text{skel}(t) \neq \text{skel}(u)$: Then obviously $s \prec u$, by the skeleton comparison case, and transitivity of $\prec_0$. $\dashv$

case $\text{skel}(s) = \text{skel}(t) = \text{skel}(u)$:

case $+ \in \{\text{sym}(s), \text{sym}(t), \text{sym}(u)\}$: Then we can use the induction hypothesis for terms in trms , and smnds of s, t, u . We implicitly use the lemma Ordering Helper (Lem. 3) here.

case $\text{trms}(s) \prec \text{trms}(t) \equiv_{\text{AC}} \text{trms}(u)$ or $\text{trms}(s) \equiv_{\text{AC}} \text{trms}(t) \prec \text{trms}(u)$: Then obviously by **AC**-compability P_8 we know that $s \prec u$. $\dashv$

case $\text{trms}(s) \prec \text{trms}(t) \prec \text{trms}(u)$: Then by transitivity of the multiset extension $s \prec u$. $\dashv$

case $\text{trms}(s) \equiv_{\text{AC}} \text{trms}(t) \equiv_{\text{AC}} \text{trms}(u)$: Then we know that $\text{smnds}(s) \prec \text{smnds}(t) \prec \text{smnds}(u)$. Then by transitivity of the multiset extension $s \prec u$. $\dashv$

case $\emptyset \neq \mathbb{Q} \cap \{\text{sym}(s), \text{sym}(t), \text{sym}(u)\}$: Thus we can use the induction hypothesis on $\{\text{trm}^*(s), \text{trm}^*(t), \text{trm}^*(u)\}$. We implicitly use the lemma Ordering Helper (Lem. 3) here.

case $\text{trm}^*(s) \prec \text{trm}^*(t) \equiv_{\text{AC}} \text{trm}^*(u)$ or $\text{trm}^*(s) \equiv_{\text{AC}} \text{trm}^*(t) \prec \text{trm}^*(u)$: Then obviously by **AC**-compability P_8 and Ordering Helper (Lem. 3) know that $s \prec u$. $\dashv$

case $\text{trm}^*(s) \prec \text{trm}^*(t) \prec \text{trm}^*(u)$: Then induction hypothesis and Ordering Helper (Lem. 3) know that $s \prec u$. $\dashv$

case $\text{trm}^*(s) \equiv_{\text{AC}} \text{trm}^*(t) \equiv_{\text{AC}} \text{trm}^*(u) = u_0$: Then by Ordering Helper (Lem. 3) we know that $\text{coeff}^*(s) \prec \text{coeff}^*(t) \prec \text{coeff}^*(u)$, thus obviously by the definition of $\circ^{\text{lex}}$ we know that $s \prec u$. $\dashv$

case $\text{sym}(s) = \text{sym}(t) = \text{sym}(u) = f$ is uninterpreted : That means that the arguments are lexicographically compared, which means transitivity follows from the induction hypothesis. $\dashv$

- P_{14} fulfils the subterm property: We proof this by induction on the subterm relation of s, t . Consider some term $s \triangleleft t$.

case $\text{skel}(s) \neq \text{skel}(t)$: Then it must be the case that $\text{skel}(s) \triangleleft \text{skel}(t)$, thus by the subterm property of $\prec_0$ we know that $s \prec t$. $\dashv$

case $\text{skel}(s) = \text{skel}(t)$:

case $s = 1$: Then t cannot be 1 as s is a strict subterm, thus $s \prec t$. $\dashv$

case $+\in \{\text{sym}(s), \text{sym}(t)\}$: Then we can use the induction hypothesis for terms in trms , and smnds of s, t .

As $s \triangleleft t$ we know that every element in $\text{trms}(s)$ must be a subterm or equal to a term in $\text{trms}(t)$. Same for smnds . If all in trms are equal, then there must be more elements in $\text{trms}(t)$ thus $s \prec t$. Similarly for smnds .

Otherwise there must be at least one element in smnds or trms that is a strict subterm, thus by the induction hypothesis $s \prec t$. $\dashv$

case $\emptyset \neq \mathbb{Q} \cap \{\text{sym}(s), \text{sym}(t)\}$: Then we can use the induction hypothesis for $\{\text{trm}^*(s), \text{trm}^*(t)\}$.

case $\text{trm}^*(s) \neq \text{trm}^*(t)$: Then it must be the case that $\text{trm}^*(s) \triangleleft \text{trm}^*(t)$, thus by induction hypothesis and Ordering Helper (Lem. 3) we know that $s \prec t$. $\dashv$

case $\text{trm}^*(s) = \text{trm}^*(t) = t_0$: Then as $s \triangleleft t$ it must be the case that $\text{coeff}^*(s) = \langle k_1 \dots k_n \rangle$, and $\text{coeff}^*(t) = \langle j_1 \dots j_m, k_1 \dots k_n \rangle$ with $m > 0$, thus by Ordering Helper (Lem. 3) we know that $s \prec t$. $\dashv$

case $\text{sym}(s) = \text{sym}(t) = f$ is uninterpreted : Then we know that s must be a strict subterm of some argument of t , thus by the induction hypothesis and lexicographic comparison we know that $s \prec t$. $\dashv$

- $P_{15} \prec$ is closed under ground substitutions: Consider $s \prec t$, and any θ . We proof the property by induction on the subterm relation modulo **AC** on s, t .

case $\text{skel}(s) \prec \text{skel}(t)$: Then by $\prec_0$ being closed under ground substitutions, we know that $s\theta \prec t\theta$. $\dashv$

case $\text{skel}(s) = \text{skel}(t)$: Then it must be that case that $\text{skel}(s\theta) = \text{skel}(t\theta)$.

case $s = 1$: Then t cannot contain a variable as $\text{skel}(s) = \text{skel}(t)$, thus obviously $s\theta \prec t\theta$. $\dashv$

case $+$ $\in \{\text{sym}(s), \text{sym}(t)\}$: Then we can use the induction hypothesis for terms in trms , and smnds of s, t .

case $\text{trms}(s) \prec \text{trms}(t)$:
We know that $\text{trms}(t) \neq \emptyset$.

case $\exists x \in \mathbf{V} \cap \text{smnds}(s)$: Then by (trm_2) we know that $x \in \text{smnds}(t)$.

case $s = x$: Thus we know that $\text{trms}(s) = \{x\}$, and $\text{trms}(t) = \{x\} \cup \text{trms}(t')$ with $\text{trms}(t') \neq \emptyset$. Therefore $\text{trms}(s\theta) = \text{trms}(x\theta) \prec \text{trms}(x\theta) \cup \text{trms}(t'\theta) = \text{trms}(t\theta)$. Thus by Ordering Helper (Lem. 3) $s\theta \prec t\theta$. $\dashv$

case $s \neq x$: We define $s' = \sum(\text{smnds}(s) \setminus \{x\})$ and $t' = \sum(\text{smnds}(t) \setminus \{x\})$. We can use the induction hypothesis on $s'\theta \prec t'\theta$. Further by Ordering Helper (Lem. 3) there are two cases

case $\text{trms}(s'\theta) \prec \text{trms}(t'\theta)$: In this case obviously $\text{trms}(s\theta) = \text{trms}(s'\theta) \cup \text{trms}(x\theta) \prec \text{trms}(t'\theta) \cup \text{trms}(x\theta) = \text{trms}(t\theta)$. Thus $s\theta \prec t\theta$ by Ordering Helper (Lem. 3). $\dashv$

case $\text{trms}(s'\theta) \equiv_{\mathbf{AC}} \text{trms}(t'\theta)$ & $\text{smnds}(s'\theta) \prec \text{smnds}(t'\theta)$: In this case obviously $\text{trms}(s\theta) = \text{trms}(s'\theta) \cup \text{trms}(x\theta) \equiv_{\mathbf{AC}} \text{trms}(t'\theta) \cup \text{trms}(x\theta) = \text{trms}(t\theta)$, and $\text{smnds}(s\theta) = \text{smnds}(s'\theta) \cup \text{smnds}(x\theta) \prec \text{smnds}(t'\theta) \cup \text{smnds}(x\theta) = \text{smnds}(t\theta)$. Thus $s\theta \prec t\theta$ by Ordering Helper (Lem. 3). $\dashv$

else :

Thus we know that $t = k \underline{t_0} + t'$.

case $\exists t'_0 \in \text{trms}(s).t_0 \equiv_{\text{AC}} t'_0$: Then $s = j \underline{t'_0} + s'$, and $\text{trms}(s') \prec \text{trms}(t')$, hence $s'\theta \prec t'\theta$, thus by Ordering Helper (Lem. 3) there are two cases

case $\text{trms}(s'\theta) \prec \text{trms}(t'\theta)$: Then obviously $\text{trms}(s) = \text{trms}(s'\theta) \cup \text{trms}(t_0\theta) \prec \text{trms}(t'\theta) \cup \text{trms}(t_0\theta) = \text{trms}(t\theta)$, thus by Ordering Helper (Lem. 3) $s\theta \prec t\theta$. $\dashv$

case $\text{trms}(s'\theta) \equiv_{\text{AC}} \text{trms}(t'\theta) \ \& \ \text{smnds}(s'\theta) \prec \text{smnds}(t'\theta)$: Then obviously $\text{trms}(s) = \text{trms}(s'\theta) \cup \text{trms}(t_0\theta) \equiv_{\text{AC}} \text{trms}(t'\theta) \cup \text{trms}(t_0\theta) = \text{trms}(t\theta)$, and $\text{smnds}(s) = \text{smnds}(s'\theta) \cup \text{smnds}(t_0\theta) \prec \text{smnds}(t'\theta) \cup \text{smnds}(t_0\theta) = \text{smnds}(t\theta)$, thus by Ordering Helper (Lem. 3) $s\theta \prec t\theta$. $\dashv$

case $\forall t'_0 \in \text{trms}(s).t_0 \not\equiv_{\text{AC}} t'_0$: Then there must be some terms $\{s_0, \dots, s_n\}$, such that $\forall i. s_i \prec t_0$, and $s = k_0 \underline{s_0} + \dots + k_n \underline{s_n} + s'$. And $\text{trms}(s') \preceq \text{trms}(t')$, thus by induction hypothesis $\text{trms}(s'\theta) \prec \text{trms}(t'\theta)$. By the definition of trms it is easy to see that for every $s_i^* \in \text{trms}(s_i\theta)$ there is some $t_0^* \in \text{trms}(t_0\theta)$ such that $s_i^* \prec t_0^*$. Thus we know that $\text{trms}(s\theta) \prec \text{trms}(t\theta)$. $\dashv$

case $\text{trms}(s) \equiv_{\text{AC}} \text{trms}(t) \ \& \ \text{smnds}(s) \prec \text{trms}(t)$: Can be done analogously to the previous case.

case $\emptyset \neq \mathbb{Q} \cap \{\text{sym}(s), \text{sym}(t)\}$: Then we can use the induction hypothesis on $\text{trm}^*(s) \text{trm}^*(t)$. Then by Ordering Helper (Lem. 3) there are two cases

case $\text{trm}^*(s) \prec \text{trm}^*(t)$: Then by induction hypothesis $\text{trm}^*(s)\theta \prec \text{trm}^*(s)\theta$.

case $\text{trm}^*(s\theta) \neq \text{trm}^*(t\theta)$: Then $\text{trm}^*(s)$ must be a variable x . Thus by closure under ground substitutions of $\text{trm}^*(t)$ and the subterm property P_{14} we know that $x \prec \text{trm}^*(t)$.

case $\text{trm}^*(t\theta) \neq \text{trm}^*(s\theta)$: Then t must also be the variable x , so we have $\text{trm}^*(s\theta) = \text{trm}^*(t\theta)$, but as $\text{trm}^*(s)\theta \prec \text{trm}^*(t)\theta$ we know by Ordering

Proof.

- (1) This obviously holds as $\prec^{\text{lit}}$ extends $\prec^{\text{sk}}$.
- (2) Suppose $s \prec t$. By Ordering Helper (Lem. 3) we know that there are two cases

case $\text{trms}(s) \prec \text{trms}(t)$: Obviously then also $\text{atms}_{\text{lit}}(s \diamond 0) \prec \text{atms}_{\text{lit}}(t \diamond 0)$. $\dashv$

case $\text{trms}(s) \equiv_{\text{AC}} \text{trms}(t)$ & $\text{smnds}(s) \prec \text{smnds}(t)$: Obviously then also $\text{atms}_{\text{lit}}(s \diamond 0) \equiv_{\text{AC}} \text{atms}_{\text{lit}}(t \diamond 0)$. Further from $\text{smnds}(s) \prec \text{smnds}(t)$ it is easy to see that we get $\text{atms}_{\text{lit}}(s \diamond 0) \preceq \text{atms}_{\text{lit}}(t \diamond 0)$. If $\text{atms}_{\text{lit}}(s \diamond 0) \prec \text{atms}_{\text{lit}}(t \diamond 0)$, then the property holds. If $\text{atms}_{\text{lit}}(s \diamond 0) \equiv_{\text{AC}} \text{atms}_{\text{lit}}(t \diamond 0)$, then $s \diamond 0 \prec t \diamond 0$ by definition. $\dashv$
- (3) This can be easily shown from the respective properties of $\prec^{\text{sk}}$.
- (4) This follows from the definition of (lit_1) .
- (5) This obviously holds by (lit_1) .
- (6), (7), (8), (9) These all hold by (lit_1) , and the definition of $\prec_{\text{sym}}$.

□

D Simplification and Redundancy Proofs

Lemma 6 (Inequality Merging). *IM is compatible with $\text{ALASCA}_{\prec}^{\text{sel}}$.*

Proof. Consider any $\iota \in \text{IM}$. Then

$$\iota = \frac{C \vee s \diamond_1 0 \vee s' \diamond_2 0}{C \vee s \diamond_3 0}$$

where

- $s \equiv_{\text{AC}} (s')^{\pm}$
- $\diamond_2 \in \{>, \geq\} \implies \diamond_1 \in \{>, \geq\}$
- $\diamond_3 = \begin{cases} \diamond_1 \sqcup \diamond_2 & \text{if } \pm = -, \diamond_2 = \geq \in \{>, \geq\} \\ \diamond_1 \sqcup \diamond_2 & \text{else} \end{cases}$
- $L \sqcup R = \text{set}^{-1}(\text{set}(L) \cup \text{set}(R))$,
- $\text{set}(\top) = \mathbb{Q}$
- $\text{set}(\diamond) = \{q \in \mathbb{Q} \mid q \diamond 0\}$
for $\diamond \in \{>, \geq, <, \leq, \approx, \not\approx\}$
- the conclusion is $\top$ if $\diamond_3 = \top$

case $(\diamond_1) = (\diamond_2)$:

case $(\pm) = (+)$: Then $\diamond_3 = \diamond_1$ and $s' \diamond_2 0 \equiv_{\text{AC}} s \diamond_1 0 = s \diamond_3 0$. Thus the conclusion is equivalent and by definition of the multiset ordering it is smaller. $\dashv$

case $(\pm) = (-)$:

case $\diamond_1 = \diamond_2 \in \{(\approx), (\not\approx)\}$: Then $\diamond_3 = \diamond_1$ and $s' \diamond_2 0 \equiv_{\mathcal{A}} s \diamond_1 0 = s \diamond_3 0$. Thus the conclusion is equivalent and by definition of the multiset ordering it is smaller. $\dashv$

case $(\diamond_1) = (\diamond_2) = (>)$: Then $\diamond_3 = \not\approx$, and thus $(s' > 0 \vee s > 0) \equiv_{\mathcal{A}} s \not\approx 0$ where $s^- \equiv_{\text{AC}} s'$. Thus the conclusion is equivalent and by ALASCA Ordering (Def. 2.8) it is smaller. $\dashv$

case $(\diamond_1) = (\diamond_2) = (\geq)$: Then $\diamond_3 = \top$, and obviously $\emptyset \models_{\mathcal{A}} (s' \geq 0 \vee s \geq 0)$ where $s' \equiv_{\mathbf{AC}} s^-$. Thus the conclusion is entailed by the empty set. $\dashv$
case $(\diamond_1) \neq (\diamond_2)$: case $\diamond_1, \diamond_2 \in \{>, \geq\}$: case $(\pm) = (+)$: Then $\diamond_3 = \geq$ obviously the conclusion entails the premise, and it is smaller by the definition of the multiset ordering. $\dashv$ case $(\pm) = (-)$: Then $\diamond_3 = \top$, and obviously $\emptyset \models_{\mathcal{A}} (s' > 0 \vee s \geq 0)$ where $s' \equiv_{\mathbf{AC}} s^-$. Thus the conclusion is entailed by the empty set. $\dashv$
case $\diamond_1, \diamond_2 \in \{>, \not\approx\}$: Then $\diamond_3 = \not\approx$, and obviously the conclusion entails the premise as $\emptyset \models_{\mathcal{A}} x \not\approx 0 \leftrightarrow (x \not\approx 0 \vee x > 0)$, and it is smaller by the definition of the multiset ordering. $\dashv$
case $\diamond_1, \diamond_2 \in \{>, \approx\}$: Then by the side condition of the rule $\diamond_1 = >$ and $\diamond_2 = \approx$. Thus $\diamond_3 = \geq$, and obviously conclusion is equivalent to the premise as $\emptyset \models_{\mathcal{A}} x \geq 0 \leftrightarrow (x > 0 \approx x \approx 0)$, and it is smaller by ALASCA Ordering (Def. 2.9). $\dashv$
case $\diamond_1, \diamond_2 \in \{\geq, \not\approx\}$: Then $\diamond_3 = \top$, and obviously the premise is an $\mathcal{A}$-tautology as $\emptyset \models_{\mathcal{A}} x \geq 0 \vee x \neq 0$, and Cs from the definition of <i>compatible</i> is the empty set. $\dashv$
case $\diamond_1, \diamond_2 \in \{\geq, \approx\}$: Then by the side condition of the rule $\diamond_1 = \geq$ and $\diamond_2 = \approx$. Thus $\diamond_3 = \geq$, and obviously conclusion is equivalent to the premise as $\emptyset \models_{\mathcal{A}} x \geq 0 \leftrightarrow (x \geq 0 \approx x \approx 0)$, and it is smaller by the definition of the multiset ordering. $\dashv$
case $\diamond_1, \diamond_2 \in \{\not\approx, \approx\}$: Then $\diamond_3 = \top$, and obviously the premise is an $\mathcal{A}$-tautology as $\emptyset \models_{\mathcal{A}} x \not\approx 0 \vee x \approx 0$, and Cs from the definition of <i>compatible</i> is the empty set. $\dashv$
□

Lemma 7 (ALASCA-Demodulation). *Dmd is compatible with $ALASCA_{\approx}^{\text{sel}}$.*

Proof. For an inference $\iota = \frac{C_1, \cancel{C_2}}{D}$ we define $\mathbf{sprem}(\iota) = C_1$, $\mathbf{mprem}(\iota) = C_2$, and $\mathbf{concl}(\iota) = D$. Consider any $\iota \in \text{Dmd}$. Then

$$\iota = \frac{k\underline{s} + t \approx 0 \quad \cancel{C[s\sigma]}}{C[-\frac{1}{k}t\sigma]}$$

where

- $(k\underline{s} + t \approx 0)\sigma \prec C[s\sigma]$
- $\forall t_0 \in \mathbf{trms}(t). t_0\sigma \prec s\sigma$
- s is atomic
- $s \notin \mathbf{V}$

Obviously the left premise $\mathbf{sprem}(\iota)$ and the conclusion $\mathbf{concl}(\iota)$ imply the right premise $\mathbf{mprem}(\iota)$.

Let θ be some grounding substitution. By the side condition $\forall t_0 \in \mathbf{trms} t. t_0\sigma \prec s\sigma$, and ALASCA Ordering (Def. 2.3) we know that $\forall t_0 \in \mathbf{trms} t. t_0\sigma\theta \prec s\sigma\theta$. Thus ALASCA Term Ordering (Def. 1.2) we know that $-\frac{1}{k}t\sigma\theta \prec s\sigma\theta$. Therefore further by ALASCA Term Ordering (Def. 1.1) and ALASCA Ordering (Def. 2.2) we know that $C[-\frac{1}{k}t]\theta \prec C[s\sigma]\theta$. $\square$