Ground Joinability and Connectedness in the Superposition Calculus

André Duarte(✉) and Konstantin Korovin(✉)

The University of Manchester, Manchester, UK
{andre.duarte,konstantin.korovin}@manchester.ac.uk

Abstract. Problems in many theories axiomatised by unit equalities (UEQ), such as groups, loops, lattices, and other algebraic structures, are notoriously difficult for automated theorem provers to solve. Consequently, there has been considerable effort over decades in developing techniques to handle these theories, notably in the context of Knuth-Bendix completion and derivatives. The superposition calculus is a generalisation of completion to full first-order logic; however it does not carry over all the refinements that were developed for it, and is therefore not a strict generalisation. This means that (i) as of today, even state of the art provers for first-order logic based on the superposition calculus, while more general, are outperformed in UEQ by provers based on completion, and (ii) the sophisticated techniques developed for completion are not available in any problem which is not in UEQ. In particular, this includes key simplifications such as ground joinability, which have been known for more than 30 years. In fact, all previous completeness proofs for ground joinability rely on proof orderings and proof reductions, which are not easily extensible to general clauses together with redundancy elimination. In this paper we address this limitation and extend superposition with ground joinability, and show that under an adapted notion of redundancy, simplifications based on ground joinability preserve completeness. Another recently explored simplification in completion is connectedness. We extend this notion to "ground connectedness" and show superposition is complete with both connectedness and ground connectedness. We implemented ground joinability and connectedness in a theorem prover, iProver, the former using a novel algorithm which we also present in this paper, and evaluated over the TPTP library with encouraging results.

Keywords: Superposition · Ground joinability · Connectedness · Closure redundancy · First-order theorem proving

1 Introduction

Automated theorem provers based on equational completion [4], such as Waldmeister, MædMax or Twee [13,21,25], routinely outperform superposition-based provers on unit equality problems (UEQ) in competitions such as CASC [22], despite the fact that the superposition calculus was developed as a generalisation

J. Blanchette et al. (Eds.): IJCAR 2022, LNAI 13385, pp. 169–187, 2022.
https://doi.org/10.1007/978-3-031-10769-6_11

of completion to full clausal first-order logic with equality [19]. One of the main ingredients for their good performance is the use of ground joinability criteria for the deletion of redundant equations [1], among other techniques. However, existing proofs of refutational completeness of deduction calculi wrt. these criteria are restricted to unit equalities and rely on proof orderings and proof reductions [1,2,4], which are not easily extensible to general clauses together with redundancy elimination.

Since completion provers perform very poorly (or not at all) on non-UEQ problems (relying at best on incomplete transformations to unit equality [8]), this motivates an attempt to transfer those techniques to the superposition calculus and prove their completeness, so as to combine the generality of the superposition calculus with the powerful simplification rules of completion. To our knowledge, no prover for first-order logic incorporates ground joinability redundancy criteria, except for particular theories such as associativity-commutativity (AC) [20].

For instance, if $f(x, y) \approx f(y, x)$ is an axiom, then the equation $f(x, f(y, z)) \approx f(x, f(z, y))$ is redundant, but this cannot be justified by any simplificaton rule in the superposition calculus. On the other hand, a completion prover which implements ground joinability can easily delete the latter equation wrt. the former. We show that ground joinability can be enabled in the superposition calculus without compromising completeness.

As another example, the simplification rule in completion can use $f(x) \approx s$ (when $f(x) \succ s$) to rewrite $f(a) \approx t$ regardless of how s and t compare, while the corresponding demodulation rule in superposition can only rewrite if $s \prec t$. Our "encompassment demodulation" rule matches the former, while also being complete in the superposition calculus.

In [11] we introduced a novel theoretical framework for proving completeness of the superposition calculus, based on an extension of Bachmair-Ganzinger model construction [5], together with a new notion of redundancy called "closure redundancy". We used it to prove that certain AC joinability criteria, long used in the context of completion [1], could also be incorporated in the superposition calculus for full first-order logic while preserving completeness.

In this paper, we extend this framework to show the completeness of the superposition calculus extended with: (i) a general ground joinability simplification rule, (ii) an improved encompassment demodulation simplification rule, (iii) a connectedness simplification rule extending [3,21], and (iv) a new ground connectedness simplification rule. The proof of completeness that enables these extensions is based on a new encompassment closure ordering. In practice, these extensions help superposition to be competitive with completion in UEQ problems, and improves the performance on non-UEQ problems, which currently do not benefit from these techniques at all.

We also present a novel incremental algorithm to check ground joinability, which is very efficient in practice; this is important since ground joinability can be an expensive criterion to test. Finally, we discuss some of the experimental results we obtained after implementing these techniques in iProver [10,16].

The paper is structured as follows. In Sect. 2 we define some basic notions to be used throughout the paper. In Sect. 3 we define the closure ordering we use to

prove redundancies. In Sect. 4 we present redundancy criteria for demodulation, ground joinability, connectedness, and ground connectedness. We prove their completeness in the superposition calculus, and discuss a concrete algorithm for checking ground joinability, and how it may improve on the algorithms used in e.g. Waldmeister [13] or Twee [21]. In Sect. 5 we discuss experimental results.

2 Preliminaries

We consider a signature consisting of a finite set of function symbols and the equality predicate as the only predicate symbol. We fix a countably infinite set of variables. First-order *terms* are defined in the usual manner. Terms without variables are called *ground* terms. A *literal* is an unordered pair of terms with either positive or negative polarity, written $s \approx t$ and $s \not\approx t$ respectively (we write $s \mathrel{\dot{\approx}} t$ to mean either of the former two). A *clause* is a multiset of literals. Collectively terms, literals, and clauses will be called *expressions*.

A *substitution* is a mapping from variables to terms which is the identity for all but finitely many variables. An injective substitution onto variables is called a *renaming*. If e is an expression, we denote application of a substitution σ by $e\sigma$, replacing all variables with their image in σ. Let $\mathrm{GSubs}(e) = \{\sigma \mid e\sigma \text{ is ground}\}$ be the set of *ground substitutions* for e. Overloading this notation for sets we write $\mathrm{GSubs}(E) = \{\sigma \mid \forall e \in E.\ e\sigma \text{ is ground}\}$. Finally, we write e.g. $\mathrm{GSubs}(e_1, e_2)$ instead of $\mathrm{GSubs}(\{e_1, e_2\})$. The identity substitution is denoted by ϵ.

A substitution θ is *more general* than σ if $\theta\rho = \sigma$ for some substitution ρ which is not a renaming. If s and t can be *unified*, that is, if there exists σ such that $s\sigma = t\sigma$, then there also exists the *most general unifier*, written $\mathrm{mgu}(s, t)$. A term s is said to be *more general* than t if there exists a substitution θ that makes $s\theta = t$ but there is no substitution σ such that $t\sigma = s$. Two terms s and t are said to be *equal modulo renaming* if there exist injective θ, σ such that $s\theta = t$ and $t\sigma = s$. The relations "less general than", "equal modulo renaming", and their union are represented respectively by the symbols $\sqsupset$, $\equiv$, and $\sqsupseteq$.

A more refined notion of instance is that of *closure* [6]. Closures are pairs $e \cdot \sigma$ that are said to *represent* the expression $e\sigma$ while retaining information about the original term and its instantiation. Closures where $e\sigma$ is ground are said to be *ground closures*. Let $\mathrm{GClos}(e) = \{e \cdot \sigma \mid e\sigma \text{ is ground}\}$ be the set of ground closures of e. Overloading the notation for sets, if N is a set of clauses then $\mathrm{GClos}(N) = \bigcup_{C \in N} \mathrm{GClos}(C)$.

We write $s[t]$ if t is a *subterm* of s. If also $s \neq t$, then it is a *strict subterm*. We denote these relations by $s \trianglerighteq t$ and $s \triangleright t$ respectively. We write $s[t \mapsto t']$ to denote the term obtained from s by replacing all occurrences of t by t'.

A (strict) partial order is a binary relation which is transitive ($a \succ b \succ c \Rightarrow a \succ c$), irreflexive ($a \not\succ a$), and asymmetric ($a \succ b \Rightarrow b \not\succ a$). A (non-strict) partial preorder (or quasiorder) is any transitive, reflexive relation. A (pre)order is total over X if $\forall x, y \in X.\ x \succeq y \vee y \succeq x$. Whenever a non-strict (pre)order $\succeq$ is given, the induced equivalence relation $\sim$ is $\succeq \cap \preceq$, and the induced strict pre(order) $\succ$ is $\succeq \setminus \sim$. The *transitive closure* of a relation $\succ$, the smallest transitive

relation that contains $\succ$, is denoted by $\succ^+$. A *transitive reduction* of a relation $\succ$, the smallest relation whose transitive closure is $\succ$, is denoted by $\succ^-$.

For an ordering $\succ$ over a set X, its *multiset extension* $\succcurlyeq$ over multisets of X is given by: $A \succcurlyeq B$ iff $A \neq B$ and $\forall x \in B.\ B(x) > A(x)\ \exists y \in A.\ y \succ x \wedge A(y) > B(y)$, where $A(x)$ is the number of occurrences of element x in multiset A (we also use $\succcurlyeq\!\!\succcurlyeq$ for the the multiset extension of $\succcurlyeq$). It is well known that the mutltiset extension of a well-founded/total order is also a well-founded/total order, respectively [9]. The *(n-fold) lexicographic extension* of $\succ$ over X is denoted $\succ_{\text{lex}}$ over ordered n-tuples of X, and is given by $\langle x_1, \ldots, x_n\rangle \succ_{\text{lex}} \langle y_1, \ldots, y_n\rangle$ iff $\exists i.\ x_1 = y_1 \wedge \cdots \wedge x_{i-1} = y_{i-1} \wedge x_i \succ y_i$. The lexicographic extension of a well-founded/total order is also a well-founded/total order, respectively.

A binary relation $\rightarrow$ over the set of terms is a *rewrite relation* if (i) $l \rightarrow r \Rightarrow l\sigma \rightarrow r\sigma$ and (ii) $l \rightarrow r \Rightarrow s[l] \rightarrow s[l \mapsto r]$. The *reflexive-transitive closure* of a relation is the smallest reflexive-transitive relation which contains it. It is denoted by $\overset{*}{\rightarrow}$. Two terms are *joinable* $(s \downarrow t)$ if $s \overset{*}{\rightarrow} u \overset{*}{\leftarrow} t$.

If a rewrite relation is also a strict ordering, then it is a *rewrite ordering*. A *reduction ordering* is a rewrite ordering which is well-founded. In this paper we consider reduction orderings which are total on ground terms, such orderings are also *simplification orderings* i.e., satisfy $s \rhd t \Rightarrow s \succ t$.

3 Ordering

In [11] we presented a novel proof of completeness of the superposition calculus based on the notion of closure redundancy, which enables the completeness of stronger redundancy criteria to be shown, including AC normalisation, AC joinability, and encompassment demodulation. In this paper we use a slightly different closure ordering ($\succ_{cc}$), in order to extract better completeness conditions for the redundancy criteria that we present in this paper (the definition of closure redundant clause and closure redundant inference is parametrised by this $\succ_{cc}$).

Let $\succ_t$ be a simplification ordering which is total on ground terms. We extend this first to an ordering on ground term closures, then to an ordering on ground clause closures. Let

$$s \cdot \sigma \succ_{tc'} t \cdot \rho \qquad \text{iff} \qquad \begin{array}{l} \text{either } s\sigma \succ_t t\rho \\ \text{or else } s\sigma = t\rho \text{ and } s \sqsupset t, \end{array} \tag{1}$$

where $s\sigma$ and $t\rho$ are ground, and let $\succ_{tc}$ be an (arbitrary) total well-founded extension of $\succ_{tc'}$. We extend this to an ordering on clause closures. First let

$$M_{lc}((s \approx t) \cdot \theta) = \{s\theta \cdot \epsilon, t\theta \cdot \epsilon\}, \tag{2}$$

$$M_{lc}((s \not\approx t) \cdot \theta) = \{s\theta \cdot \epsilon, t\theta \cdot \epsilon, s\theta \cdot \epsilon, t\theta \cdot \epsilon\}, \tag{3}$$

and let M_{cc} be defined as follows, depending on whether the clause is unit or non-unit:

$$M_{cc}(\emptyset \cdot \theta) = \emptyset, \tag{4}$$

$$M_{cc}((s \approx t) \cdot \theta) = \{\{s \cdot \theta\}, \{t \cdot \theta\}\}, \tag{5}$$

$$M_{cc}((s \not\approx t) \cdot \theta) = \{\{s \cdot \theta, t \cdot \theta, s\theta \cdot \epsilon, t\theta \cdot \epsilon\}\}, \tag{6}$$

$$M_{cc}((s \mathrel{\dot{\approx}} t \vee \cdots) \cdot \theta) = \{M_{lc}(L \cdot \theta) \mid L \in (s \mathrel{\dot{\approx}} t \vee \cdots)\}, \tag{7}$$

then $\succ_{cc}$ is defined by

$$C \cdot \sigma \succ_{cc} D \cdot \rho \qquad \text{iff} \quad M_{cc}(C \cdot \sigma) \gg_{tc} M_{cc}(D \cdot \rho). \tag{8}$$

The main purpose of this definition is twofold: (i) that when $s\theta \succ_t t\theta$ and u occurs in a clause D, then $s\theta \lhd u$ or $s \sqsubset s\theta = u$ implies $(s \approx t) \cdot \theta\rho \prec_{cc} D \cdot \rho$, and (ii) that when C is a positive unit clause, D is not, s is the maximal subterm in $C\theta$ and t is the maximal subterm in $D\sigma$, then $s \succeq_t t$ implies $C \cdot \theta \prec_{cc} D \cdot \sigma$. These two properties enable unconditional rewrites via oriented unit equations on positive unit clauses to succeed whenever they would also succeed in unfailing completion [4], and rewrites on negative unit and non-unit clauses to always succeed. This will enable us to prove the correctness of the simplification rules presented in the following section.

4 Redundancies

In this section we present several redundancy criteria for the superposition calculus and prove their completeness. Recall the definitions in [11]: a clause C is redundant in a set S if all its ground closures $C \cdot \theta$ follow from closures in $\mathrm{GClos}(S)$ which are smaller wrt. $\succ_{cc}$; an inference $C_1, \ldots, C_n \vdash D$ is redundant in a set S if, for all $\theta \in \mathrm{GSubs}(C_1, \ldots, C_n, D)$ such that $C_1\theta, \ldots, C_n\theta \vdash D\theta$ is a valid inference, the closure $D \cdot \theta$ follows from closures in $\mathrm{GClos}(S)$ such that each is smaller than some $C_1 \cdot \theta, \ldots, C_n \cdot \theta$. These definitions (in terms of ground closures rather than in terms of ground clauses, as in [19]) arise because they enable us to justify stronger redundancy criteria for application in superposition theorem provers, including the AC criteria developed in [11] and the criteria in this section.

Theorem 1. The superposition calculus [19] is refutationally complete wrt. closure redundancy, that is, if a set of clauses is saturated up to closure redundancy (meaning any inference with non-redundant premises in the set is redundant) and does not contain the empty clause, then it is satisfiable.

Proof. The proof of completeness of the superposition calculus wrt. this closure ordering carries over from [11] with some modifications, which are presented in a full version of this paper [12].

4.1 Encompassment Demodulation

We introduce the following definition, to be re-used throughout the paper.

Definition 1. A rewrite via $l \approx r$ in clause $C[l\theta]$ is *admissible* if one of the following conditions holds: (i) C is not a positive unit, or (let $C = s[l\theta] \approx t$ for some θ) (ii) $l\theta \neq s$, or (iii) $l\theta \sqsupset l$, or (iv) $s \prec_t t$, or (v) $r\theta \prec_t t$.[1]

[1] We note that (iv) is superfluous, but we include it since in practice it is easier to check, as it is local to the clause being rewritten and therefore needs to be checked only once, while (v) needs to be checked with each demodulation attempt.

We then have

$$\begin{array}{ll} \text{Encompassment} \\ \text{Demodulation} \end{array} \quad \frac{l \approx r \quad C[l\theta]}{C[l\theta \mapsto r\theta]}, \quad \begin{array}{l} \text{where } l\theta \succ_t r\theta, \text{ and} \\ \text{rewrite via } l \approx r \text{ in } C \text{ is admissible.} \end{array} \tag{9}$$

In other words, given an equation $l \approx r$, if an instance $l\theta$ is a subterm in C, then the rewrite is admissible (meaning, for example, that an unconditional rewrite is allowed when $l\theta \succ_t r\theta$) if C is not a positive unit, or if $l\theta$ occurs at a strict subterm position, or if $l\theta$ is less general than l, or if $l\theta$ occurs outside a maximal side, or if $r\theta$ is smaller than the other side. This restriction is much weaker than the one given for the usual demodulation rule in superposition [17], and equivalent to the one in equational completion when we restrict ourselves to unit equalities [4].

Example 1. If $f(x) \succ_t s$, we can use $f(x) \approx s$ to rewrite $f(x) \approx t$ when $s \prec_t t$, and $f(a) \approx t$, $f(x) \not\approx t$, or $f(x) \approx t \vee C$ regardless of how s and t compare.

4.2 General Ground Joinability

In [11] we developed redundancy criteria for the theory of AC functions in the superposition calculus. In this section we extend these techniques to develop redundancy criteria for ground joinability in arbitrary equational theories.

Definition 2. Two terms are *strongly joinable* ($s \lessdot t$), in a clause C wrt. a set of equations S, if either $s = t$, or $s \to s[l_1\sigma_1 \mapsto r_1\sigma_1] \xrightarrow{*} t$ via rules $l_i \approx r_i \in S$, where the rewrite via $l_1 \approx r_1$ is admissible in C, or $s \to s[l_1\sigma_1 \mapsto r_1\sigma_1] \downarrow t[l_2\sigma_2 \mapsto r_2\sigma_2] \leftarrow t$ via rules $l_i \approx r_i \in S$, where the rewrites via $l_1 \approx r_1$ and $l_2 \approx r_2$ are admissible in C. To make the ordering explicit, we may write $s \lessdot_\succ t$. Two terms are *strongly ground joinable* ($s \lessdot\!\!\lessdot t$), in a clause C wrt. a set of equations S, if for all $\theta \in \mathrm{GSubs}(s,t)$ we have $s\theta \lessdot t\theta$ in C wrt. S.

We then have:

$$\text{Ground joinability} \quad \frac{s \approx t \vee C \quad S}{}, \quad \text{where } s \lessdot\!\!\lessdot t \text{ in } s \approx t \vee C \text{ wrt. } S, \tag{10a}$$

$$\text{Ground joinability} \quad \frac{s \not\approx t \vee C \quad S}{C}, \quad \text{where } s \lessdot\!\!\lessdot t \text{ in } s \not\approx t \vee C \text{ wrt. } S. \tag{10b}$$

Theorem 2. Ground joinability is a sound and admissible redundancy criterion of the superposition calculus wrt. closure redundancy.

Proof. We will show the positive case first. If $s \lessdot\!\!\lessdot t$, then for any instance $(s \approx t \vee C) \cdot \theta$ we either have $s\theta = t\theta$, and therefore $\emptyset \models (s \approx t) \cdot \theta$, or we have wlog. $s\theta \succ_t t\theta$, with $s\theta \downarrow t\theta$. Then $s\theta$ and $t\theta$ can be rewritten to the same normal form u by $l_i\sigma_i \to r_i\sigma_i$ where $l_i \approx r_i \in S$. Since $u \prec_t s\theta$ and $u \succeq_t t\theta$, then $(s \approx t \vee C) \cdot \theta$

follows from smaller $(u \approx u \vee C) \cdot \theta$[2] (a tautology, i.e. follows from $\emptyset$) and from the instances of clauses in S used to rewrite $s\theta \to u \leftarrow t\theta$. It only remains to show that these latter instances are also smaller than $(s \approx t \vee C) \cdot \theta$. Since we have assumed $s\theta \succ_t t\theta$, then at least one rewrite step must be done on $s\theta$. Let $l_1\sigma_1 \to r_1\sigma_1$ be the instance of the rule used for that step, with $(l_1 \approx r_1) \cdot \sigma_1$ the closure that generates it. By Definition 1 and 2, one of the following holds:

- $C \neq \emptyset$, therefore $(l_1 \approx r_1) \cdot \sigma_1 \prec_{cc} (s \approx t \vee C) \cdot \theta$, or
- $l_1\sigma_1 \lhd s\theta$, therefore $l_1\sigma_1 \prec_t s\theta \Rightarrow l_1 \cdot \sigma_1 \prec_{tc} s \cdot \theta \Rightarrow (l_1 \approx r_1) \cdot \sigma_1 \prec_{cc} (s \approx t) \cdot \theta$, or
- $l_1\sigma_1 = s\theta$ and $s \sqsupset l_1$, therefore $l_1 \cdot \sigma_1 \prec_{tc} s \cdot \theta \Rightarrow (l_1 \approx r_1) \cdot \sigma_1 \prec_{cc} (s \approx t) \cdot \theta$, or
- $l_1\sigma_1 = s\theta$ and $s \equiv l_1$ and $r_1\sigma_1 \prec_t t\theta$, therefore $r_1 \cdot \sigma_1 \prec_{tc} t \cdot \theta \Rightarrow (l_1 \approx r_1) \cdot \sigma_1 \prec_{cc} (s \approx t) \cdot \theta$.

As for the remaining steps, they are done on the smaller side $t\theta$ or on the other side after this first rewrite, which is smaller than $s\theta$. Therefore all subsequent steps done by any $l_j\sigma_j \to r_j\sigma_j$ will have $r_j \cdot \sigma_j \prec_{tc} l_j \cdot \sigma_j \prec_{tc} s \cdot \theta \Rightarrow (l_j \approx r_j) \cdot \sigma_j \prec_{cc} (s \approx t \vee C) \cdot \theta$. As such, since this holds for all ground closures $(s \approx t \vee C) \cdot \theta$, then $s \approx t \vee C$ is redundant wrt. S.

For the negative case, the proof is similar. We will conclude that $(s \not\approx t \vee C) \cdot \theta$ follows from smaller $(l_i \approx r_i) \cdot \sigma_i \in \mathrm{GClos}(S)$ and smaller $(u \not\approx u \vee C) \cdot \theta$. The latter, of course, follows from smaller $C \cdot \theta$, therefore $s \not\approx t \vee C$ is redundant wrt. $S \cup \{C\}$. □

Example 2. If $S = \{f(x,y) \approx f(y,x)\}$, then $f(x, f(y,z)) \approx f(x, f(z,y))$ is redundant wrt. S. Note that $f(x,y) \approx f(y,x)$ is not orientable by any simplification ordering, therefore this cannot be justified by demodulation alone.

Testing for Ground Joinability. The general criterion presented above begs the question of how to test, in practice, whether $s \not\downarrow t$ in a clause $s \not\approx t \vee C$. Several such algorithms have been proposed [1,18,21]. All of these are based on the observation that if we consider all total preorders $\succeq_v$ on $\mathrm{Vars}(s,t)$ and for all of them show strong joinability with a modified ordering—which we denote $\succ_{t[v]}$—then we have shown strong *ground* joinability in the order $\succ_t$ [18].

Definition 3. A simplification order on terms $\succ_t$ *extended with* a preorder on variables $\succeq_v$, denoted $\succeq_{t[v]}$, is a simplification preorder (i.e. satisfies all the relevant properties in Sect. 2) such that $\succeq_{t[v]} \supseteq \succ_t \cup \succeq_v$.

Example 3. If $x \succ_v y$, then $g(x) \succ_{t[v]} g(y)$, $g(x) \succ_{t[v]} y$, $f(x,y) \succ_{t[v]} f(y,x)$, etc.

The simplest algorithm based on this approach would be to enumerate all possible total preorders $\succeq_v$ over $\mathrm{Vars}(s,t)$, and exhaustively reduce both sides

[2] Wlog. $u\theta = u$, renaming variables in u if necessary.

via equations in S orientable by $\succ_{t[v]}$, checking if the terms can be reduced to the same normal form for all total preorders. This is very inefficient since there are $\mathcal{O}(n!e^n)$ such total preorders [7], where n is the cardinality of $\text{Vars}(s,t)$. Another approach is to consider only a smaller number of partial preorders, based on the obvious fact that $s \not\downarrow_{\succ_{t[v]}} t \Rightarrow \forall \succeq'_v \supseteq \succeq_v .\ s \not\downarrow_{\succ_{t[v']}} t$, so that joinability under a smaller number of partial preorders can imply joinability under all the total preorders, necessary to prove ground joinability.

However, this poses the question of how to choose which partial preorders to check. Intuitively, for performance, we would like that whenever the two terms are *not* ground joinable, that some total preorder where they are not joinable is found as early as possible, and that whenever the two terms *are* joinable, that all total preorders are covered in as few partial preorders as possible.

Example 4. Let $S = \{f(x, f(y,z)) \approx f(y, f(x,z))\}$. Then $f(x, f(y, f(z, f(w,u)))) \approx f(x, f(y, f(w, f(z,u))))$ can be shown to be ground joinable wrt. S by checking just three cases: $\succeq_v \in \{z \succ w\ ,\ z \sim w\ ,\ z \prec w\}$, even though there are 6942 possible preorders.

Waldmeister first tries all partial preorders relating two variables among $\text{Vars}(s,t)$, then three, etc. until success, failure (by trying a total order and failing to join) or reaching a predefined limit of attempts [1]. Twee tries an arbitrary total strict order, then tries to weaken it, and repeats until all total preorders are covered [21]. We propose a novel algorithm—incremental ground joinability—whose main improvement is *guiding* the process of picking which preorders to check by finding, during the process of searching for rewrites on subterms of the terms we are attempting to join, minimal extensions of the term order with a variable preorder which allow the rewrite to be done in the $\succ$ direction.

Our algorithm is summarised as follows. We start with an empty queue of variable preorders, V, initially containing only the empty preorder. Then, while V is not empty, we pop a preorder $\succeq_v$ from the queue, and attempt to perform a rewrite via an equation which is newly orientable by some extension $\succeq'_v$ of $\succeq_v$. That is, during the process of finding generalisations of a subterm of s or t among left-hand sides of candidate unoriented unit equations $l \approx r$, when we check that the instance $l\theta \approx r\theta$ used to rewrite is oriented, we try to force this to be true under some minimal extension $\succ_{t[v']}$ of $\succ_{t[v]}$, if possible. If no such rewrite exists, the two terms are not strongly joinable under $\succ_{t[v]}$ or any extension, and so are not strongly ground joinable and we are done. If it exists, we exhaustively rewrite with $\succ_{t[v']}$, and check if we obtain the same normal form. If we do not obtain it yet, we repeat the process of searching rewrites via equations orientable by further extensions of the preorder. But if we do, then we have proven joinability in the extended preorder; now we must add back to the queue a set of preorders O such that all the total preorders which are $\supseteq \succeq_v$ (popped from the queue) but not $\supseteq \succeq'_v$ (minimal extension under which we have proven joinability) are $\supseteq$ of some $\succeq''_v \in O$ (pushed back into the queue to be checked). Obtaining this O is implemented by order_diff$(\succeq_v, \succeq'_v)$, defined below. Whenever there are no more preorders in the queue to check, then we have checked that the terms are strongly joinable under all possible total preorders, and we are done.

Together with this, some book-keeping for keeping track of completeness conditions is necessary. We know that for completeness to be guaranteed, the conditions in Definition 1 must hold. They automatically do if C is not a positive unit or if the rewrite happens on a strict subterm. We also know that after a term has been rewritten at least once, rewrites on that side are always complete (since it was rewritten to a smaller term). Therefore we store in the queue, together with the preorder, a flag in $\mathcal{P}(\{\mathsf{L},\mathsf{R}\})$ indicating on which sides does a top rewrite need to be checked for completeness. Initially the flag is $\{\mathsf{L}\}$ if $s \succ_t t$, $\{\mathsf{R}\}$ if $s \prec_t t$, $\{\mathsf{L},\mathsf{R}\}$ if s and t are incomparable, and $\{\}$ if the clause is not a positive unit. When a rewrite at the top is attempted (say, $l \approx r$ used to rewrite $s = l\theta$ with t being the other side), if the flag for that side is set, then we check if $l\theta \sqsupset l$ or $r\theta \prec t$. If this fails, the rewrite is rejected. Whenever a side is rewritten (at any position), the flag for that side is cleared.

The definition of order_diff is as follows. Let the transitive reduction of $\succeq$ be represented by a set of links of the form $x \succ y$ / $x \sim y$.

$$\text{order\_diff}(\succeq_1, \succeq_2) = \{\succeq^+ \mid \succeq \in \text{order\_diff}'(\succeq_1, \succeq_2^-)\}\,, \tag{11a}$$

$$\text{order\_diff}'(\succeq_1, \succeq_2^-) = \tag{11b}$$

$$\begin{cases} \succeq_2^- = \{x \succ y\} \uplus \succeq_2^{-\prime} \;\Rightarrow \begin{cases} x \succ_1 y \;\Rightarrow\; \text{order\_diff}'(\succeq_1, \succeq_2^{-\prime}) \\ x \not\succ_1 y \;\Rightarrow\; \{\succeq_1 \cup \{y \succ x\}\,,\, \succeq_1 \cup \{x \sim y\}\} \\ \qquad\qquad \cup\, \text{order\_diff}'(\succeq_1 \cup \{x \succ y\}, \succeq_2^{-\prime}) \end{cases} \\ \succeq_2^- = \{x \sim y\} \uplus \succeq_2^{-\prime} \;\Rightarrow \begin{cases} x \sim_1 y \;\Rightarrow\; \text{order\_diff}'(\succeq_1, \succeq_2^{-\prime}) \\ x \not\sim_1 y \;\Rightarrow\; \{\succeq_1 \cup \{x \succ y\}\,,\, \succeq_1 \cup \{y \succ x\}\} \\ \qquad\qquad \cup\, \text{order\_diff}'(\succeq_1 \cup \{x \sim y\}, \succeq_2^{-\prime}) \end{cases} \\ \succeq_2^- = \emptyset \;\Rightarrow\; \emptyset\,. \end{cases}$$

where $\succeq_1 \subseteq \succeq_2$. In other words, we take a transitive reduction of $\succeq_2$, and for all links ℓ in that reduction which are not part of $\succeq_1$, we return orders $\succeq_1$ augmented with the reverse of ℓ and recurse with $\succeq_1 = \succeq_1 \cup \ell$.

Example 5.

$\succeq_1$	$\succeq_2$	order_diff$(\succeq_1, \succeq_2)$
$x \succ y$	$x \succ y \succ z \succ w$	$x \succ y \sim z\,,\, x \succ y \prec z\,,\, x \succ y \succ z \sim w\,,\, x \succ y \succ z \prec w$
$y \prec x \succ z$	$x \succ y \succ z$	$x \succ y \sim z\,,\, x \succ z \succ y$

Theorem 3. For all total $\succeq_v^T \supseteq \succeq_1$, there exists one and only one $\succeq_i \in \{\succeq_2\} \cup \text{order\_diff}(\succeq_1, \succeq_2)$ such that $\succeq_v^T \supseteq \succeq_i$. For all $\succeq_v^T \not\supseteq \succeq_1$, there is no $\succeq_i \in \{\succeq_2\} \cup \text{order\_diff}(\succeq_1, \succeq_2)$ such that $\succeq_v^T \supseteq \succeq_i$.

Proof. See full version of the paper [12].

An algorithm based on searching for rewrites in minimal extensions of a variable preorder (starting with minimal extensions of the bare term ordering, $\succ_{t[\emptyset]}$), has several advantages. The main benefit of this approach is that, instead of imposing an a priori ordering on variables and then checking joinability under that ordering, we instead build a minimal ordering *while* searching for candidate unit equations to rewrite subterms of s, t. For instance, if two terms are *not* ground joinable, or not even rewritable in any $\succ_{t[v]}$ where it was not rewritable in $\succ_t$, then an approach such as the one used in Avenhaus, Hillenbrand and Löchner [1] cannot detect this until it has extended the preorder arbitrarily to a total ordering, while our incremental algorithm immediately realises this. We should note that empirically this is what happens in most cases: most of the literals we check during a run are *not* ground joinable, so for practical performance it is essential to optimise this case.

Theorem 4. Algorithm 1 returns "Success" only if $s \not\Updownarrow t$ in C wrt. S.[3]

Proof. We will show that Algorithm 1 returns "Success" if and only if $s \not\downarrow_{\succ_{t[v^T]}} t$ for all total $\succeq_v^T$ over $\mathrm{Vars}(s,t)$, which implies $s \not\Updownarrow_{\succ_t} t$.

When $\langle \succeq_v, s, t, c\rangle$ is popped from V, we exhaustively reduce s, t via equations in S oriented wrt. $\succ_{t[v]}$, obtaining s^r, t^r. If $s^r \sim_{t[v]} t^r$, then $s \not\downarrow_{\succ_{t[v]}} t$, and so $s \not\downarrow_{\succ_{t[v^T]}} t$ for all total $\succeq_v^T \supseteq \succeq_v$. If $s^r \not\sim_{t[v]} t^r$, we will attempt to rewrite one of s^r, t^r using *some* extended $\succ_{t[v']}$ where $\succeq'_v \supset \succeq_v$. If this is impossible, then $s \downarrow\!\!\!\!\!\!\not\;\;_{\succ_{t[v']}} t$ for any $\succeq'_v \supseteq \succeq_v$, and therefore there exists at least one total $\succeq_v^T$ such that $s \downarrow\!\!\!\!\!\!\not\;\;_{\succeq_v^T} t$, and we return "Fail".

If this is possible, then we repeat the process: we exhaustively reduce wrt. $\succ_{t[v']}$, obtaining s', t'. If $s' \not\sim_{t[v']} t'$, then we start again the process from the step where we attempt to rewrite via an extension of $\succeq'_v$: we either find a rewrite with some $\succ_{t[v'']}$ with $\succeq''_v \supset \succeq'_v$, and exhaustively normalise wrt. $\succ_{t[v'']}$ obtaining s'', t'', etc., or we fail to do so and return "Fail".

If in any such step (after exhaustively normalising wrt. $\succ_{t[v']}$) we find $s' \sim_{t[v']} t'$, then $s \not\downarrow_{\succ_{t[v']}} t$, and so $s \not\downarrow_{\succ_{t[v^T]}} t$ for all total $\succeq_v^T \supseteq \succeq'_v$. Now at this point we must add back to the queue a set of preorders $\succeq''_{v\,i}$ such that: for all total $\succeq_v^T \supseteq \succeq_v$, either $\succeq_v^T \supseteq \succeq'_v$ (proven to be $\not\downarrow$) or $\succeq_v^T \supseteq$ some $\succeq''_{v\,i}$ (added to V to be checked). For efficiency, we would also like for there to be no overlap: no total $\succeq_v^T \supseteq \succeq_v$ is an extension of more than one of $\{\succeq'_v, \succeq''_{v\,1}, \ldots\}$.

This is true because of Theorem 3. So we add $\{\langle \succeq''_{v\,i}, s^r, t^r, c^r\rangle \mid \succeq''_{v\,i} \in \mathrm{order\_diff}(\succeq_v, \succeq'_v)\}$ to V, where $c^r = c \setminus (\text{if } s^r \neq s \text{ then } \{\mathsf{L}\} \text{ else } \{\}) \setminus (\text{if } t^r \neq t \text{ then } \{\mathsf{R}\} \text{ else } \{\})$. Note also that $s \not\downarrow_{\succ_{t[v]}} s^r$ and $t \not\downarrow_{\succ_{t[v]}} t^r$, therefore also $s \not\downarrow_{\succ_{t[v\,i''_1]}} s^r$ and $t \not\downarrow_{\succ_{t[v\,i''_1]}} t^r$ if $\succeq''_{v\,i} \supset \succeq_v$.

[3] Note that the other direction may not always hold, there are strongly ground joinable terms which are not detected by this method of analysing all preorders between variables, e.g. $f(x, g(y)) \not\Updownarrow f(g(y), x)$ wrt. $S = \{f(x, y) \approx f(y, x)\}$.

Algorithm 1: Incremental ground joinability test

Input: literal $s \mathrel{\dot{\approx}} t \in C$; set of unorientable equations S
Output: whether $s \lessgtr t$ in C wrt. S
begin
  $c \leftarrow \emptyset$ if C is not pos. unit, $\{\mathsf{L}\}$ if $s \succ t$, $\{\mathsf{R}\}$ if $s \prec t$, $\{\mathsf{L},\mathsf{R}\}$ otherwise
  $V \leftarrow \{\langle \emptyset, s, t, c\rangle\}$
  while V is not empty **do**
    $\langle \succeq_v, s, t, c\rangle \leftarrow$ pop from V
    $s, t \leftarrow$ normalise s, t wrt. $\succ_{t[v]}$, with completeness flag c
    $c \leftarrow c \setminus (\{\mathsf{L}\}$ if s was changed$) \setminus (\{\mathsf{R}\}$ if t was changed$)$
    if $s \sim_{t[v]} t$ **then**
      continue
    else
      $s', t', c' \leftarrow s, t, c$
      while there exists $l \approx r \in S$ that can rewrite s' or t' wrt. some $\succeq'_v \supset \succeq_v$, with completeness flag c **do**
        $s', t' \leftarrow$ normalise s', t' wrt. $\succ_{t[v']}$, with completeness flag c
        $c' \leftarrow c' \setminus (\{\mathsf{L}\}$ if s' was changed$) \setminus (\{\mathsf{R}\}$ if t' was changed$)$
        if $s' \sim_{t[v']} t'$ **then**
          for $\succeq''_v$ in order_diff$(\succeq_v, \succeq'_v)$ **do** push $\langle \succeq''_v, s, t, c\rangle$ to V
          break
        end
        $\succeq_v \leftarrow \succeq'_v$
      else
        return Fail
      end
    end
  else
    return Success
  end
end
where rewriting u in s, t wrt. $\succ$ with completeness flag c succeeds **if**
  (i) u is a strict subterm of s or t,
  (ii) $u = s$ with $\mathsf{L} \notin c$,
  (iii) $u = t$ with $\mathsf{R} \notin c$,
  (iv) instance $l\sigma \approx r\sigma$ used to rewrite has $l \sqsubset u$,
  (v) $u = s$ with $r\sigma \prec t$,
  (vi) or $u = t$ with $r\sigma \prec s$.
end

During this whole process, any rewrites must pass a completeness test mentioned previously, such that the conditions in the definition of $\lessgtr$ hold. Let s_0, t_0 be the original terms and s, t be the ones being rewritten and c the completeness flag. If the rewrite is at a strict subterm position, it succeeds by Definition 2. If the rewrite is at the top, then we check c. If L is unset ($\mathsf{L} \notin c$), then either $s \succeq s_0 \prec t_0$ or $s \prec s_0$ or the clause is not a positive unit, so we allow a rewrite at the top of s, again by Definition 2. If L is set ($\mathsf{L} \in c$), then an explicit check

must be done: we allow a rewrite at the top of s ($= s_0$) iff it is done by $l\sigma \to r\sigma$ with $l\sigma \sqsupset l$ or $r\sigma \prec t_0$. Respectively for $\mathtt{R}$, with the roles of s and t swapped.

In short, we have shown that if $\langle \succeq_v, s', t', c' \rangle$ is popped from V, then V is only ever empty, and so the algorithm only terminates with "Success", if $s' \not\downarrow_{\succ_{t[v^T]}} t'$ for all total $\succeq_v^T \supseteq \succeq_v$. Since V is initialised with $\langle \emptyset, s, t, c \rangle$, then the algorithm only returns "Success" if $s \not\downarrow_{\succ_{t[v^T]}} t$ for all total $\succeq_v^T$. □

Orienting via Extension of Variable Ordering. In order to apply the ground joinability algorithm we need a way to check, for a given $\succ_t$ and $\succeq_v$ and some s, t, whether there exists a $\succeq'_v \supset \succeq_v$ such that $s \succ_{t[v']} t$. Here we show how to do this when $\succ_t$ is a Knuth-Bendix Ordering (KBO) [15].

Recall the definition of KBO. Let $\succ_s$ be a partial order on symbols, w be an $\mathbb{N}$-valued weight function on symbols and variables, with the property that $\exists m\ \forall x \in \mathcal{V}.\ w(x) = m$, $w(c) \geq m$ for all constants c, and there may only exist one unary symbol f with $w(f) = 0$ and in this case $f \succ_s g$ for all other symbols g. For terms, their weight is $w(f(s_1, \dots)) = w(f) + w(s_1) + \cdots$. Let also $|s|_x$ be the number of occurrences of x in s. Then

$$f(s_1,\dots) \succ_{\mathrm{KBO}} g(t_1,\dots) \quad \text{iff} \quad \begin{cases} \text{either } w(f(s_1,\dots)) > w(g(t_1,\dots)), \\ \text{or } w(f(s_1,\dots)) = w(g(t_1,\dots)) \\ \quad \text{and } f \succ_s g, \\ \text{or } w(f(s_1,\dots)) = w(g(t_1,\dots)) \\ \quad \text{and } f = g, \\ \quad \text{and } s_1,\dots \succ_{\mathrm{KBOlex}} t_1,\dots; \\ \text{and } \forall x \in \mathcal{V}.\ |f(\dots)|_x \geq |g(\dots)|_x. \end{cases} \tag{12a}$$

$$f(s_1,\dots) \succ_{\mathrm{KBO}} x \quad \text{iff} \quad |f(s_1,\dots)|_x \geq 1\,. \tag{12b}$$

$$x \succ_{\mathrm{KBO}} y \quad \text{iff} \quad \bot\,. \tag{12c}$$

The conditions on variable occurrences ensure that $s \succ_{\mathrm{KBO}} t \Rightarrow \forall \theta.\ s\theta \succ_{\mathrm{KBO}} t\theta$.

When we extend the order $\succ_{\mathrm{KBO}}$ with a variable preorder $\succeq_v$, the starting point is that $x \succ_v y \Rightarrow x \succ_{\mathrm{KBO}[v]} y$ and $x \sim_v y \Rightarrow x \sim_{\mathrm{KBO}[v]} y$. Then, to ensure that all the properties of a simplification order (included the one mentioned above) hold, we arrive at the following definition (similar to [1]).

$$f(s_1,\dots) \succ_{\mathrm{KBO}[v]} g(t_1,\dots) \quad \text{iff} \quad \begin{cases} \text{either } w(f(\dots)) > w(g(\dots)), \\ \text{or } w(f(s_1,\dots)) = w(g(t_1,\dots)) \\ \quad \text{and } f \succ_s g, \\ \text{or } w(f(s_1,\dots)) = w(g(t_1,\dots)) \\ \quad \text{and } f = g, \\ \quad \text{and } s_1,\dots \succ_{\mathrm{KBO}[v]_{\mathrm{lex}}} t_1,\dots; \\ \text{and } \forall x \in \mathcal{V}.\ \sum_{y \succeq_v x} |f(\dots)|_y \\ \quad \geq \sum_{y \succeq_v x} |g(\dots)|_y. \end{cases} \tag{13a}$$

$$f(s_1,\dots) \succ_{\mathrm{KBO}[v]} x \quad \text{iff} \quad \exists y \succeq_v x.\ |f(s_1,\dots)|_y \geq 1\,. \tag{13b}$$

$$x \succ_{\mathrm{KBO}[v]} y \quad \text{iff} \quad x \succ_v y\,. \tag{13c}$$

To check whether there exists a $\succeq'_v \supset \succeq_v$ such that $s \succ_{\mathrm{KBO}[v']} t$, we need to check whether there are some $x \succ y$ or $x = y$ relations that we can add to $\succeq_v$ such that all the conditions above hold (and such that it still remains a valid preorder). Let us denote "there exists a $\succeq'_v \supset \succeq_v$ such that $s \succ_{\mathrm{KBO}[v']} t$" by $s \succ_{\mathrm{KBO}[v,v']} t$. Then the definition is

$$f(s_1,\dots) \succ_{\mathrm{KBO}[v,v']} g(t_1,\dots) \quad \text{iff} \quad \begin{cases} \text{either } w(f(\dots)) > w(g(\dots)), \\ \text{or } w(f(s_1,\dots)) = w(g(t_1,\dots)) \\ \quad \text{and } f \succ_s g, \\ \text{or } w(f(s_1,\dots)) = w(g(t_1,\dots)) \\ \quad \text{and } f = g, \\ \quad \text{and } s_1,\dots \succ_{\mathrm{KBOlex}} t_1,\dots; \\ \text{and } \exists x_1, y_1, \dots \\ \quad \succeq'_v = (\succeq_v \cup \{\langle x_1, y_1\rangle, \dots\})^+ \text{ is a preorder} \\ \quad \text{such that } \forall x \in \mathcal{V}.\ \sum_{y \succeq'_v x} |f(\dots)|_y \\ \quad \geq \sum_{y \succeq'_v x} |g(\dots)|_y. \end{cases} \tag{14a}$$

$$f(s_1,\dots) \succ_{\mathrm{KBO}[v,v']} x \quad \text{iff} \quad \begin{cases} \exists y \not\succ_v x.\ |f(s_1,\dots)|_y \geq 1\,, \\ \quad \text{with } \succeq'_v = \succeq_v \cup \{x \succ y\} \\ \quad \text{or } \succeq'_v = \succeq_v \cup \{x = y\}\,. \end{cases} \tag{14b}$$

$$x \succ_{\mathrm{KBO}[v,v']} y \quad \text{iff} \quad \begin{cases} x \not\succ_v y \\ \quad \text{with } \succeq'_v = \succeq_v \cup \{x \succ y\}\,. \end{cases} \tag{14c}$$

This check can be used in Algorithm 1 for finding extensions of variable orderings that orient rewrite rules allowing required normalisations.

4.3 Connectedness

Testing for joinability (i.e. demodulating to $s \approx s$ or $s \not\approx s$) and ground joinability (presented in the previous section) require that each step in proving them is done via an oriented instance of an equation in the set. However, we can weaken this restriction, if we also change the notion of redundancy being used.

As criteria for redundancy of a clause, finding either joinability or ground joinability of a literal in the clause means that the clause can be deleted or the literal removed from the clause (in case of a positive or negative literal, resp.) in any context, that is, we can for example add them to a set of deleted clauses, and for any new clause, if it appears in that set, then immediately remove it since we already saw that it is redundant. The criterion of connectedness [3,21], however, is a criterion for redundancy of *inferences*. This means that a conclusion simplified by this criterion can be deleted (or rather, not added), but in that context only; if it ever comes up again as a conclusion of a different inference, then it is not necessarily also redundant. Connectedness was introduced in the context of equational completion, here we extend it to general clauses and show that it is a redundancy in the superposition calculus.

Definition 4. Terms s and t are *connected* under clauses U and unifier ρ wrt. a set of equations S if there exist terms $v_1, \dots, v_n$, equations $l_1 \approx r_1, \dots, l_{n-1} \approx r_{n-1}$, and substitutions $\sigma_1, \dots, \sigma_{n-1}$ such that:

(i) $v_1 = s$ and $v_n = t$,
(ii) for all $i \in 1, \ldots, n-1$, either $v_{i+1} = v_i[l_i\sigma_i \mapsto r_i\sigma_i]$ or $v_i = v_{i+1}[l_i\sigma_i \mapsto r_i\sigma_i]$, with $l_i \approx r_i \in S$,
(iii) for all $i \in 1, \ldots, n-1$, there exists w in $\bigcup_{C \in U} \bigcup_{p \approx q \in C} \{p, q\}$[4] such that for $u_i \in \{l_i, r_i\}$, either (a) $u_i\sigma_i \prec w\rho$, or (b) $u_i\sigma_i = w\rho$ and either $u_i \sqsubset w$ or $w \in C$ such that C is not a positive unit.

Theorem 5. Superposition inferences of the form

$$\frac{\underline{l \approx r} \vee C \quad \underline{s[u] \approx t} \vee D}{(s[u \mapsto r] \approx t \vee C \vee D)\rho}, \quad \begin{array}{l}\text{where } \rho = \mathrm{mgu}(l, u),\\ l\rho \not\preceq r\rho,\ s\rho \not\preceq t\rho,\\ \text{and } u \text{ not a variable,}\end{array} \tag{15}$$

where $s[u \mapsto r]\rho$ and $t\rho$ are connected under $\{l \approx r \vee C,\ s \approx t \vee D\}$ and unifier ρ wrt. some set of clauses S, are redundant inferences wrt. S.

Proof. Let us denote $s' = s[u \mapsto r]$. Let also $U = \{l \approx r \vee C,\ s \approx t \vee D\}$ and $M = \bigcup_{C \in U} \bigcup_{p \approx q \in C} \{p, q\}$. We will show that if $s'\rho$ and $t\rho$ are connected under U and ρ, by equations in S, then every instance of that inference obeys the condition for closure redundancy of an inference (see, Sect. 4), wrt. S.

Consider any $(s' \approx t \vee C \vee D)\rho \cdot \theta$ where $\theta \in \mathrm{GSubs}(U\rho)$. Either $s'\rho\theta = t\rho\theta$, and we are done (it follows from $\emptyset$), or $s'\rho\theta \succ t\rho\theta$, or $s'\rho\theta \prec t\rho\theta$.

Consider the case $s'\rho\theta \succ t\rho\theta$. For all $i \in 1, \ldots, n-1$, there exists a $C' \in U$ and a $w \in C'$ such that either (iii.a) $l_i\sigma_i\theta \prec w\rho\theta$, or (iii.b) $l_i\sigma_i\theta = w\rho\theta$ and $l_i \sqsubset v$, or (iii.b) $l_i\sigma_i\theta = w\rho\theta$ and C' is not a positive unit. Likewise for r_i. Therefore, for all $i \in 1, \ldots, n-1$, there exists a $C' \in U$ such that $(l_i \approx r_i) \cdot \sigma_i\theta \prec C' \cdot \rho\theta$. Since $(t \approx t \vee \cdots)\rho \cdot \theta$ is also smaller than $(s' \approx t \vee \cdots)\rho \cdot \theta$ and a tautology, then the instance $(s' \approx t \vee \cdots)\rho \cdot \theta$ of the conclusion follows from closures in $\mathrm{GClos}(S)$ such that each is smaller than one of $(l \approx r \vee C) \cdot \rho\theta$, $(s \approx t \vee D) \cdot \rho\theta$.

In the case that $s'\rho\theta \prec t\rho\theta$, the same idea applies, but now it is $(s' \approx s' \vee \cdots)\rho \cdot \theta$ which is smaller than $(s' \approx t \vee \cdots)\rho \cdot \theta$ and is a tautology.

Therefore, we have shown that for all $\theta \in \mathrm{GSubs}((l \approx r \vee C)\rho,\ (s \approx t \vee D)\rho)$, the instance $(s' \approx t \vee \cdots)\rho \cdot \theta$ of the conclusion follows from closures in $\mathrm{GClos}(S)$ which are all smaller than one of $(l \approx r \vee C) \cdot \rho\theta$, $(s \approx t \vee D) \cdot \rho\theta$. Since any valid superposition inference with ground clauses has to have $l = u$, then any $\theta' \in \mathrm{GSubs}(l \approx r \vee C,\ s \approx t \vee D,\ (s' \approx t \vee C \vee D)\rho)$ such that the inference $(l \approx r \vee C)\theta', (s \approx t \vee D)\theta' \vdash (s' \approx t \vee C \vee D)\rho\theta'$ is valid must have $\theta' = \rho\theta''$, since ρ is the most general unifier. Therefore, we have shown that for all $\theta' \in \mathrm{GSubs}(l \approx r \vee C,\ s \approx t \vee D,\ (s' \approx t \vee C \vee D)\rho)$ for which $(l \approx r \vee C)\theta', (s \approx t \vee D)\theta' \vdash (s' \approx t \vee C \vee D)\rho\theta'$ is a valid superposition inference, the instance $(s' \approx t \vee \cdots)\rho \cdot \theta'$ of the conclusion follows from closures in $\mathrm{GClos}(S)$ which are all smaller than one of $(l \approx r \vee C) \cdot \theta'$, $(s \approx t \vee D) \cdot \theta'$, so the inference is redundant. □

[4] That is, in the set of top-level terms of literals of clauses in U.

Theorem 6. Superposition inferences of the form

$$\frac{\underline{l \approx r} \vee C \quad \underline{s[u] \not\approx t} \vee D}{(s[u \mapsto r] \not\approx t \vee C \vee D)\rho}, \quad \begin{array}{l}\text{where } \rho = \text{mgu}(l, u),\\ l\rho \not\preceq r\rho,\ s\rho \not\preceq t\rho,\\ \text{and } u \text{ not a variable,}\end{array} \tag{16}$$

where $s[u \mapsto r]\rho$ and $t\rho$ are connected under $\{l \approx r \vee C,\ s \not\approx t \vee D\}$ and unifier ρ wrt. some set of clauses S, are redundant inferences wrt. $S \cup \{(C \vee D)\rho\}$.

Proof. Analogously to the previous proof, we find that for all instances of the inference, the closure $(s' \not\approx t \vee \cdots)\rho\cdot\theta$ follows from smaller closure $(t \not\approx t \vee \cdots)\rho\cdot\theta$ or $(s' \not\approx s' \vee \cdots)\rho\cdot\theta$ and closures $(l_i \approx r_i)\cdot\sigma_i\theta$ smaller than $\max\{(l \approx r \vee C)\cdot\theta\ ,\ (s \not\approx t \vee D)\cdot\theta\ ,\ (s' \not\approx t \vee C \vee D)\rho\cdot\theta\}$. But $(t \not\approx t \vee C \vee D)\rho\cdot\theta$ and $(s' \not\approx s' \vee C \vee D)\rho\cdot\theta$ both follow from smaller $(C \vee D)\rho\cdot\theta$, therefore the inference is redundant wrt. $S \cup \{(C \vee D)\rho\}$. □

4.4 Ground Connectedness

Just as joinability can be generalised to ground joinability, so can connectedness be generalised to ground connectedness. Two terms s, t are *ground connected* under U and ρ wrt. S if, for all $\theta \in \text{GSubs}(s, t)$, $s\theta$ and $t\theta$ are connected under D and ρ wrt. S. Analogously to strong ground joinability, we have that if s and t are connected using $\succ_{t[v]}$ for all total $\succeq_v$ over $\text{Vars}(s, t)$, then s and t are ground connected.

Theorem 7. Superposition inferences of the form

$$\frac{\underline{l \approx r} \vee C \quad \underline{s[u] \approx t} \vee D}{(s[u \mapsto r] \approx t \vee C \vee D)\rho}, \quad \begin{array}{l}\text{where } \rho = \text{mgu}(l, u),\\ l\rho \not\preceq r\rho,\ s\rho \not\preceq t\rho,\\ \text{and } u \text{ not a variable,}\end{array} \tag{17}$$

where $s[u \mapsto r]\rho$ and $t\rho$ are ground connected under $\{l \approx r \vee C,\ s \approx t \vee D\}$ and unifier ρ wrt. some set of clauses S, are redundant inferences wrt. S.

Theorem 8. Superposition inferences of the form

$$\frac{\underline{l \approx r} \vee C \quad \underline{s[u] \not\approx t} \vee D}{(s[u \mapsto r] \not\approx t \vee C \vee D)\rho}, \quad \begin{array}{l}\text{where } \rho = \text{mgu}(l, u),\\ l\rho \not\preceq r\rho,\ s\rho \not\preceq t\rho,\\ \text{and } u \text{ not a variable,}\end{array} \tag{18}$$

where $s[u \mapsto r]\rho$ and $t\rho$ are ground connected under $\{l \approx r \vee C,\ s \not\approx t \vee D\}$ and unifier ρ wrt. some set of clauses S, are redundant inferences wrt. $S \cup \{(C \vee D)\rho\}$.

Proof. The proof of Theorem 7 and 8 is analogous to that of Theorem 5 and 6. The weakening of connectedness to ground connectedness only means that the proof of connectedness (e.g. the v_i, $l_i \approx r_i$, σ_i) may be different for different ground instances. For all the steps in the proof to hold we only need that for all the instances $\theta \in \text{GSubs}(l \approx r \vee C\ ,\ s \mathrel{\dot{\approx}} t \vee D\ ,\ (s[u \mapsto r] \mathrel{\dot{\approx}} t \vee C \vee D)\rho)$ of the inference, $\theta = \sigma\theta'$ with $\sigma \in \text{GSubs}(s[u \mapsto r]\rho, t\rho)$, which is true. □

Discussion about the strategy for implementation of connectedness and ground connectedness is outside the scope of this paper.

5 Evaluation

We implemented ground joinability in a theorem prover for first-order logic, iProver [10,16].[5] iProver combines superposition, Inst-Gen, and resolution calculi. For superposition, iProver implements a range of simplifications including encompassment demodulation, AC normalisation [10], light normalisation [16], subsumption and subsumption resolution. We run our experiments over FOF problems of the TPTP v7.5 library [23] (17 348 problems) on a cluster of Linux servers with 3 GHz 11 core CPUs, 128 GB memory, with each problem running on a single core with a time limit of 300 s. We used a default strategy (which has not yet been fine-tuned after the introduction of ground joinability), with superposition enabled and the rest of the components disabled. With ground joinability enabled, iProver solved 133 problems more which it did not solve without ground joinability. Note that this excludes the contribution of AC ground joinability or encompassment demodulation [11] (always enabled).

Some of the problems are not interesting for this analysis because ground joinability is not even tried, either because they are solved before superposition saturation begins, or because they are ground. If we exclude these, we are left with 10 005 problems. Ground joinability is successfully used to eliminate clauses in 3057 of them (30.6%, Fig. 1a). This indicates that ground joinability is useful in many classes of problems, including in non-unit problems where it previously had never been used.

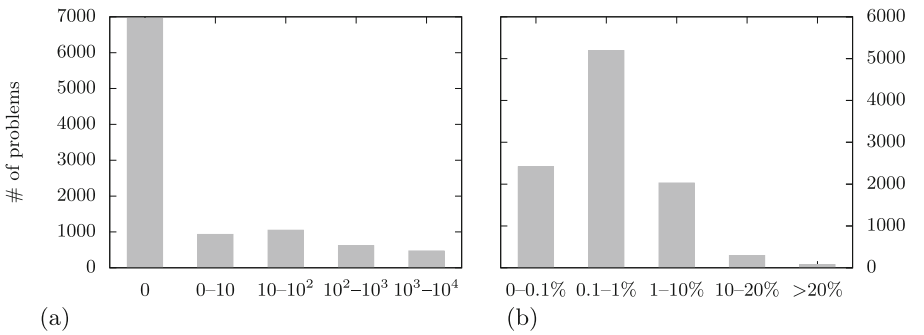

Fig. 1. (a) Clauses simplified by ground joinability. (b) % of runtime spent in gr. joinability

In terms of the performance impact of enabling ground joinability, we measure that among problems whose runtime exceeds 1 s, only in 72 out of 8574 problems does the time spent inside the ground joinability algorithm exceed 20% of runtime, indicating that our incremental algorithm is efficient and suitable for broad application (Fig. 1b).

[5] iProver is available at http://www.cs.man.ac.uk/~korovink/iprover.

TPTP classifies problems by rating in [0,1]. Problems with rating ≥0.9 are considered to be very challenging. Problems with rating 1.0 have never been solved by any automated theorem prover. iProver using ground joinability solves 3 previously unsolved rating 1.0 problems, and 7 further problems with rating in [0.9,1.0[(Table 1). We note that some of these latter (e.g. `LAT140-1`, `ROB018-10`, `REL045-1`) were previously only solved by UEQ or SMT provers, but not by any full first-order prover.

Table 1. Hard or unsolved problems in TPTP, solved by iProver with ground joinability.

Name	Rating	Name	Rating
`LAT140-1`	0.90	`ROB018-10`	0.95
`REL045-1`	0.90	`LCL477+1`	0.97
`LCL557+1`	0.92	`LCL478+1`	1.00
`LCL563+1`	0.92	`CSR039+6`	1.00
`LCL474+1`	0.94	`CSR040+6`	1.00

6 Conclusion and Further Work

In this work we extended the superposition calculus with ground joinability and connectedness, and proved that these rules preserve completness using a modified notion of redundancy, thus bringing for the first time these techniques for use in full first-order logic problems. We have also presented an algorithm for checking ground joinability which attempts to check as few variable preorders as possible.

Preliminary results show three things: (1) ground joinability is applicable in a sizeable number of problems across different domains, including in non-unit problems (where it was never applied before), (2) our proposed algorithm for checking ground joinability is efficient, with over $\frac{3}{4}$ of problems spending less than 1% of runtime there, and (3) application of ground joinability in the superposition calculus of iProver improves overall performance, including discovering solutions to hitherto unsolved problems.

These results are promising, and further optimisations can be done. Immediate next steps include fine-tuning the implementation, namely adjusting the strategies and strategy combinations to make full use of ground joinability and connectedness. iProver uses a sophisticated heuristic system which has not yet been tuned for ground joinability and connectedness [14].

In terms of practical implementation of connectedness and ground connectedness, further research is needed on the interplay between those (criteria for redundancy of inferences) and joinability and ground joinability (criteria for redundancy of clauses).

On the theoretical level, recent work [24] provides a general framework for saturation theorem proving, and we will investigate how techniques developed in this paper can be incorporated into this framework.

References

1. Avenhaus, J., Hillenbrand, T., Löchner, B.: On using ground joinable equations in equational theorem proving. J. Symb. Comput. **36**(1), 217–233 (2003). https://doi.org/10.1016/S0747-7171(03)00024-5
2. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998). ISBN 978-0521779203
3. Bachmair, L., Dershowitz, N.: Critical pair criteria for completion. J. Symb. Comput. **6**(1), 1–18 (1988). https://doi.org/10.1016/S0747-7171(88)80018-X
4. Bachmair, L., Dershowitz, N., Plaisted, D.A.: Completion without failure. In: Aït-Kaci, H., Nivat, M. (eds.) Resolution of Equations in Algebraic Structures, vol. II: Rewriting Techniques, pp. 1–30. Academic Press (1989). https://doi.org/10.1016/B978-0-12-046371-8.50007-9
5. Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. J. Log. Comput. **4**(3), 217–247 (1994). https://doi.org/10.1093/logcom/4.3.217
6. Bachmair, L., Ganzinger, H., Lynch, C.A., Snyder, W.: Basic paramodulation. Inf. Comput. **121**(2), 172–192 (1995). https://doi.org/10.1006/inco.1995.1131. ISSN 0890-5401
7. Barthelemy, J.P.: An asymptotic equivalent for the number of total preorders on a finite set. Discret. Math. **29**(3), 311–313 (1980). https://doi.org/10.1016/0012-365x(80)90159-4
8. Claessen, K., Smallbone, N.: Efficient encodings of first-order horn formulas in equational logic. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNCS (LNAI), vol. 10900, pp. 388–404. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94205-6_26
9. Dershowitz, N., Manna, Z.: Proving termination with multiset orderings. Commun. ACM **22**(8), 465–476 (1979). https://doi.org/10.1145/359138.359142
10. Duarte, A., Korovin, K.: Implementing superposition in iProver (system description). In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020. LNCS (LNAI), vol. 12167, pp. 388–397. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51054-1_24
11. Duarte, A., Korovin, K.: AC simplifications and closure redundancies in the superposition calculus. In: Das, A., Negri, S. (eds.) TABLEAUX 2021. LNCS (LNAI), vol. 12842, pp. 200–217. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86059-2_12
12. Duarte, A., Korovin, K.: Ground Joinability and Connectedness in the Superposition Calculus (2022, to appear)
13. Hillenbrand, T., Buch, A., Vogt, R., Löchner, B.: Waldmeister—high-performance equational deduction. J. Autom. Reason. **18**(2), 265–270 (1997). https://doi.org/10.1023/A:1005872405899
14. Holden, E.K., Korovin, K.: Heterogeneous heuristic optimisation and scheduling for first-order theorem proving. In: Kamareddine, F., Sacerdoti Coen, C. (eds.) CICM 2021. LNCS (LNAI), vol. 12833, pp. 107–123. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81097-9_8
15. Knuth, D.E., Bendix, P.: Simple word problems in universal algebras. In: Leech, J. (ed.) Computational Problems in Abstract Algebra, pp. 263–297. Pergamon (1970). https://doi.org/10.1016/B978-0-08-012975-4.50028-X
16. Korovin, K.: iProver—an instantiation-based theorem prover for first-order logic (system description). In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR

2008. LNCS (LNAI), vol. 5195, pp. 292–298. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-71070-7_24
17. Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 1–35. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_1
18. Martin, U., Nipkow, T.: Ordered rewriting and confluence. In: Stickel, M.E. (ed.) CADE 1990. LNCS, vol. 449, pp. 366–380. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-52885-7_100
19. Nieuwenhuis, R., Rubio, A.: Paramodulation-based theorem proving. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 2, pp. 371–443. Elsevier and MIT Press (2001). ISBN 0-444-50813-9
20. Schulz, S.: System description: E 1.8. In: McMillan, K., Middeldorp, A., Voronkov, A. (eds.) LPAR 2013. LNCS, vol. 8312, pp. 735–743. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45221-5_49
21. Smallbone, N.: Twee: an equational theorem prover. In: Platzer, A., Sutcliffe, G. (eds.) CADE 2021. LNCS (LNAI), vol. 12699, pp. 602–613. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-79876-5_35
22. Sutcliffe, G.: The CADE ATP system competition–CASC. AI Mag. **37**(2), 99–101 (2016). https://doi.org/10.1609/aimag.v37i2.2620
23. Sutcliffe, G.: The TPTP problem library and associated infrastructure–from CNF to TH0, TPTP v6.4.0. J. Autom. Reason. **59**(4), 483–502 (2017). https://doi.org/10.1007/s10817-017-9407-7
24. Waldmann, U., Tourret, S., Robillard, S., Blanchette, J.: A comprehensive framework for saturation theorem proving. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020. LNCS (LNAI), vol. 12166, pp. 316–334. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-51074-9_18
25. Winkler, S., Moser, G.: MædMax: a maximal ordered completion tool. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNCS (LNAI), vol. 10900, pp. 472–480. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94205-6_31