Reducibility Constraints in Superposition

Márton Hajdu[1] (✉), Laura Kovács[1], Michael Rawson[1], and Andrei Voronkov[2,3]

[1] TU Wien, Vienna, Austria
marton.hajdu@tuwien.ac.at
[2] University of Manchester, Manchester, UK
[3] EasyChair, Manchester, UK

Abstract. Modern superposition inference systems aim at reducing the search space by introducing redundancy criteria on clauses and inferences. This paper focuses on reducing the number of superposition inferences with a single clause by blocking inferences into some terms, provided there were previously made inferences of a certain form performed with predecessors of this clause. Other calculi based on blocking inferences, for example basic superposition, rely on variable abstraction or equality constraints to express irreducibility of terms, resulting however in blocking inferences with all subterms of the respective terms. Here we introduce reducibility constraints in superposition to enable a more expressive blocking mechanism for inferences. We show that our calculus remains (refutationally) complete and present redundancy notions. Our implementation in the theorem prover Vampire demonstrates a considerable reduction in the size of the search space when using our new calculus.

Keywords: Saturation · Superposition · Redundancy· Reducibility constraints

1 Introduction

Automated reasoners in first-order logic with equality commonly rely on the *superposition calculus* [25,5]. This calculus has been extended with various improvements in order to reduce the search space. For instance, avoiding superposition into variables and ordering literals and clauses are common practices in modern theorem provers [21,29,34].

To reduce generation of redundant clauses in equational reasoning, the "basicness" restriction [16] was introduced at the term level. This idea aided, for example, in finding the proof of the Robbins problem [24]. This restriction blocks superposition (rewriting) inferences into terms resulting from (quantifier) instantiations, considering such terms irreducible in further proof steps. This approach was further generalised to block superposition into terms above variable positions in basic superposition/paramodulation [7,26], while preserving refutational completeness. However, blocking and applying different rewrite steps among equal

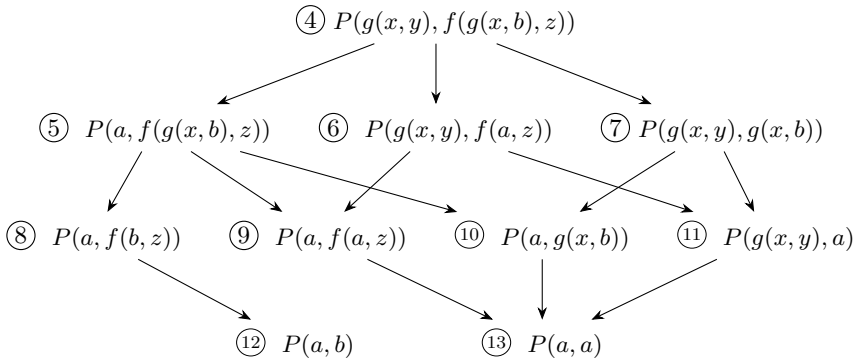

Fig. 1. Possible superposition sequences into ④.

terms impacts proof search. In this paper, we propose a number of different ways to block inferences, so that the resulting calculus remains complete. The effect of these restrictions resembles some strategies from term rewriting, such as innermost and outermost strategies.

Motivating example. Consider the following satisfiable set $\mathcal{C}$ of clauses:

$$\mathcal{C} = \left\{ \begin{array}{ll} ① \; g(x,b) \simeq a, & ② \; f(x,b) \simeq x, \\ ③ \; g(a,x) \simeq x, & ④ \; P(g(x,y), f(g(x,b),z)) \end{array} \right\}$$

where x, y are variables, a, b constants, f, g function symbols, and P is a predicate symbol. In this paper $\simeq$ denotes equality. Figure 1 shows some derivations of $P(a,a)$ by consecutively superposing into ④ with ① and ②. It also shows a derivation of $P(a,b)$ by superposing into ④ with ①, then with ③ and ②. Note that Figure 1 contains many redundant clauses. For example, ④ is redundant w.r.t. ⑥ and ①, as it is a logical consequence of (smaller) ⑥ and ①. Similarly, ⑦ is redundant w.r.t. ⑪ and ①.

Many derivations of Figure 1 could however be avoided by using a rewrite order between the inferences. For example, a *leftmost-innermost rewrite order* on inferences derives ⑬ along the path ④–⑤–⑨–⑬. Whenever we would deviate from the leftmost-innermost rewrite order when rewriting a term t, we enforce the order by requiring that any term t' that is to the left of or inside t is irreducible in further derivations. In other words, we block further inferences with t'. With such a restriction, we cannot rewrite $g(x,y)$ in clause ⑥, as $g(x,y)$ was to the left of the previously rewritten term $f(g(x,b),z)$. Hence, when using a leftmost-innermost rewrite upon in Figure 1, ⑨ is only generated by the derivation path ④–⑤–⑨. Similarly, ⑪ cannot be derived from ⑦ but can be derived from ⑥.

Our contributions. We introduce a new superposition calculus that enables various ways to block (rewrite) inferences during proof search. Key to our calculus are *reducibility constraints* to restrict the order of superposition inferences

(Section 3). Our approach supports and generalizes, among others, the leftmost-innermost rewrite orders mentioned in the motivating example by means of *irreducibility constraints*, allowing us to reduce the number of generated clauses. Furthermore, in our motivating example the derivation ⑤–⑧–⑫ of Figure 1 is not needed for the following reason. By superposing into ② with ③, we derive $a \simeq b$, which makes one of ⑫ and ⑬ redundant w.r.t. the other. As ① was used to rewrite $g(x, b)$ in Figure 1 and derive ⑤, we block superposition into $g(x, b)$ with all clauses except ① in further derivations. We express this requirement via a *one-step reducibility constraint* (Definition 1), resulting in the BLINC – BLocked INference Calculus. As such, BLINC is parameterized by a rewrite ordering and (ir)reducibility constraints.

We prove [4] that our BLINC calculus is refutationally complete, for which we use a model construction technique (Section 4) with new features introduced to take care of constraints. We extend our calculus with redundancy elimination (Section 5). When evaluating the BLINC calculus implemented in the Vampire theorem prover, our experiments show that reducibility constraints significantly reduce the search space (Section 6).

2 Preliminaries

We work in standard *first-order logic with equality*, where equality is denoted by $\simeq$. We use variables x, y, z, v, w and terms s, t, u, l, r, all possibly with indices. A term is *ground* if it contains no variables. A literal is an unordered pair of terms with polarity, i.e. an equality $s \simeq t$ or a disequality $s \not\simeq t$. We write $s \bowtie t$ for either an equality or a disequality. A *clause* is a multiset of literals. We denote clauses by B, C, D and denote by $\Box$ the *empty clause* that is logically equivalent to $\bot$.

An *expression* E is a term, literal or clause. We will also consider as expressions constraints and constrained clauses introduced later. An expression is called *ground* if it contains no variables. We write $E[s]$ to state that the expression E contains a distinguished occurrence of the term s at some position. Further, $E[s \mapsto t]$ denotes that this occurrence of s is replaced with t; when s is clear from the context, we simply write $E[t]$. We say that t is a *subterm* of $s[t]$, denoted by $t \trianglelefteq s[t]$; and a *strict subterm* if additionally $t \neq s[t]$, denoted by $t \triangleleft s[t]$. A *substitution* σ is a mapping from variables to terms, such that the set of variables $\{x \mid \sigma(x) \neq x\}$ is finite. We denote substitutions by θ, σ, ρ, μ, η. The application of a substitution θ on an expression E is denoted $E\theta$. A substitution θ is called *grounding for an expression* E if $E\theta$ is ground. We denote the set of grounding substitutions for an expression E by GSubs, that is $\mathsf{GSubs}(E) = \{\theta \mid E\theta \text{ is ground}\}$. We denote the empty substitution by ε.

A substitution θ is more general than a substitution σ if $\theta\eta = \sigma$ for some substitution η. A substitution θ is a *unifier* of two terms s and t if $s\theta = t\theta$, and is a *most general unifier*, denoted $\mathsf{mgu}(s, t)$, if for every unifier η of s and t, there

[4] detailed proofs are in the full version of this paper [15]

exists a substitution μ s.t. $\eta = \theta\mu$. We assume that the most-general unifiers of terms are idempotent [2].

A binary relation $\rightarrow$ over the set of terms is a *rewrite relation* if (i) $l \rightarrow r \Rightarrow l\theta \rightarrow r\theta$ and (ii) $l \rightarrow r \Rightarrow s[l] \rightarrow s[r]$ for any term l, r, s and substitution θ. The *reflexive and transitive closure* of a relations $\rightarrow$ is denoted by $\rightarrow^*$. We write $\leftarrow$ to denote the inverse of $\rightarrow$. Two terms are *joinable*, denoted by $s \downarrow t$, if there exists a term u s.t. $s \rightarrow^* u \leftarrow^* t$. A *rewrite system* R is a set of rewrite rules. A term l is *irreducible* in R if there is no r s.t. $l \rightarrow r \in R$. Joinability w.r.t. R will be denoted by $s \downarrow_R t$. A *rewrite ordering* is a strict rewrite relation. A *reduction ordering* is a well-founded rewrite ordering. In this paper we consider reduction orderings which are total on ground terms, that is they satisfy $s \rhd t \Rightarrow s \succ t$; such orderings are also called *simplification orderings*.

We use the standard definition of a *bag extension* of an ordering [12]. An ordering $\succ$ on terms induces an ordering on literals, by identifying $s \simeq t$ with the multiset $\{s, t\}$ and $s \not\simeq t$ with the multiset $\{s, s, t, t\}$, and using the bag extension of $\succ$. We denote this induced ordering on literals also with $\succ$. Likewise, the ordering $\succ$ on literals induces the ordering on clauses by using the bag extension of $\succ$. Again, we denote this induced ordering on clauses also with $\succ$. The induced relations $\succ$ on literals and clauses are well-founded (resp. total) if so is the original relation $\succ$ on terms. In examples used in this paper, we assume a KBO simplification ordering with constant weight [19].

Many first-order theorem provers work with clauses [29,34,21]. Let S be a set of clauses. Often, systems *saturate* S by computing all logical consequences of S with respect to a sound inference system $\mathcal{I}$. The process of saturating S is called *saturation*. An inference system $\mathcal{I}$ is a set of inference rules of the form

$$\frac{C_1 \quad \dots \quad C_n}{C},$$

where $C_1, \dots, C_n$ are the *premises* and C is the *conclusion* of the inference. The inference rule is *sound* if its conclusion is the logical consequence of its premises, that is $C_1, \dots, C_n \models C$. The inference is *reductive* w.r.t. an ordering $\succ$ if $C \succ C_i$, for some $i = 1, \dots, n$. An inference system $\mathcal{I}$ is *sound* if all its inferences are sound. An inference system $\mathcal{I}$ is *refutationally complete* if for every unsatisfiable clause set S, there is a derivation of the empty clause in $\mathcal{I}$. An interpretation I is a model of an expression E if E is true in I. A clause C that is false in an interpretation I is a *counterexample* for I. If a clause set contains a counterexample, then it also contain a minimal counterexample w.r.t. a simplification ordering $\succ$ [6].

3 Reducibility Constraints

This section presents a new blocking calculus, called BLINC (BLocked INference Calculus). This calculus uses specific constraints to block inferences.

Definition 1 (Constraints). Let l be a non-variable term and r a term. We call the expression $l \rightsquigarrow r$ a *one-step reducibility constraint*, and the expression $\downarrow l$ an *irreducibility constraint*. A *constraint* is one of the two. □

$$(\mathsf{Sup}_{\ni}) \frac{\underline{l \simeq r} \vee C \mid \Pi \quad s[u] \bowtie \underline{t} \vee D \mid \Gamma}{(s[r] \bowtie t \vee C \vee D)\sigma \mid \Delta} \quad \text{where} \quad \begin{array}{l}(1)\ u \text{ is not a variable,}\\ (2)\ \sigma = \mathsf{mgu}(l,u),\\ (3)\ t\sigma \not\succeq s[u]\sigma,\ r\sigma \not\succeq l\sigma,\\ (4)\ \Delta = \Gamma\sigma \cup \{l\sigma \rightsquigarrow r\sigma\} \\ \qquad\qquad \cup\, \mathcal{B}_{\ni}(s[u]\sigma, l\sigma),\\ (5)\ \text{the conclusion is not blocked,}\end{array}$$

$$(\mathsf{EqRes}_{\ni}) \frac{\underline{s \not\simeq t} \vee C \mid \Gamma}{C\sigma \mid \Gamma\sigma} \quad \text{where} \quad \begin{array}{l}(1)\ \sigma = \mathsf{mgu}(s,t),\\ (2)\ \text{the conclusion is not blocked,}\end{array}$$

$$(\mathsf{EqFac}_{\ni}) \frac{\underline{s \simeq t} \vee \underline{u \simeq w} \vee C \mid \Gamma}{(s \simeq t \vee t \not\simeq w \vee C)\sigma \mid \Gamma\sigma} \quad \text{where} \quad \begin{array}{l}(1)\ \sigma = \mathsf{mgu}(s,u),\\ (2)\ t\sigma \not\succeq s\sigma,\ w\sigma \not\succeq t\sigma,\\ (3)\ \text{the conclusion is not blocked.}\end{array}$$

Fig. 2. The BLINC calculus

Now let us define the semantics of these constraints.

Definition 2 (Satisfied Constraints, Violated Constraints). Let R be a rewrite system. We say that R *satisfies* $l \rightsquigarrow r$ if $l \to r \in R$ and *satisfies* $\downarrow l$ if l is irreducible in R. We say that R *violates* a constraint if it does not satisfy it. □

An expression $C \mid \Gamma$ is a *constrained clause*, where C is a clause and Γ a finite set of constraints. A constrained clause $C \mid \Gamma$ is true iff C is true. We denote constrained clauses $\mathcal{C}$, $\mathcal{D}$, possibly with indices.

Definition 3 (Blocked Constrained Clause, Blocked Inference). Let $\mathcal{C} = C \mid \Gamma$ be a constrained clause. We call the constraint $l \rightsquigarrow r \in \Gamma$ *active in* $\mathcal{C}$ if $s \succ l$ for some term s in C. Likewise, we call $\downarrow l \in \Gamma$ *active in* $\mathcal{C}$ if $s \succ l$ for some term s in C. We call $\mathcal{C}$ *blocked* if either it contains two active constraints $l \rightsquigarrow r$ and $l \rightsquigarrow r'$ such that r and r' are not unifiable, or it contains two active constraints $l \rightsquigarrow r$ and $\downarrow l$. An inference is *blocked* if its conclusion is blocked. □

Our superposition calculus BLINC uses constrained clauses and bans inferences with blocked conclusions. For that, we attach constraints to clauses, as follows.

Definition 4 (S-ordering). An *S-ordering* is a partial strict well-order $\ni$ on terms that is stable under substitutions. We use the function $\mathcal{B}_{\ni}$ defined below to attach constraints to clauses.

$$\mathcal{B}_{\ni}(s,l) := \{\downarrow u \mid u \Subset l, u \text{ is non-variable and } u \trianglelefteq s\} \qquad \square$$

BLINC is shown in Figure 2. We assume a literal selection function satisfying the standard condition on $\succ$ and underline selected literals. The next example illustrates blocked BLINC inferences.

Example 1. We use the order $\succ$ on terms as the S-ordering. A $\mathsf{Sup}_{\ni}$ inference of BLINC into ④ with ② from our motivating example from page 2 results in

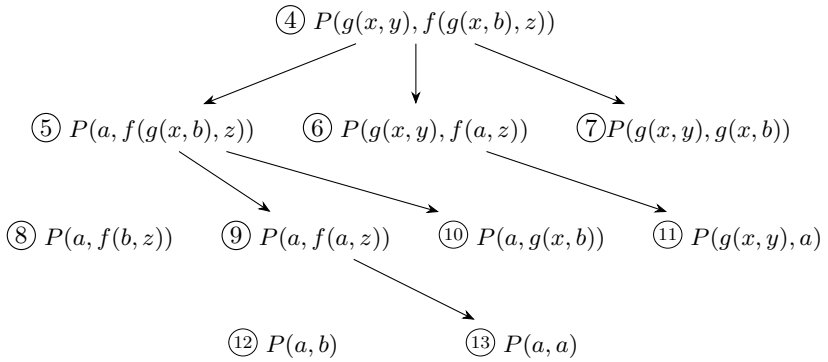

Fig. 3. Inferences from Figure 1 with blocked inferences in BLINC removed. Figure 3 illustrates the effectiveness of reducibility constraints when compared to Figure 1: we removed arcs corresponding to inferences blocked when the order $\succ$ is used as the S-ordering. Of the 14 original inferences as in Figure 1, only 7 are not blocked in Figure 3.

$$\frac{f(x,b) \simeq x \quad P(g(x,y), f(g(x,b),z))}{P(g(x,y), g(x,b)) \mid \{\downarrow b, \downarrow g(x,y), \downarrow g(x,b), f(g(x,b),b) \rightsquigarrow g(x,b)\}}.$$

Note that the conclusion is a constrained variant of clause ⑦ of Figure 1. Now, the superposition of ① into $g(x,y)$, and hence the following variant of clause ⑩ of Figure 1, is blocked:

$$\frac{g(x,b) \simeq a \quad P(g(x,y), g(x,b)) \mid \{\downarrow g(x,y), \downarrow g(x,b), f(g(x,b),b) \rightsquigarrow g(x,b)\}}{P(a, g(x,b)) \mid \{\downarrow b, \downarrow g(x,b), f(g(x,b),b) \rightsquigarrow g(x,b), g(x,b) \rightsquigarrow a\}}$$

Note that the conclusion is blocked by the active constraints $g(x,b) \rightsquigarrow a$ and $\downarrow g(x,b)$. Figure 3 shows the modified version of Figure 1, when using the inference rules of BLINC to generate fewer clauses than in Figure 1. □

Example 2. Consider now a sequence of superposition inferences into ④ by ① and then by ③. That is, we consider the derivation ④–⑤–⑧ from Figure 1 as:

$$\frac{g(a,x) \simeq x \quad \dfrac{g(x,b) \simeq a \quad P(g(x,y), f(g(x,b),z))}{P(a, f(g(x,b),z)) \mid \{\downarrow b, g(x,b) \rightsquigarrow a\}}}{P(a, f(b,z)) \mid \{\downarrow a, \downarrow b, g(a,b) \rightsquigarrow a, g(a,b) \rightsquigarrow b\}}$$

The resulting conclusion is constrained and blocked, as we have two active constraints $g(a,b) \rightsquigarrow a$ and $g(a,b) \rightsquigarrow b$. As such and as shown in Figure 3, clause ⑧ (and also clause ⑫) will not be generated by BLINC, in contrast to Figure 1. □

4 Model Construction in BLINC

This section shows completeness of BLINC, with a proof which resembles that of Duarte and Korovin [13]. We start by adjusting terminology to our setting and discussing key differences compared with standard completeness proofs.

Definition 5 (Closure). Let $\mathcal{C} = C \mid \Gamma$ be a constrained clause and θ a substitution. The pair $\mathcal{C} \cdot \theta$ is called a *closure* and is logically equivalent to $C\theta$. A closure $(C \mid \Gamma) \cdot \theta$ is *ground* if $C\theta \mid \Gamma\theta$ is ground, in which case we say that θ is *grounding* for $C \mid \Gamma$ and call $(C \mid \Gamma) \cdot \theta$ a *ground instance of* $C \mid \Gamma$.

The set of all ground instances of $\mathcal{C}$ is denoted $\mathcal{C}^*$. We will denote ground closures by $\mathbb{C}, \mathbb{D}$, maybe with indexes. If N is a set of constrained clauses, then N^* is defined as $\bigcup_{\mathcal{C} \in N} \mathcal{C}^*$. If $C \succ D$, we write $C \mid \Gamma \succ D \mid \Delta$. Similarly, if $C\theta \mid \Gamma\theta \succ D\sigma \mid \Delta\sigma$, then we write $(C \mid \Gamma) \cdot \theta \succ (D \mid \Delta) \cdot \sigma$. □

A crucial part in the completeness proof of BLINC is reducing minimal counterexamples to smaller ones. However, due to blocked inference conditions (5) in $\mathsf{Sup}_{\Rsh}$, (2) in $\mathsf{EqRes}_{\Rsh}$, and (3) in $\mathsf{EqFac}_{\Rsh}$, such a counterexample reduction may not be possible. We solve this problem in three steps:

1. Given a saturated set of clauses N, we construct a model for a subset of its closures $\mathcal{U}(N) \subseteq N^*$, namely, for so-called *unblocked closures* (Definition 6).
2. We show that if the empty clause $\square$ is not in $\mathcal{U}(N)$, then the model satisfies each closure in $\mathcal{U}(N)$ (Theorem 1). That is, we show that counterexamples in $\mathcal{U}(N)$ can be reduced to smaller counterexamples that are also in $\mathcal{U}(N)$. This avoids the aforementioned problem with blocked inferences.
3. We then show that the model also satisfies all closures in N^* (Theorem 2).

Definition 6 (Partial/Total Models, Blocked/Productive Closures). Let N be a set of constrained clauses. We will define, for every closure $\mathbb{C} \in N^*$, the rewrite system $R_{\mathbb{C}}$ and refer to all such rewrite systems as *partial models*. The definition will be made by induction on the relation $\succ$ on ground closures. In parallel to defining $R_{\mathbb{C}}$, we also define the rewrite system

$$R_{\prec \mathbb{C}} = \bigcup_{\mathbb{D} \prec C} R_{\mathbb{D}}.$$

The partial model $R_{\mathbb{C}}$ will either be the same as $R_{\prec \mathbb{C}}$, or obtained from $R_{\prec \mathbb{C}}$ by adding a single rewrite rule. In the latter case will call the closure $\mathbb{C}$ *productive*.

The *reduced closure* of a ground closure $\mathcal{C} \cdot \theta$ is defined as the closure $\mathcal{C} \cdot \theta'$ such that for each variable x occurring in $\mathcal{C}$, we have that $\theta'(x)$ is the normal form of $\theta(x)$ in $R_{\prec \mathcal{C} \cdot \theta}$. We call a ground closure *reduced* if its reduced form coincides with this closure. Let $\mathcal{C} \cdot \theta$ be a ground closure and $\mathcal{C} \cdot \theta'$ be its reduced form. We say that $\mathcal{C} \cdot \theta$ is *blocked w.r.t. N* if $R_{\prec \mathcal{C} \cdot \theta'}$ violates some constraint in $\Gamma\theta'$ that is active in $\mathcal{C}\theta'$. A closure that is not blocked w.r.t. N is called *unblocked w.r.t. N*. Let $\mathcal{C} = (l \simeq r \vee C') \mid \Gamma$. The closure $\mathcal{C} \cdot \theta$ is called *productive* if

(i) $\mathcal{C} \cdot \theta$ is false in $R_{\prec \mathcal{C} \cdot \theta}$,
(ii) $l\theta \simeq r\theta$ is strictly maximal in $\mathcal{C}\theta$,
(iii) $l\theta \succ r\theta$,
(iv) $C'\theta$ is false in $R_{\prec C \cdot \theta} \cup \{l\theta \to r\theta\}$,
(v) $l\theta$ is irreducible in $R_{\prec \mathcal{C} \cdot \theta}$,
(vi) $\mathcal{C} \cdot \theta$ is unblocked w.r.t N.

Now we define

$$R_{\mathcal{C}\cdot\theta} = \begin{cases} R_{\prec\mathcal{C}\cdot\theta} \cup \{l\theta \to r\theta\}, & \text{if } \mathcal{C}\cdot\theta \text{ is productive;} \\ R_{\prec\mathcal{C}\cdot\theta}, & \text{otherwise.} \end{cases}$$

$$R_\infty = \bigcup_{\mathbb{C}\in N^*} R_\mathbb{C}$$

Finally, we call R_∞ *the total model* and define $\mathcal{U}(N)$ as the set of all closures in N^* unblocked w.r.t. N. □

This construction has two standard properties that we will use in our proofs:

1. $R_\mathbb{C} \models \mathbb{C}$ if and only if for all $\mathbb{D} \succ \mathbb{C}$ we have $R_\mathbb{D} \models \mathbb{C}$, if and only if $R_\infty \models \mathbb{C}$.
2. R_∞ is non-overlapping, terminating and hence canonical.

The crucial difference between our model construction and the standard model construction is the condition on productive closures to be unblocked w.r.t. N. Let us now define our redundancy notions based on $\mathcal{U}(N)$ as follows.

Definition 7 (Redundant Clause/Inference). A constrained clause $\mathcal{C}$ is *redundant w.r.t.* N if every ground instance of $\mathcal{C}$ is either blocked w.r.t. N, or follows from smaller ground closures in $\mathcal{U}(N)$. An inference $\mathcal{C}_1, ..., \mathcal{C}_n \vdash \mathcal{D}$ is *redundant w.r.t.* N if for each θ grounding for $\mathcal{C}_1, \ldots, \mathcal{C}_n$ and $\mathcal{D}$ either

(i) one of $\mathcal{C}_1\cdot\theta, ..., \mathcal{C}_n\cdot\theta, \mathcal{D}\cdot\theta$ is blocked w.r.t. N, or
(ii) $\mathcal{D}\cdot\theta$ follows from the set of ground closures $\{\mathbb{C} \mid \mathbb{C} \in \mathcal{U}(N) \text{ and } \mathcal{C}_i\cdot\theta \succ \mathbb{C} \text{ for some } i\}$. □

Definition 8 (Saturation up to Redundancy). A set of constrained clauses N is *saturated up to redundancy* if, given non-redundant constrained clauses $\mathcal{C}_1, ..., \mathcal{C}_n \in N$, any BLINC inference $\mathcal{C}_1, ..., \mathcal{C}_n \vdash \mathcal{D}$ is redundant w.r.t. N. □

From now on, let N be an arbitrary but fixed set of constrained clauses. We will formulate a sequence of lemmas used in the completeness proof, whose proofs are included in the the full version of the paper [15]. The following lemma is used to show that unary inferences with an unblocked premise result in an unblocked conclusion.

Lemma 1. (Unblocking Inferences) *Suppose $\mathcal{C}, \mathcal{D} \in N$ and θ and σ are substitutions irreducible in $R_{\prec\mathcal{C}\cdot\theta}$ and in $R_{\prec\mathcal{D}\cdot\sigma}$, respectively. If $\mathcal{C}\cdot\theta \succ \mathcal{D}\cdot\sigma$, $\Gamma\theta \supseteq \Delta\sigma$ and $\mathcal{C}\cdot\theta$ is unblocked w.r.t. N, then $\mathcal{D}\cdot\sigma$ is unblocked w.r.t. N.*

We next prove that the conclusion of a blocked inference is redundant, that is, the conditions that block inferences in BLINC are correct.

Lemma 2. (Redundancy with Blocked Clauses) *Let $\mathcal{C}$ be a constrained clause. If $\mathcal{C}$ is blocked, then all ground instances of $\mathcal{C}$ are blocked w.r.t. N.*

The next lemma resembles the standard lemma on counterexample reduction.

Lemma 3 (Unblocked $\mathsf{Sup}_{\ni}$). *Suppose that (a) $\mathcal{D} = \underline{s \bowtie t} \vee D \mid \Gamma$ is a constrained clause in N, (b) $\mathcal{D} \cdot \theta$ a ground closure unblocked w.r.t. N, (c) θ is irreducible in $R_{\prec \mathcal{D} \cdot \theta}$, (d) $s\theta \succeq t\theta$, (e) $s\theta$ is reducible in $R_{\prec \mathcal{D} \cdot \theta}$.*

Then there exist a constrained clause $(\underline{l \simeq r} \vee C \mid \Pi) \in N$, a $\mathsf{Sup}_{\ni}$-inference

$$(\mathsf{Sup}_{\ni}) \frac{\underline{l \simeq r} \vee C \mid \Pi \quad \underline{s[u] \bowtie t} \vee D \mid \Gamma}{(s[r] \bowtie t \vee C \vee D)\sigma \mid \Delta}$$

and a substitution τ such that (i) $\mathcal{D}\sigma\tau = \mathcal{D}\theta$, (ii) $\underline{l \simeq r} \vee C \mid \Pi \cdot \sigma\tau$ is productive, and $(s[r] \bowtie t \vee C \vee D)\sigma \mid \Delta \cdot \sigma\tau$ is unblocked w.r.t. N.

We are now ready to show completeness of BLINC, starting with the following.

Theorem 1 (Model of $\mathcal{U}(N)$). *Let N be saturated up to redundancy and $\square \notin N$. Then for each $\mathbb{C} \in \mathcal{U}(N)$ we have $R_{\mathbb{C}} \models \mathbb{C}$.*

When $R_{\mathbb{C}} \models \mathbb{C}$, we will simply say that $\mathbb{C}$ is true. Note that this implies that $R_{\mathbb{D}} \models \mathbb{C}$ for all $\mathbb{D} \succeq \mathcal{C}$, and also $R_{\infty} \models \mathbb{C}$. We say that $\mathbb{C}$ is false if it not true.

Here, we only prove a few representative cases and refer to [15] for complete argumentation. Assume, by contradiction, that $\mathcal{U}(N)$ contains a ground closure $\mathbb{C}$ such that $R_{\mathbb{C}} \not\models \mathbb{C}$. Since $\succ$ is well-founded, then N^* contains a minimal unblocked closure $\mathcal{C} \cdot \theta$ such that $R_{\mathcal{C} \cdot \theta} \not\models \mathcal{C} \cdot \theta$.

Case 1. *$\mathcal{C}$ is redundant w.r.t. N.*

Proof. The closure $\mathcal{C} \cdot \theta$ is unblocked, so it follows from smaller closures $\mathcal{C}_1, \ldots, \mathcal{C}_n$ in $\mathcal{U}(N)$. Then there is some $\mathcal{C}_i$ which is false too, and we are done. □

Case 2. *$\mathcal{C}$ contains a variable x such that $x\theta$ is reducible in $R_{\prec \mathcal{C} \cdot \theta}$.*

Proof. The reduced closure $\mathcal{C} \cdot \theta'$ of $\mathcal{C} \cdot \theta$ is unblocked w.r.t. N, so $\mathcal{C} \cdot \theta' \in \mathcal{U}(N)$. Since $x\theta \succ x\theta'$ and for all variables y different from x we have $y\theta \succeq y\theta'$, we have $\mathcal{C} \cdot \theta \succ \mathcal{C} \cdot \theta'$, then $\mathcal{C} \cdot \theta'$ is true. Since $y\theta = y\theta'$ is true in R_{∞} for all variables y, we also have that $\mathcal{C} \cdot \theta'$ is equivalent to $\mathcal{C} \cdot \theta$ in R_{∞}, hence $\mathcal{C} \cdot \theta$ is true and we obtain a contradiction. □

Case 3. *There is a reductive inference $\mathcal{C}_1, \ldots, \mathcal{C}_n \vdash \mathcal{D}$ with $\mathcal{C}_1, \ldots, \mathcal{C}_n \in N$ which is redundant w.r.t. N such that (a) $\{\mathcal{C}_1 \cdot \theta, \ldots, \mathcal{C}_n \cdot \theta\} \subseteq \mathcal{U}(N)$, (b) $\mathcal{D} \cdot \theta$ is unblocked w.r.t. N, (c) $\mathcal{C} \cdot \theta = \max\{\mathcal{C}_1 \cdot \theta, \ldots, \mathcal{C}_n \cdot \theta\}$, and (d) $\mathcal{D} \cdot \theta \models \mathcal{C} \cdot \theta$.*

Proof. $\mathcal{D} \cdot \theta$ is implied by ground closures in $\mathcal{U}(N)$ smaller than $\mathcal{C} \cdot \theta$. These ground closures are then true in $R_{\mathcal{C} \cdot \theta}$, so $\mathcal{D} \cdot \theta$ is true, and hence $\mathcal{C} \cdot \theta$ is also true in $R_{\mathcal{C} \cdot \theta}$, contradiction. □

Case 4. *None of the previous cases apply, and a negative literal $s \not\simeq t$ is selected in $\mathcal{C}$, i.e. $\mathcal{C} = \underline{s \not\simeq t} \vee C \mid \Gamma$.*

Proof. $\mathcal{C} \cdot \theta$ is false in $R_{\mathcal{C} \cdot \theta}$, so $s\theta \downarrow_{R_{\mathcal{C} \cdot \theta}} t\theta$. W.l.o.g., assume $s\theta \succeq t\theta$.

Subcase 4.1. *$s\theta = t\theta$.*

Proof. Then s and t are unifiable. Consider the $\mathsf{EqRes}_{\ni}$ inference

$$\frac{\underline{s \not\simeq t} \vee C \mid \Gamma}{C\sigma \mid \Gamma\sigma}$$

where $\sigma = \mathsf{mgu}(s, t)$. Take any ground instance $\mathcal{D} \cdot \rho = (C\sigma \mid \Gamma\sigma) \cdot \rho$ such that $\sigma\rho = \theta$; by the idempotence of σ, we have $\mathcal{D} \cdot \rho = \mathcal{D} \cdot \theta$. Clearly, $\mathcal{C} \cdot \theta \succ \mathcal{D} \cdot \theta$ and $\mathcal{D} \cdot \theta$ implies $\mathcal{C} \cdot \theta$. As $\mathcal{C} \cdot \theta \succ \mathcal{D} \cdot \theta$ and $\Gamma\sigma\rho = \Gamma\sigma\theta = \Gamma\theta$, Lemma 1 implies that $\mathcal{D} \cdot \theta$ is unblocked w.r.t. N. By Case 1, $\mathcal{D}$ is not redundant, hence $\mathcal{D} \in N$. But then $\mathcal{D} \cdot \theta$ is a false closure in $\mathcal{U}(N)$, which is strictly smaller than $\mathcal{C} \cdot \theta$, so we obtain a contradiction. □

Subcase 4.2. $s\theta \succ t\theta$.

Proof. By conditions on the literal selection, we assume that $s\theta \succ t\theta$ is maximal in $\mathcal{C}$. By Lemma 3, there is a $\mathsf{Sup}_{\ni}$ inference into $s\theta$ with a ground closure such that the result $\mathcal{C}' \cdot \theta$ is unblocked w.r.t. N. This closure is of the form $\mathcal{D} \cdot \theta = (l \simeq r \vee D \mid \Pi) \cdot \theta$ and we have the following $\mathsf{Sup}_{\ni}$ inference

$$\frac{\underline{l \simeq r} \vee D \mid \Pi \quad \underline{s[l'] \not\simeq t} \vee C \mid \Gamma}{(s[r] \not\simeq t \vee C \vee D)\sigma \mid \Delta}$$

where $\sigma = \mathsf{mgu}(l, l')$. Note that $\mathcal{C}' = s[r] \not\simeq t \vee C \vee D$ and $\mathcal{C}' \cdot \rho = \mathcal{C}' \cdot \theta$. Then, $\mathcal{C} \cdot \theta \succ \mathcal{C}' \cdot \theta$ and $\mathcal{D} \cdot \theta$ and $\mathcal{C}' \cdot \theta$ imply $\mathcal{C} \cdot \theta$. Since $\mathcal{C}' \cdot \theta$ is unblocked w.r.t. N, using Lemma 2, we get that $\mathcal{C}'$ is not blocked w.r.t. N, and condition (5) of $\mathsf{Sup}_{\ni}$ is satisfied. Similarly to Case 4.1, we have that the conclusion is a smaller false unblocked closure, so we obtain a contradiction. □

Next we show that for a saturated set of clauses N, if R_∞ is a model for $\mathcal{U}(N)$, then it is also a model of N^*, that is, R_∞ satisfies also all blocked closures in N^*. This follows from the next theorem.

Theorem 2 (Model of N^*). *Let N be a saturated set of clauses. Every blocked closure $\mathcal{C} \cdot \theta \in N^*$ follows from $\mathcal{U}(N)$.*

Using Theorems 1–2, we obtain completeness of BLINC.

Corollary 1 (Completeness of BLINC). *Let N be saturated up to redundancy. If N does not contain $\square$, then N is satisfiable.*

We conclude with a remark on *constraint inheritance* in BLINC. Note that in the $\mathsf{Sup}_{\ni}$ inference rule of Figure 2, constraints are inherited only from the right premise. It is possible to block more inferences without losing refutational completeness of BLINC, by allowing constraint inheritance from the left premise in the $\mathsf{Sup}_{\ni}$ rule as well. However, we cannot propagate constraints that are non-active in the left premise, as they may become active in the conclusion, making the inference blocked. This effect is illustrated in the following example.

Example 3. Consider a superposition into ① with ③

$$\frac{g(x, b) \simeq a \quad g(a, x) \simeq x}{a \simeq b \mid \{\downarrow a, \downarrow b, g(a, b) \rightsquigarrow a\}}$$

If $b \succ a$, then $\downarrow a$ is the only active constraint in the conclusion. Consider a superposition with ④ where constraints are inherited from both premises:

$$\frac{a \simeq b \mid \{\downarrow a, \downarrow b, g(a,b) \rightsquigarrow a\} \quad P(g(x,y), f(g(x,b),z))}{P(g(x,y), f(g(x,a),z)) \mid \{\downarrow a, \downarrow b, g(a,b) \simeq a, b \rightsquigarrow a\}}$$

In the conclusion, $\downarrow b$ and $b \rightsquigarrow a$ are both active, which blocks the inference. □

5 Redundancy Detection in BLINC

In this section we discuss redundancy detection in BLINC. We give sufficient conditions for a clause to be redundant when inferences of a specific form are applied. As usual, we call a *simplifying inference*, or *simplification*, any inference such that one of the premises becomes redundant after the conclusion is added to the current set of clauses. Inference rules whose instances are simplifications are called *simplification rules*. When we display a simplification rule, we will denote clauses that become redundant by drawing a line through them.

Definition 7 gives rise to two kinds of simplification criteria: (i) based on blocking, and (ii) when one of the premises $\mathcal{C} \cdot \theta$ follows from smaller constrained clauses. The following definition captures the first redundancy criterion.

Definition 9 (Closure/Clause Blocked Relative to Closure/Clause). A ground closure $\mathbb{C}$ is *blocked relative to* a ground closure $\mathbb{D}$ if for every set of constrained clauses N, if $\mathbb{D}$ is blocked w.r.t. N^*, then $\mathbb{C}$ is blocked w.r.t. N^* too. A constrained clause $\mathcal{C}$ is *blocked relative to* a constrained clause $\mathcal{D}$, if every ground instance of $\mathcal{C}$ is blocked relative to some ground instance of $\mathcal{D}$. □

This notion will be used for defining simplification rules. We will next present sufficient conditions for checking that a constrained clause is blocked relative to another constrained clause. For example, each ground closure of a clause $C \mid \emptyset$ is unblocked w.r.t. any set N, hence everything is blocked relative to that ground closure. Further, each ground closure with a reducible substitution is blocked relative to its reduced closure.

Definition 10 (Well-Behaved Constrained Clause). Let $\mathcal{C} = C \mid \Gamma$ be a constrained clause. We say that $\mathcal{C}$ is *well-behaved* if (i) all constraints in Γ are active in C, and for each $\gamma \in \Gamma$, (ii) if $\gamma = \downarrow l$, then $\downarrow u \in \Gamma$ for all $u \lhd l$, and (iii) if $\gamma = l \rightsquigarrow r$, then $\downarrow u \in \Gamma$ for all $u \lhd l$ and l contains all variables of r. □

Example 4. The clause $P(a, f(b,z)) \mid \{\downarrow a, g(a,b) \rightsquigarrow a\}$ is not well-behaved but $P(a, f(b,z)) \mid \{\downarrow a, \downarrow b, g(a,b) \rightsquigarrow a\}$ is. The clause $a \simeq b \mid \{\downarrow a, \downarrow b, g(a,b) \rightsquigarrow a\}$ is not well-behaved since it contains constraints not active in the clause. □

Lemma 4. (Relatively Blocked Well-Behavedness) *Let $\mathcal{C} = C \mid \Gamma$ and $\mathcal{D} = D \mid \Delta$ be well-behaved constrained clauses, and σ a substitution. Then $\mathcal{C}$ is blocked relative to $\mathcal{D}$ if $\mathcal{C} \succ \mathcal{D}\sigma$ and $\Gamma \supseteq \Delta\sigma$.*

In the sequel, we assume that each constrained clause is well-behaved. We next adjust two standard simplifications within superposition [14], namely demodulation in Theorem 3 and subsumption in Theorem 4. Our analogue of *demodulation* is the following special case of $\mathsf{Sup}_{\ni}$ in BLINC:

$$(\mathsf{Dem}_{\ni})\ \frac{l \simeq r \mid \Delta \quad \cancel{C[l\sigma] \mid \Gamma}}{C[r\sigma] \mid \Gamma} \quad \text{where} \quad \begin{array}{l}(1)\ l\sigma \succ r\sigma,\\ (2)\ C[l\sigma] \succ (l \simeq r)\sigma,\\ (3)\ \Delta\sigma \subseteq \Gamma.\end{array}$$

Theorem 3. (BLINC Demodulation) $\mathsf{Dem}_{\ni}$ *is a simplification rule. That is,* $C[l\sigma] \mid \Gamma$ *is redundant w.r.t. any constrained clause set that contains* $l \simeq r \mid \Delta$ *and* $C[r\sigma] \mid \Gamma$*.*

In addition to simplification rules, we will also consider *deletion rules*. These rules delete a (redundant) constrained clause from N provided that N contains another constrained clause or set of constrained clauses. The below deletion rule is our analogue of *subsumption*:

$$(\mathsf{Subs}_{\ni})\ \frac{D \mid \Delta \quad \cancel{C \mid \Gamma}}{} \quad \text{where} \quad \begin{array}{l}(1)\ D\sigma \subsetneq C,\\ (2)\ \Delta\sigma \subseteq \Gamma,\end{array} \quad \text{for some substitution } \sigma.$$

Theorem 4. (BLINC Subsumption) $\mathsf{Subs}_{\ni}$ *is a deletion rule. That is,* $C \mid \Gamma$ *is redundant w.r.t. any constrained clause set that contains* $D \mid \Delta$*.*

We also introduce two deletion rules based on properties of the constraints of a clause. Namely, in Theorem 5 we introduce a deletion rule resembling "basic blocking" [25], whereas Theorem 6 exploits deletion based on rewrite orders. Consider therefore the following rule:

$$(\mathsf{Block}_{\ni})\ \frac{l \simeq r \mid \Delta \quad \cancel{C \mid \Gamma}}{} \quad \text{where} \quad \begin{array}{l}(1)\ C \succ (l \simeq r)\sigma \text{ and } l\sigma \succ r\sigma,\\ (2)\ \Delta\sigma \subseteq \Gamma,\\ (3)\ \text{either (i) } {\downarrow}l\sigma \in \Gamma\\ \quad \text{or (ii) } l\sigma \rightsquigarrow r' \in \Gamma \text{ and } r' \succ r\sigma.\end{array}$$

Theorem 5. (BLINC Blocking) $\mathsf{Block}_{\ni}$ *is a deletion rule. That is,* $C \mid \Gamma$ *is redundant w.r.t. any constrained clause that contains* $l \simeq r \mid \Delta$*.*

Our last deletion inference relies on the fact that all rewrite rules in any partial model have to be oriented left-to-right according to $\succ$. That is,

$$(\mathsf{Orient}_{\ni})\ \frac{\cancel{C \mid \Gamma \cup \{l \rightsquigarrow r\}}}{} \quad \text{where} \quad \begin{array}{l}(1)\ r \succ l,\\ (2)\ C \succ (l \simeq r).\end{array}$$

Theorem 6. (BLINC Orientation) $\mathsf{Orient}_{\ni}$ *is a deletion rule. That is,* $C \mid \Gamma \cup \{l \rightsquigarrow r\}$ *is redundant w.r.t. any constrained clause set.*

We illustrate the above simplification and deletion rules with the following example.

Example 5. Consider the following well-behaved constrained clauses:

$$\begin{array}{ll} (1)\ P(g(a,x),b) \mid \{{\downarrow}b, f(x,b) \rightsquigarrow b\}, & (2)\ P(g(y,z),w) \mid \{f(z,w) \rightsquigarrow b\} \\ (3)\ g(a,z) \simeq b \mid \{{\downarrow}b\}, & (4)\ f(x,y) \simeq a \mid \emptyset \end{array}$$

By Theorem 4, clause (2) subsumes clause (1). By Theorem 3, clause (1) can be simplified with clause (3) into $P(b,b) \mid \{{\downarrow}b, f(x,b) \rightsquigarrow a\}$. Finally, assuming $b \succ a$, clauses (1) and (2) are redundant w.r.t. clause (4) by Theorem 5. □

Remark 1. **(Simplification Heuristics via Unblocking)** We note that further simplifications (and heuristics) can be implemented by removing constraints from constrained clauses. This process of removing constraints is captured via the following rule:

$$(\mathsf{Unblock})\ \frac{C \mid \cancel{\Gamma}}{C \mid \Delta}\ \text{where}\ \ \Delta \subset \Gamma.$$

Clearly, Unblock is a simplification rule, as removing constraints from a constrained clause preserves completeness in BLINC. □

We note that using the general notion of well-behaved clauses and Lemma 4, any further redundancy elimination technique can be adapted to BLINC. We conclude this section by showing that Theorems 3–6 can be adjusted and combined using the ground redundancy of Definition 7. This results in stronger redundancy detection, as the following example illustrates.

Example 6. Consider the following $\mathsf{Sup}_{\Subset}$ inference:

$$\frac{g(f(v,w),a) \simeq g(w,a) \mid \emptyset \quad f(g(f(x,y),z),f(y,x)) \simeq z \mid \emptyset}{f(g(y,a),f(y,x)) \simeq a \mid \Delta}\,\sigma = \left\{\begin{array}{c} v \mapsto x, \\ w \mapsto y, \\ z \mapsto a \end{array}\right\},$$

where $\Delta = \{{\downarrow}f(x,y), {\downarrow}f(y,x), {\downarrow}a, g(f(x,y),a) \rightsquigarrow g(y,a)\}$. Note that the conclusion is a well-behaved constrained clause. The conclusion cannot be simplified by clauses

$$(1)\ f(x,y) \simeq f(y,x) \quad \text{and} \quad (2)\ f(x,x) \simeq x,$$

using any of Theorems 3–6. However, using similar conditions as in the $\mathsf{Block}_{\Subset}$ deletion rule, we can do the following. Let θ be a substitution that makes the conclusion ground. By a comparative case distinction on $x\theta$ and $y\theta$,

(i) if $x\theta \succ y\theta$, then using clause (1), by ${\downarrow}f(x,y) \in \Delta$ and $f(x,y)\theta \succ f(y,x)\theta$;
(ii) if $x\theta = y\theta$, then using clause (2) by ${\downarrow}f(x,y) \in \Delta$ (or ${\downarrow}f(y,x) \in \Delta$), $f(x,y)\theta = f(x,x)\theta \succ x\theta$ (or $f(y,x)\theta = f(x,x)\theta \succ x\theta$); and
(iii) if $x\theta \prec y\theta$, then using clause (1) again, by ${\downarrow}f(y,x) \in \Delta$ and $f(y,x)\theta \succ f(x,y)\theta$;

we conclude that the ground closure $(f(g(y,a),f(y,x)) \simeq a \mid \Delta) \cdot \theta$ is redundant in all cases, hence the conclusion is redundant w.r.t. clauses (1) and (2). □

Variant	UEQ		PEQ	
	Solved	Uniques	Solved	Uniques
`baseline`	778	15	1276	34
`blinc1`	316	0	411	0
`blinc2`	327	0	425	0
`blinc3`	610	0	809	0
`blinc4`	775	13	1270	28

Fig. 4. Experimental comparison using variants BLINC in Vampire, using 1455 UEQ problems and 2422 PEQ problems.

6 Evaluation

We implemented[5] BLINC in Vampire [21], together with the simplification rules of Section 5. We have also implemented a redundancy check called *orderedness* that eagerly checks if the result of a superposition can be deleted. We experimented with several variants of BLINC with redundancy elimination (all techniques discussed in Section 5), using different heuristics for removing constraints from clauses via Unblock: (i) `blinc1` does not use Unblock; (ii) `blinc2` uses Unblock to remove constraints inherited from premises, hence only conclusions of $\mathsf{Sup}_{\ni}$ will contain constraints; (iii) `blinc3` uses Unblock occasionally on the clause that would simplify the most clauses in the search space when unconstrained; (iv) `blinc4` uses Unblock on all clauses at activation. We compare these to standard superposition (`baseline`).

Solving unit equality (UEQ) problems is still very hard for superposition-based theorem provers, a claim substantiated by results in the CADE ATP System Competition (CASC) [30]. For this reason, our evaluation focused on the UEQ domain of the TPTP benchmark suite, version 8.1.2 [31]. Since our work does not consider (variants of) resolution, but proper superposition, we also restricted further evaluation to the pure equality (PEQ) benchmarks of TPTP. As a result, our experiments use all benchmarks of the unit equality (UEQ) and pure equality (PEQ) divisions from TPTP version 8.1.2 [31].

All our experiments are based on a DISCOUNT saturation loop [11] and a Knuth-Bendix ordering, with a timeout of 100 seconds and without AVATAR [32]. Our results are summarized in Figure 4. The results show that `blinc1` performs poorly compared to `baseline`, `blinc3` and `blinc4`, and that `blinc2` performs only slightly better than `blinc1`. The variant `blinc3` performs much better than `blinc1` and `blinc2` but it is still does not solve any new problems. The variant blinc4 performs comparably to the state-of-the-art baseline but solves different problems, 28 uniquely. Our preliminary results are therefore encouraging for complementing state-of-the-art superposition proving with BLINC reasoning, possibly in a portfolio solver.

We also analysed the impact of BLINC variants on skipping superposition inferences during proof search. Figure 5 shows the distribution of benchmarks

[5] `https://github.com/vprover/vampire/commit/9c42b448996947e8`

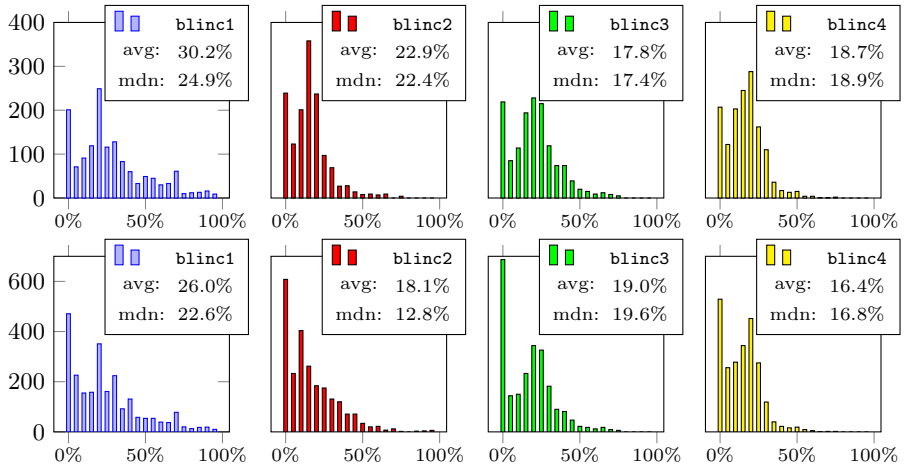

Fig. 5. Distribution of UEQ (top) and PEQ (bottom) benchmarks by ratio of skipped superpositions to all superpositions, showing also average (avg) and median (mdn). For example, using `blinc1`, on average 30.2%, resp. 26.0% of superpositions can be skipped in UEQ, resp. PEQ benchmarks.

by percentage of skipped superposition inferences among all superposition inferences during our runs for `blinc` variants. `blinc1` skips more than half of superposition inferences in a significant number of benchmarks, while the least restrictive `blinc4` still reduces the number of superposition inferences by a significant amount in most benchmarks.

7 Related work

The basicness restriction [27,16] was extended to first-order logic, for example, in *basic superposition* [26] and *basic paramodulation* [7]. The former uses ground unification, the latter closures and variable abstraction to capture irreducibility constraints. In basic paramodulation, *redex orderings* are used similarly to S-orderings in our framework. BLINC expresses more fine-grained blocking, for example, distinguishing between different superpositions on the same term. Related notions in basic superposition have also been formalized [33].

Several critical pair criteria in completion-based theorem proving use irreducibility notions. *Blocking* [4] is similar to basicness, while *compositeness* [4,17] forbids any superpositions into terms with reducible subterms. *General superposition* [35,36] avoids superpositions when more general ones or ones symmetric in variables have been performed. Our BLINC framework handles all such restrictions. These criteria are instances of the *connectedness* criterion [3], which has been also explored in *ground joinability* [1], *ground reducibility* [22] and *ground connectedness* [13].

More general irreducibility constraints were considered in completion [23] and in superposition [18], the latter using a semantic tree method for completeness. Ordering constraints [9,10,20] and unification constraints [8,28] have also been considered, usually moving them to the calculus level. Extending and generalizing our BLINC framework with such constraints is a future challenge.

8 Conclusions

We introduce reducibility constraints to block inferences during superposition reasoning. Our resulting BLINC calculus is refutationally complete and is extended with redundancy elimination, allowing us to maintain efficient reasoning when compared to state-of-the-art superposition proving. Integrating our approach with further inference-blocking constraints, such as blocking more general or outermost superpositions, is an interesting line for future work. Adapting our framework to domain-specific inference rules, e.g. in linear arithmetic or higher-order superposition, is another line for future work.

Other interesting directions are (i) the use of a stronger semantics of constraints, as in Definition 10, and (ii) a "hybrid calculus", improving on `blinc3`, where we still use constraints for blocking generating inferences but relax them whenever they prevent us from applying a simplification or a deletion rule.

Acknowledgements. We thank Konstantin Korovin for fruitful discussions. We acknowledge funding from the ERC Consolidator Grant ARTIST 101002685, the TU Wien SecInt Doctoral College, the FWF SFB project SpyCoDe F8504, the WWTF ICT22-007 grant ForSmart, and the Amazon Research Award 2023 QuAT.

References

1. Avenhaus, J., Hillenbrand, T., Löchner, B.: On Using Ground Joinable Equations in Equational Theorem Proving. Journal of Symbolic Computation **36**(1), 217–233 (2003). https://doi.org/10.1016/S0747-7171(03)00024-5
2. Baader, F., Nipkow, T.: Equational problems. In: Term Rewriting and All That, p. 58–92. Cambridge University Press (1998). https://doi.org/10.1017/CBO9781139172752
3. Bachmair, L.: Canonical Equational Proofs. Progress in theoretical computer science, Birkhäuser (1991). https://doi.org/10.1007/978-1-4684-7118-2
4. Bachmair, L., Dershowitz, N.: Critical Pair Criteria for Completion. Journal of Symbolic Computation **6**(1), 1–18 (1988). https://doi.org/10.1016/S0747-7171(88)80018-X
5. Bachmair, L., Ganzinger, H.: Equational Reasoning in Saturation-Based Theorem Proving. In: Bibel, W., Schmitt, P.H. (eds.) Automated Deduction: A Basis for Applications, vol. I, chap. 11, pp. 353–397. Springer (1998). https://doi.org/10.1007/978-94-017-0437-3

6. Bachmair, L., Ganzinger, H.: Resolution Theorem Proving. In: Handbook of Automated Reasoning, pp. 19–99. Elsevier and MIT Press (2001). https://doi.org/10.1016/B978-044450813-3/50004-7
7. Bachmair, L., Ganzinger, H., Lynch, C., Snyder, W.: Basic Paramodulation and Superposition. In: CADE. pp. 462–476 (1992). https://doi.org/10.1007/3-540-55602-8_185
8. Bhayat, A., Schoisswohl, J., Rawson, M.: Superposition with Delayed Unification. In: CADE. pp. 23–40 (2023). https://doi.org/10.1007/978-3-031-38499-8_2
9. Comon, H.: Solving Symbolic Ordering Constraints. International Journal of Foundations of Computer Science **01**(04), 387–411 (1990). https://doi.org/10.1142/S0129054190000278
10. Comon, H., Nieuwenhuis, R., Rubio, A.: Orderings, AC-Theories and Symbolic Constraint Solving (Extended Abstract). In: Annual IEEE Symposium on Logic in Computer Science. pp. 375–385 (1995). https://doi.org/10.1109/LICS.1995.523272
11. Denzinger, J., Kronenburg, M., Schulz, S.: DISCOUNT - A Distributed and Learning Equational Prover. J. Autom. Reason. **18**(2), 189–198 (1997). https://doi.org/10.1023/A:1005879229581
12. Dershowitz, N., Manna, Z.: Proving Termination with Multiset Orderings. Commun. ACM **22**(8), 465–476 (aug 1979). https://doi.org/10.1145/359138.359142
13. Duarte, A., Korovin, K.: Ground Joinability and Connectedness in the Superposition Calculus. In: IJCAR. pp. 169–187 (2022). https://doi.org/10.1007/978-3-031-10769-6_11
14. Gleiss, B., Kovács, L., Rath, J.: Subsumption Demodulation in First-Order Theorem Proving. In: IJCAR. pp. 297–315 (2020). https://doi.org/10.1007/978-3-030-51074-9_17
15. Hajdu, M., Kovács, L., Rawson, M., Voronkov, A.: Reducibility constraints in superposition. EasyChair Preprint no. 12142 (EasyChair, 2024)
16. Hullot, J.M.: Canonical Forms and Unification. In: CADE. pp. 318–334 (1980). https://doi.org/10.1007/3-540-10009-1_25
17. Kapur, D., Musser, D.R., Narendran, P.: Only Prime Superpositions Need be Considered in the Knuth-Bendix Completion Procedure. Journal of Symbolic Computation **6**(1), 19–36 (1988). https://doi.org/10.1016/S0747-7171(88)80019-1
18. Kirchner, C., Kirchner, H., Rusinowitch, M.: Deduction with Symbolic Constraints. Research Report RR-1358, INRIA (1990)
19. Knuth, D.E., Bendix, P.B.: Simple word problems in universal algebras. In: Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970, pp. 342–376. Springer (1983). https://doi.org/10.1007/978-3-642-81955-1_23
20. Korovin, K., Voronkov, A.: Knuth-Bendix Constraint Solving Is NP-Complete. In: Automata, Languages and Programming. pp. 979–992 (2001). https://doi.org/10.1007/3-540-48224-5_79
21. Kovács, L., Voronkov, A.: First-Order Theorem Proving and Vampire. In: CAV. pp. 1–35 (2013). https://doi.org/10.1007/978-3-642-39799-8_1
22. Löchner, B.: A Redundancy Criterion Based on Ground Reducibility by Ordered Rewriting. In: IJCAR. pp. 45–59 (2004). https://doi.org/10.1007/978-3-540-25984-8_2
23. Lynch, C., Snyder, W.: Redundancy Criteria for Constrained Completion. In: RTA. pp. 2–16 (1993). https://doi.org/10.1007/978-3-662-21551-7_2
24. McCune, W.: Solution of the Robbins Problem. Journal of Automated Reasoning **19**, 263–276 (1997). https://doi.org/10.1023/A:1005843212881

25. Nieuwenhuis, R., Rubio, A.: Paramodulation-Based Theorem Proving. In: Handbook of Automated Reasoning, vol. I, chap. 7, pp. 371–443. Elsevier and MIT Press (2001). https://doi.org/10.1016/B978-044450813-3/50009-6
26. Nieuwenhuis, R., Rubio, A.: Basic Superposition is Complete. In: ESOP. pp. 371–389 (1992). https://doi.org/10.1007/3-540-55253-7_22
27. Nutt, W., Réty, P., Smolka, G.: Basic Narrowing Revisited. Journal of Symbolic Computation **7**(3-4), 295–317 (1989). https://doi.org/10.1016/S0747-7171(89)80014-8
28. Reger, G., Suda, M., Voronkov, A.: Unification with Abstraction and Theory Instantiation in Saturation-Based Reasoning. In: TACAS. pp. 3–22 (2018). https://doi.org/10.1007/978-3-319-89960-2_1
29. Schulz, S., Cruanes, S., Vukmirović, P.: Faster, Higher, Stronger: E 2.3. In: CADE. pp. 495–507 (2019). https://doi.org/10.1007/978-3-030-29436-6_29
30. Sutcliffe, G.: The CADE ATP System Competition - CASC. AI Magazine **37**(2), 99–101 (2016)
31. Sutcliffe, G.: The Logic Languages of the TPTP World. Logic Journal of the IGPL (2022). https://doi.org/10.1093/jigpal/jzac068
32. Voronkov, A.: AVATAR: The Architecture for First-Order Theorem Provers. In: CAV. pp. 696–710 (2014). https://doi.org/10.1007/978-3-319-08867-9_46
33. Waldmann, U., Tourret, S., Robillard, S., Blanchette, J.: A comprehensive framework for saturation theorem proving. J. Autom. Reason. **66**(4), 499–539 (2022). https://doi.org/10.1007/S10817-022-09621-7
34. Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: SPASS Version 3.5. In: CADE. pp. 140–145 (2009). https://doi.org/10.1007/978-3-642-02959-2_10
35. Zhang, H., Kapur, D.: Consider only General Superpositions in Completion Procedures. In: RTA. pp. 513–527 (1989). https://doi.org/10.1007/3-540-51081-8_129
36. Zhang, H., Kapur, D.: Unnecessary Inferences in Associative-Commutative Completion Procedures. In: Mathematical Systems Theory (1990). https://doi.org/10.1007/BF02090774