A Comprehensive Framework for Saturation Theorem Proving (Technical Report)

Uwe Waldmann[1(✉)], Sophie Tourret[1], Simon Robillard[2], and Jasmin Blanchette[3,1,4]

[1] Max-Planck-Institut für Informatik, Saarland Informatics Campus, Saarbrücken, Germany
{uwe,stourret,jblanche}@mpi-inf.mpg.de
[2] IMT Atlantique, Nantes, France
simon.robillard@imt-atlantique.fr
[3] Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
[4] Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

Abstract. One of the indispensable operations of realistic saturation theorem provers is (backward and forward) deletion of subsumed formulas. In presentations of proof calculi, however, this is usually discussed only informally, and in the rare cases where there is a formal exposition, it is typically clumsy. This is because the equivalence of dynamic and static refutational completeness holds only for derivations where all deleted formulas are redundant, but using a standard notion of redundancy, a clause C does not make an instance $C\sigma$ redundant.

We present a framework for formal refutational completeness proofs of abstract provers that implement saturation calculi, such as ordered resolution or superposition. The framework relies on modular extensions of lifted redundancy criteria. It permits us to extend redundancy criteria so that they cover subsumption, and also to model entire prover architectures in such a way that the static refutational completeness of a calculus immediately implies the dynamic refutational completeness of, e.g., an Otter or DISCOUNT loop prover implementing the calculus. Our framework is mechanized in Isabelle/HOL.

1 Introduction

In their *Handbook of Automated Reasoning* chapter [5, Sect. 4], Bachmair and Ganzinger remark that

> unfortunately, comparatively little effort has been devoted to a formal analysis of redundancy and other fundamental concepts of theorem proving strategies, while more emphasis has been placed on investigating the refutational completeness of a variety of modifications of inference rules, such as resolution.

As a remedy, they present an abstract framework for saturation up to redundancy. Briefly, theorem proving derivations take the form $N_0 \rhd N_1 \rhd \cdots$, where N_0 is the initial clause set and each step either adds inferred clauses or deletes redundant clauses. Given a suitable notion of fairness, the limit N_* of a fair derivation is saturated up to redundancy. If the calculus is refutationally complete and N_* does not contain the false clause $\bot$, then N_0 has a model.

Bachmair and Ganzinger also define a concrete prover, RP, based on a first-order ordered resolution calculus and the given clause procedure. However, like all realistic resolution provers, RP implements subsumption deletion. This operation is not covered by the standard definition of redundancy, according to which a clause C is redundant w.r.t. a clause set N if all its ground instances $C\theta$ are entailed by *strictly* smaller ground instances of clauses belonging to N. As a result, RP-derivations are *not* $\rhd$-derivations, and the framework is *not* applicable.

There are two ways to address this problem. In the *Handbook*, Bachmair and Ganzinger start from scratch and prove the dynamic refutational completeness of RP by relating nonground derivations to ground derivations. This proof, though, turns out to be rather nonmodular—it refers simultaneously to properties of the calculus, to properties of the prover, and to the fairness of the derivations. Extending it to other calculi or prover architectures would be costly. As a result, most authors stop after proving static refutational completeness of their calculi.

An alternative approach is to extend the redundancy criterion so that subsumed clauses become redundant. As demonstrated by Bachmair and Ganzinger in 1990 [3], this is possible by redefining redundancy in terms of closures (C, θ) instead of ground instances $C\theta$. We show that this approach can be generalized and modularized: First, any redundancy criterion that is obtained by lifting a ground criterion can be extended to a redundancy criterion that supports subsumption without affecting static refutational completeness (Sect. 3). Second, by applying this property to labeled formulas, it becomes possible to give generic completeness proofs for prover architectures in a straightforward way.

Most saturation provers implement a variant of the given clause procedure. We present an abstract version of the procedure (Sect. 4) that can be refined to obtain an Otter [18] or DISCOUNT [1] loop and prove it refutationally complete. We also present a generalization that decouples scheduling and computation of inferences, to support orphan deletion [15, 25] and dovetailing [10].

When these provers are instantiated with a concrete saturation calculus, the dynamic refutational completeness of the combination follows in a modular way from the properties of the prover architecture and the static refutational completeness proof for the calculus. Thus, the framework is applicable to a wide range of calculi, including ordered resolution [5], unfailing completion [2], standard superposition [4], constraint superposition [19], theory superposition [27], hierarchic superposition [7], and clausal higher-order superposition [10].

When Schlichtkrull, Blanchette, Traytel, and Waldmann [24] mechanized Bachmair and Ganzinger's chapter using the Isabelle/HOL proof assistant [21], they found quite a few mistakes, including one that compromised RP's static refutational completeness. This motivated us to mechanize our framework as well (Sect. 5). Identifiers are given in the margin for crossreference.

2 Preliminaries

Our framework is parameterized by abstract notions of formulas, inferences, and redundancy criteria, defined below. We also introduce various auxiliary concepts, notably static and dynamic refutational completeness, and study variations found in the literature.

2.1 Inferences and Redundancy

Let A be a set. An A-sequence is a finite sequence $(a_i)_{i=0}^{k} = a_0, a_1, \ldots, a_k$ or an infinite sequence $(a_i)_{i=0}^{\infty} = a_0, a_1, \ldots$ with $a_i \in A$ for all indices i. We use the notation $(a_i)_{i\geq 0}$ or $(a_i)_i$ for both finite and infinite sequences. A nonempty sequence $(a_i)_i$ can be decomposed into a head a_0 and a tail $(a_i)_{i\geq 1}$. Given $\rhd \subseteq A \times A$, a $\rhd$-derivation is an A-sequence such that $a_i \rhd a_{i+1}$ for all $0 \leq i < k-1$ (for finite sequences) or for all $0 \leq i$ (for infinite sequences). A $\rhd$-derivation is *full* if it is infinite or it has length k and $a_k \not\rhd a$ for all $a \in A$.

A set $\mathbf{F}$ of *formulas* is a nonempty set with a nonempty subset $\mathbf{F}_\bot \subseteq \mathbf{F}$. Elements of $\mathbf{F}_\bot$ represent *false*. Typically, $\mathbf{F}_\bot$ is a singleton—i.e., $\mathbf{F}_\bot = \{\bot\}$. The possibility to distinguish between several *false* elements will be useful when we model concrete prover architectures, where different elements of $\mathbf{F}_\bot$ represent different situations in which a contradiction has been derived.

A *consequence relation* $\models$ over $\mathbf{F}$ is a relation $\models \subseteq \mathcal{P}(\mathbf{F}) \times \mathcal{P}(\mathbf{F})$ with the following properties for all $N_1, N_2, N_3 \subseteq \mathbf{F}$:

(C1) $\{\bot\} \models N_1$ for every $\bot \in \mathbf{F}_\bot$;
(C2) $N_2 \subseteq N_1$ implies $N_1 \models N_2$;
(C3) if $N_1 \models \{C\}$ for every $C \in N_2$, then $N_1 \models N_2$;
(C4) if $N_1 \models N_2$ and $N_2 \models N_3$, then $N_1 \models N_3$.

It is easy to show that (C2)–(C4) imply that $N_1 \models N_2$ if and only if $N_1 \models \{C\}$ for every $C \in N_2$, and that $N \models \bigcup_{i\in I} N_i$ if and only if $N \models N_i$ for every $i \in I$. Moreover, all elements of $\mathbf{F}_\bot$ are logically equivalent—i.e., if $N \models \{\bot\}$ for some $\bot \in \mathbf{F}_\bot$, then $N \models \{\bot'\}$ for every $\bot' \in \mathbf{F}_\bot$.

Consequence relations are used (1) when one discusses the soundness of a calculus (and hence, when we justify the addition of formulas) and (2) when one discusses the refutational completeness of a calculus (and hence, when we justify the deletion of redundant formulas). Somewhat surprisingly, the consequence relations used for these purposes may be different ones. A typical example is theory superposition, where one may use entailment w.r.t. all theory axioms for (1), but only entailment w.r.t. a subset of the (instances of the) theory axioms for (2). Another example is constraint superposition, where one uses entailment w.r.t. the set of all ground instances for (1), but entailment w.r.t. a subset of those instances for (2). Usually, the consequence relation $\models\!\!\!\approx$ that is used for (1) is the intended one, and some additional calculus-dependent argument is necessary to show that refutational completeness w.r.t. the consequence relation $\models$ that is used for (2) implies refutational completeness w.r.t. $\models\!\!\!\approx$.

An $\mathbf{F}$-*inference* ι is a tuple $(C_n, \ldots, C_0) \in \mathbf{F}^{n+1}$, $n \geq 0$. The formulas C_n, $\ldots, C_1$ are called *premises* of ι; C_0 is called the *conclusion* of ι, denoted by $\mathit{concl}(\iota)$. An $\mathbf{F}$-*inference system* Inf is a set of $\mathbf{F}$-inferences. If $N \subseteq \mathbf{F}$, we write $\mathit{Inf}(N)$ for the set of all inferences in Inf whose premises are contained in N, and $\mathit{Inf}(N, M) := \mathit{Inf}(N \cup M) \setminus \mathit{Inf}(N \setminus M)$ for the set of all inferences in Inf such that one premise is in M and the other premises are contained in $N \cup M$.

An inference system Inf is called *sound* w.r.t. a consequence relation $\mathrel{|\!\approx}$ if $\{C_i \mid 1 \leq i \leq n\} \mathrel{|\!\approx} \{C_0\}$ for every inference $(C_n, \ldots, C_0) \in \mathit{Inf}$.

A *redundancy criterion* for an inference system Inf and a consequence relation $\models$ is a pair $\mathit{Red} = (\mathit{Red}_\mathrm{I}, \mathit{Red}_\mathrm{F})$, where $\mathit{Red}_\mathrm{I} : \mathcal{P}(\mathbf{F}) \to \mathcal{P}(\mathit{Inf})$ and $\mathit{Red}_\mathrm{F} : \mathcal{P}(\mathbf{F}) \to \mathcal{P}(\mathbf{F})$ are mappings from sets of formulas to sets of inferences and from sets of formulas to sets of formulas that satisfy the following conditions for all sets of formulas N and N':

- (R1) if $N \models \{\bot\}$ for some $\bot \in \mathbf{F}_\bot$, then $N \setminus \mathit{Red}_\mathrm{F}(N) \models \{\bot\}$;
- (R2) if $N \subseteq N'$, then $\mathit{Red}_\mathrm{F}(N) \subseteq \mathit{Red}_\mathrm{F}(N')$ and $\mathit{Red}_\mathrm{I}(N) \subseteq \mathit{Red}_\mathrm{I}(N')$;
- (R3) if $N' \subseteq \mathit{Red}_\mathrm{F}(N)$, then $\mathit{Red}_\mathrm{F}(N) \subseteq \mathit{Red}_\mathrm{F}(N \setminus N')$ and $\mathit{Red}_\mathrm{I}(N) \subseteq \mathit{Red}_\mathrm{I}(N \setminus N')$;
- (R4) if $\iota \in \mathit{Inf}$ and $\mathit{concl}(\iota) \in N$, then $\iota \in \mathit{Red}_\mathrm{I}(N)$.

Inferences in $\mathit{Red}_\mathrm{I}(N)$ and formulas in $\mathit{Red}_\mathrm{F}(N)$ are called *redundant* w.r.t. N.[1] We define the relation $\rhd_{\mathit{Red}} \subseteq \mathcal{P}(\mathbf{F}) \times \mathcal{P}(\mathbf{F})$ such that $N \rhd_{\mathit{Red}} N'$ if and only if $N \setminus N' \subseteq \mathit{Red}_\mathrm{F}(N')$.

lem:red-concl-implies-red-inf

Lemma 1. *If $\iota \in \mathit{Inf}$ and $\mathit{concl}(\iota) \in \mathit{Red}_\mathrm{F}(N)$, then $\iota \in \mathit{Red}_\mathrm{I}(N)$.*

Proof. Let $\iota \in \mathit{Inf}$ and $\mathit{concl}(\iota) \in \mathit{Red}_\mathrm{F}(N)$. Then $\iota \in \mathit{Red}_\mathrm{I}(\mathit{Red}_\mathrm{F}(N)) \subseteq \mathit{Red}_\mathrm{I}(N \cup \mathit{Red}_\mathrm{F}(N))$. Since $\mathit{Red}_\mathrm{F}(N) \setminus N \subseteq \mathit{Red}_\mathrm{F}(N) \subseteq \mathit{Red}_\mathrm{F}(N \cup \mathit{Red}_\mathrm{F}(N))$, we obtain $\iota \in \mathit{Red}_\mathrm{I}(N \cup \mathit{Red}_\mathrm{F}(N)) \subseteq \mathit{Red}_\mathrm{I}((N \cup \mathit{Red}_\mathrm{F}(N)) \setminus (\mathit{Red}_\mathrm{F}(N) \setminus N)) = \mathit{Red}_\mathrm{I}(N)$. □

2.2 Refutational Completeness

Let $\models$ be a consequence relation, let Inf be an inference system, and let Red be a redundancy criterion w.r.t. Inf and $\models$.

A set $N \subseteq \mathbf{F}$ is called *saturated* w.r.t. Inf and Red if $\mathit{Inf}(N) \subseteq \mathit{Red}_\mathrm{I}(N)$. The pair $(\mathit{Inf}, \mathit{Red})$ is called *statically refutationally complete* w.r.t. $\models$ if for every saturated set $N \subseteq \mathbf{F}$ such that $N \models \{\bot\}$ for some $\bot \in \mathbf{F}_\bot$, there exists a $\bot' \in \mathbf{F}_\bot$ such that $\bot' \in N$.

Let $(N_i)_i$ be a $\mathcal{P}(\mathbf{F})$-sequence. Its *limit* is the set $N_* := \bigcup_i \bigcap_{j \geq i} N_j$. Its *union* is the set $N_\infty := \bigcup_i N_i$. A sequence is called *fair* if $\mathit{Inf}(N_*) \subseteq \bigcup_i \mathit{Red}_\mathrm{I}(N_i)$. The pair $(\mathit{Inf}, \mathit{Red})$ is called *dynamically refutationally complete* w.r.t. $\models$ if for every fair $\rhd_{\mathit{Red}}$-derivation $(N_i)_i$ such that $N_0 \models \{\bot\}$ for some $\bot \in \mathbf{F}_\bot$, we have $\bot' \in N_i$ for some i and some $\bot' \in \mathbf{F}_\bot$. Well-known saturation results follow:

lem:nonpersistent-is-redundant

Lemma 2. *If $(N_i)_i$ is a $\rhd_{\mathit{Red}}$-derivation, then $N_\infty \setminus N_* \subseteq \mathit{Red}_\mathrm{F}(N_\infty)$.*

[1] One can find several slightly differing definitions for redundancy criteria, fairness, and saturation in the literature. We discuss the differences in Sect. 2.3. Here we mostly follow Waldmann [27].

Proof. If $C \in N_\infty \setminus N_*$, then there must exist some i such that $C \in N_i \setminus N_{i+1}$. Consequently, $C \in \mathit{Red}_\mathrm{F}(N_{i+1})$. By property (R2), $C \in \mathit{Red}_\mathrm{F}(N_\infty)$. □

lem:redundant-remains-redundant-during-run

Lemma 3. *If $(N_i)_i$ is a $\rhd_{\mathit{Red}}$-derivation, then $\mathit{Red}_\mathrm{I}(N_i) \subseteq \mathit{Red}_\mathrm{I}(N_*)$ and $\mathit{Red}_\mathrm{F}(N_i) \subseteq \mathit{Red}_\mathrm{F}(N_*)$ for every i.*

Proof. By property (R2), $\mathit{Red}_\mathrm{I}(N_i) \subseteq \mathit{Red}_\mathrm{I}(N_\infty)$; by property (R3), $\mathit{Red}_\mathrm{I}(N_\infty) \subseteq \mathit{Red}_\mathrm{I}(N_\infty \setminus (N_\infty \setminus N_*)) = \mathit{Red}_\mathrm{I}(N_*)$. Analogously, $\mathit{Red}_\mathrm{F}(N_i) \subseteq \mathit{Red}_\mathrm{F}(N_\infty) \subseteq \mathit{Red}_\mathrm{F}(N_\infty \setminus (N_\infty \setminus N_*)) = \mathit{Red}_\mathrm{F}(N_*)$. □

lem:N-i-is-persistent-or-redundant

Lemma 4. *If $(N_i)_i$ is a $\rhd_{\mathit{Red}}$-derivation, then $N_i \subseteq N_* \cup \mathit{Red}_\mathrm{F}(N_*)$ for every i.*

Proof. Let $C \in N_i$. If $C \notin N_*$, then there exists some $j \geq i$ such that $C \in N_j \setminus N_{j+1}$. Consequently, $C \in \mathit{Red}_\mathrm{F}(N_{j+1})$ and therefore $C \in \mathit{Red}_\mathrm{F}(N_*)$. □

lem:fairness-implies-saturation

Lemma 5. *If $(N_i)_i$ is a fair $\rhd_{\mathit{Red}}$-derivation, then the limit N_* is saturated w.r.t. Inf and Red.*

Proof. By fairness, every $\iota \in \mathit{Inf}(N_*)$ is contained in $\bigcup_i \mathit{Red}_\mathrm{I}(N_i)$, so there exists some i such that $\iota \in \mathit{Red}_\mathrm{I}(N_i)$, and by the previous lemma, $\iota \in \mathit{Red}_\mathrm{I}(N_*)$. □

lem:static-ref-compl-implies-dynamic

Lemma 6. *If $(\mathit{Inf}, \mathit{Red})$ is statically refutationally complete w.r.t. $\models$, then it is dynamically refutationally complete w.r.t. $\models$.*

Proof. Assume $(\mathit{Inf}, \mathit{Red})$ is statically refutationally complete w.r.t. $\models$, and let $(N_i)_i$ be a $\rhd_{\mathit{Red}}$-derivation. Assume that $N_0 \models \{\bot\}$ for some $\bot \in \mathbf{F}_\bot$. Since $N_0 \subseteq N_\infty$, we get $N_\infty \models N_0 \models \{\bot\}$, and by property (R1), this implies $N_\infty \setminus \mathit{Red}_\mathrm{F}(N_\infty) \models \{\bot\}$. By Lemma 2, we know that $N_\infty \setminus N_* \subseteq \mathit{Red}_\mathrm{F}(N_\infty)$, or equivalently, $N_\infty \setminus \mathit{Red}_\mathrm{F}(N_\infty) \subseteq N_*$; hence $N_* \models N_\infty \setminus \mathit{Red}_\mathrm{F}(N_\infty) \models \{\bot\}$.

If the sequence is fair, then N_* is saturated, so by static refutational completeness, $\bot' \in N_*$ for some $\bot' \in \mathbf{F}_\bot$. Consequently, $\bot' \in N_i$ for some i, implying dynamic refutational completeness. □

In fact, the converse holds as well:

lem:dynamic-ref-compl-implies-static

Lemma 7. *If $(\mathit{Inf}, \mathit{Red})$ is dynamically refutationally complete w.r.t. $\models$, then it is statically refutationally complete w.r.t. $\models$.*

Proof. Assume $(\mathit{Inf}, \mathit{Red})$ is dynamically refutationally complete w.r.t. $\models$, and let $N_0 \subseteq \mathbf{F}$ be saturated w.r.t. Inf and Red. Assume that $N_0 \models \bot$ for some $\bot \in \mathbf{F}_\bot$. Now consider the one-element sequence $(N_i)_{i=0}^0$. Since $N_* = N_0$ and N_0 is saturated, we know that $\mathit{Inf}(N_*) = \mathit{Inf}(N_0) \subseteq \mathit{Red}_\mathrm{I}(N_0) = \bigcup_i \mathit{Red}_\mathrm{I}(N_i)$, so the sequence is fair. By dynamic refutational completeness, this implies $\bot' \in N_0$ for some $\bot' \in \mathbf{F}_\bot$. Therefore $(\mathit{Inf}, \mathit{Red})$ is statically refutationally complete. □

2.3 Variations on a Theme

For some of the notions in Sects. 2.1 and 2.2 one can find alternative definitions in the literature.

Redundancy Criteria. As in Bachmair and Ganzinger's chapter [5, Sect. 4.1], we have specified in condition (R1) of redundancy criteria that the deletion of redundant formulas must preserve inconsistency. Alternatively, one can require that redundant formulas must be entailed by the nonredundant ones—i.e., $N \setminus \mathit{Red}_{\mathrm{F}}(N) \models \mathit{Red}_{\mathrm{F}}(N)$—leading to some obvious changes in Lemmas 6 and 32.

Bachmair and Ganzinger's definition of a redundancy criterion differs from ours in that they require only conditions (R1)–(R3). They call a redundancy criterion *effective* if an inference $\iota \in \mathit{Inf}$ is in $\mathit{Red}_{\mathrm{I}}(N)$ whenever $\mathit{concl}(\iota) \in N \cup \mathit{Red}_{\mathrm{F}}(N)$. As demonstrated by Lemma 1, that condition is equivalent to our condition (R4).

Inferences from Redundant Premises. Should inferences from redundant premises be necessarily redundant themselves, or not? Should we ignore inferences from redundant premises in the definitions of saturation, fairness, and refutational completeness, or not? For each question, both variants can be found in the literature, and in fact there are good arguments for both. On the one hand, one can argue that the saturation of a set of formulas should not depend on the presence or absence of redundant formulas. On the other hand, in any reasonable proof system, formulas are deleted from the set of formulas as soon as they are shown to be redundant, so why should we care whether the set is saturated even if we do not delete formulas that have been proved to be redundant?

We define "reduced" variants of the definitions in Sects. 2.1 and 2.2. A set $N \subseteq \mathbf{F}$ is called *reducedly saturated* w.r.t. Inf and Red if $\mathit{Inf}(N \setminus \mathit{Red}_{\mathrm{F}}(N)) \subseteq \mathit{Red}_{\mathrm{I}}(N)$. The pair $(\mathit{Inf}, \mathit{Red})$ is *reducedly statically refutationally complete* w.r.t. $\models$ if for every reducedly saturated set $N \subseteq \mathbf{F}$ with $N \models \{\bot\}$ for some $\bot \in \mathbf{F}_{\bot}$, there exists a $\bot' \in \mathbf{F}_{\bot}$ such that $\bot' \in N$. A sequence is called *reducedly fair* if $\mathit{Inf}(N_* \setminus \bigcup_i \mathit{Red}_{\mathrm{F}}(N_i)) \subseteq \bigcup_i \mathit{Red}_{\mathrm{I}}(N_i)$. The pair $(\mathit{Inf}, \mathit{Red})$ is *reducedly dynamically refutationally complete* w.r.t. $\models$ if for every reducedly fair $\rhd_{\mathit{Red}}$-derivation $(N_i)_i$ such that $N_0 \models \{\bot\}$ for some $\bot \in \mathbf{F}_{\bot}$, we have $\bot' \in N_i$ for some i and some $\bot' \in \mathbf{F}_{\bot}$. Recall that $\mathit{Inf}(N, M)$ denotes the set of Inf-inferences with at least one premise in M and the others in $N \cup M$. A *reduced redundancy criterion* for Inf and $\models$ is a redundancy criterion $\mathit{Red} = (\mathit{Red}_{\mathrm{I}}, \mathit{Red}_{\mathrm{F}})$ that additionally satisfies $\mathit{Inf}(\mathbf{F}, \mathit{Red}_{\mathrm{F}}(N)) \subseteq \mathit{Red}_{\mathrm{I}}(N)$ for every $N \subseteq \mathbf{F}$.

For reduced redundancy criteria, saturation and reduced saturation agree:

lem:reduced-rc-implies-sat-equiv-reduced-sat

Lemma 8. *If Red is a reduced redundancy criterion, then N is saturated w.r.t. Inf and Red if and only if N is reducedly saturated w.r.t. Inf and Red.*

Proof. If N is saturated w.r.t. Inf and Red, then $\mathit{Inf}(N) \subseteq \mathit{Red}_{\mathrm{I}}(N)$, so $\mathit{Inf}(N \setminus \mathit{Red}_{\mathrm{F}}(N)) \subseteq \mathit{Inf}(N) \subseteq \mathit{Red}_{\mathrm{I}}(N)$, which implies that N is reducedly saturated w.r.t. Inf and Red.

Conversely, assume that N is reducedly saturated w.r.t. Inf and Red—i.e., $\mathit{Inf}(N \setminus \mathit{Red}_{\mathrm{F}}(N)) \subseteq \mathit{Red}_{\mathrm{I}}(N)$. Let $\iota \in \mathit{Inf}(N)$. If no premise of ι is contained in $\mathit{Red}_{\mathrm{F}}(N)$, then $\iota \in \mathit{Inf}(N \setminus \mathit{Red}_{\mathrm{F}}(N)) \subseteq \mathit{Red}_{\mathrm{I}}(N)$. Otherwise $\iota \in \mathit{Inf}(\mathbf{F}, \mathit{Red}_{\mathrm{F}}(N))$, and since Red is reduced, we get again $\iota \in \mathit{Red}_{\mathrm{I}}(N)$. □

cor:reduced-rc-implies-st-ref-comp-equiv-reduced-st-ref-comp

Corollary 9. *If Red is a reduced redundancy criterion, then (Inf, Red) is statically refutationally complete if and only if it is reducedly statically refutationally complete.*

An arbitrary redundancy criterion $Red = (Red_{\mathrm{I}}, Red_{\mathrm{F}})$ can always be extended to a reduced redundancy criterion $Red' = (Red'_{\mathrm{I}}, Red_{\mathrm{F}})$, where Red'_{I} is defined by $Red'_{\mathrm{I}}(N) := Red_{\mathrm{I}}(N) \cup Inf(\mathbf{F}, Red_{\mathrm{F}}(N))$.

lem:red'-is-reduced-redcrit

Lemma 10. *Red' is a reduced redundancy criterion.*

Proof. Since Red_{F} is left unchanged, (R1) and the first parts of (R2) and (R3) are obvious. (R4) holds because $\iota \in Red_{\mathrm{I}}(N) \subseteq Red'_{\mathrm{I}}(N)$ for every inference ι with $concl(\iota) \in N$. Moreover, Red' is clearly reduced. It remains to prove the second parts of (R2) and (R3).

For (R2), assume $N \subseteq N'$. Then $Red_{\mathrm{I}}(N) \subseteq Red_{\mathrm{I}}(N')$ and $Red_{\mathrm{F}}(N) \subseteq Red_{\mathrm{F}}(N')$. Moreover, Inf is clearly monotonic, so $Inf(\mathbf{F}, Red_{\mathrm{F}}(N)) \subseteq Inf(\mathbf{F}, Red_{\mathrm{F}}(N'))$, and therefore $Red'_{\mathrm{I}}(N) \subseteq Red'_{\mathrm{I}}(N')$.

For (R3), assume $N' \subseteq Red_{\mathrm{F}}(N)$. Then $Red_{\mathrm{F}}(N) \subseteq Red_{\mathrm{F}}(N \setminus N')$ and $Red_{\mathrm{I}}(N) \subseteq Red_{\mathrm{I}}(N \setminus N')$. By monotonicity of Inf, we have $Inf(\mathbf{F}, Red_{\mathrm{F}}(N)) \subseteq Inf(\mathbf{F}, N \setminus N')$, so $Red'_{\mathrm{I}}(N) \subseteq Red'_{\mathrm{I}}(N \setminus N')$. □

lem:saturation-red-vs-red'-1

Lemma 11. *If $N \subseteq \mathbf{F}$ is saturated w.r.t. Inf and Red, then N is saturated w.r.t. Inf and Red'.*

Proof. Since $Red_{\mathrm{I}}(N) \subseteq Red'_{\mathrm{I}}(N)$, $Inf(N) \subseteq Red_{\mathrm{I}}(N)$ implies obviously $Inf(N) \subseteq Red'_{\mathrm{I}}(N)$. □

The converse does not hold:

Example 12. Consider a signature consisting of the four propositional variables (or nullary predicate symbols) P, Q, R, S. Let Inf be the set of inferences of the ordered resolution calculus with selection over clauses over the signature. Define Red_{F} such that a clause C is contained in $Red_{\mathrm{F}}(N)$ if it is entailed by clauses in N that are smaller than C. Define Red_{I} such that an inference is contained in $Red_{\mathrm{I}}(N)$ if its conclusion is entailed by clauses in N that are smaller than its largest premise. Then $Red = (Red_{\mathrm{I}}, Red_{\mathrm{F}})$ is a redundancy criterion.

Let N be the set of clauses (1) $\neg\mathsf{Q} \vee \mathsf{P}$, (2) $\neg\mathsf{S} \vee \mathsf{R} \vee \mathsf{Q}$, (3) $\neg\mathsf{S} \vee \mathsf{Q}$, where the atom ordering is $\mathsf{P} > \mathsf{Q} > \mathsf{R} > \mathsf{S}$ and the first literals of (1) and (3) are selected. Due to the selection, $Inf(N)$ contains only a single inference, namely the ordered resolution inference ι between (2) and (1). The largest premise of ι is (1). The premise (2) is entailed by the smaller clause (3) and therefore contained in $Red_{\mathrm{F}}(N)$. Consequently, $\iota \in Red'_{\mathrm{I}}(N)$, which means that N is saturated w.r.t. Red'. On the other hand, the conclusion $\neg\mathsf{S} \vee \mathsf{R} \vee \mathsf{P}$ is not entailed by the clauses in N that are smaller than (1)—i.e., (2) and (3)—so $\iota \notin Red_{\mathrm{I}}(N)$. Therefore, N is not saturated w.r.t. Red.

lem:saturation-red-vs-red'-2

Lemma 13. *The following properties are equivalent for every $N \subseteq \mathbf{F}$:*

(i) *N is reducedly saturated w.r.t. Inf and Red;*

(ii) N *is saturated w.r.t.* Inf *and* Red'*;*
(iii) $N \setminus \mathit{Red}_{\mathrm{F}}(N)$ *is saturated w.r.t.* Inf *and* Red*.*

Proof. To show that (i) implies (ii), assume that N is reducedly saturated w.r.t. Inf and Red—i.e., $\mathit{Inf}(N \setminus \mathit{Red}_{\mathrm{F}}(N)) \subseteq \mathit{Red}_{\mathrm{I}}(N)$. We must show that $\mathit{Inf}(N) \subseteq \mathit{Red}'_{\mathrm{I}}(N)$. Let $\iota \in \mathit{Inf}(N)$. If no premise of ι is contained in $\mathit{Red}_{\mathrm{F}}(N)$, then $\iota \in \mathit{Inf}(N \setminus \mathit{Red}_{\mathrm{F}}(N)) \subseteq \mathit{Red}_{\mathrm{I}}(N)$. Otherwise, $\iota \in \mathit{Inf}(\mathbf{F}, \mathit{Red}_{\mathrm{F}}(N))$. In both cases, we conclude $\iota \in \mathit{Red}'_{\mathrm{I}}(N)$.

To show that (ii) implies (i), assume that N is saturated w.r.t. Inf and Red'—i.e., $\mathit{Inf}(N) \subseteq \mathit{Red}'_{\mathrm{I}}(N)$. We must show that $\mathit{Inf}(N \setminus \mathit{Red}_{\mathrm{F}}(N)) \subseteq \mathit{Red}_{\mathrm{I}}(N)$. Let $\iota \in \mathit{Inf}(N \setminus \mathit{Red}_{\mathrm{F}}(N))$. Observe first that $\iota \in \mathit{Inf}(N \setminus \mathit{Red}_{\mathrm{F}}(N)) \subseteq \mathit{Inf}(N) \subseteq \mathit{Red}'_{\mathrm{I}}(N) = \mathit{Red}_{\mathrm{I}}(N) \cup \mathit{Inf}(\mathbf{F}, \mathit{Red}_{\mathrm{F}}(N))$. Moreover, $\iota \in \mathit{Inf}(N \setminus \mathit{Red}_{\mathrm{F}}(N))$ implies $\iota \notin \mathit{Inf}(\mathbf{F}, \mathit{Red}_{\mathrm{F}}(N))$. Combining both, we get $\iota \in \mathit{Red}_{\mathrm{I}}(N)$.

The equivalence of (i)—i.e., $\mathit{Inf}(N \setminus \mathit{Red}_{\mathrm{F}}(N)) \subseteq \mathit{Red}_{\mathrm{I}}(N)$—and (iii)—i.e., $\mathit{Inf}(N \setminus \mathit{Red}_{\mathrm{F}}(N)) \subseteq \mathit{Red}_{\mathrm{I}}(N \setminus \mathit{Red}_{\mathrm{F}}(N))$—follows from the fact that $\mathit{Red}_{\mathrm{I}}(N) \subseteq \mathit{Red}_{\mathrm{I}}(N \setminus \mathit{Red}_{\mathrm{F}}(N))$ by (R3) and $\mathit{Red}_{\mathrm{I}}(N \setminus \mathit{Red}_{\mathrm{F}}(N)) \subseteq \mathit{Red}_{\mathrm{I}}(N)$ by (R2). □

Even though Red and Red' are not equivalent as far as saturation is concerned, they are equivalent w.r.t. refutational completeness:

thm:reduced-stat-ref-compl

Theorem 14. *The following properties are equivalent:*

(i) $(\mathit{Inf}, \mathit{Red})$ *is statically refutationally complete w.r.t.* $\models$*;*
(ii) $(\mathit{Inf}, \mathit{Red})$ *is reducedly statically refutationally complete w.r.t.* $\models$*;*
(iii) $(\mathit{Inf}, \mathit{Red}')$ *is statically refutationally complete w.r.t.* $\models$*;*
(iv) $(\mathit{Inf}, \mathit{Red}')$ *is reducedly statically refutationally complete w.r.t.* $\models$*.*

Proof. To show that (iii) implies (i), assume that $(\mathit{Inf}, \mathit{Red}')$ is statically refutationally complete. That is, the property

$$N \models \{\bot\} \text{ for some } \bot \in \mathbf{F}_{\bot} \text{ implies } \bot' \in N \text{ for some } \bot' \in \mathbf{F}_{\bot} \qquad (*)$$

holds for every set $N \subseteq \mathbf{F}$ that is saturated w.r.t. Inf and Red'. By Lemma 11, every set $N \subseteq \mathbf{F}$ that is saturated w.r.t. Inf and Red is also saturated w.r.t. Inf and Red', so property $(*)$ holds in particular for every set $N \subseteq \mathbf{F}$ that is saturated w.r.t. Inf and Red.

To show that (i) implies (iii), assume that $(\mathit{Inf}, \mathit{Red})$ is statically refutationally complete. Assume N is saturated w.r.t. Inf and Red' and suppose that $N \models \{\bot\}$ for some $\bot \in \mathbf{F}_{\bot}$. By Lemma 13, $N \setminus \mathit{Red}_{\mathrm{F}}(N)$ is saturated w.r.t. Inf and Red. Furthermore, by (R1), $N \setminus \mathit{Red}_{\mathrm{F}}(N) \models \{\bot\}$. So the static refutational completeness of $(\mathit{Inf}, \mathit{Red})$ implies that $\bot' \in N \setminus \mathit{Red}_{\mathrm{F}}(N)$ for some $\bot' \in \mathbf{F}_{\bot}$; hence $\bot' \in N$. Thus, $(\mathit{Inf}, \mathit{Red}')$ is statically refutationally complete.

The equivalence of (iii) and (iv) follows immediately from Lemma 10 and Corollary 9.

It remains to show the equivalence of (ii) and (iii). Observe that (ii) means that $(*)$ holds for every set $N \subseteq \mathbf{F}$ that is reducedly saturated w.r.t. Inf and Red, and that (iii) means that $(*)$ holds for every set $N \subseteq \mathbf{F}$ that is saturated w.r.t. Inf and Red'. By Lemma 13, these two properties are equivalent. □

The limit of a reducedly fair $\rhd_{Red}$-derivation is a reducedly saturated set.[2] This is proved analogously to Lemma 5:

lem:red-fairness-implies-red-saturation

Lemma 15. *If $(N_i)_i$ is a reducedly fair $\rhd_{Red}$-derivation, then the limit N_* is reducedly saturated w.r.t. Inf and Red.*

Proof. Since $Red_{\mathrm{F}}(N_i) \subseteq Red_{\mathrm{F}}(N_*)$ for every i, we have $Inf(N_* \setminus Red_{\mathrm{F}}(N_*)) \subseteq Inf(N_* \setminus \bigcup_i Red_{\mathrm{F}}(N_i))$. By reduced fairness, every inference ι in $Inf(N_* \setminus Red_{\mathrm{F}}(N_*))$ is contained in $\bigcup_i Red_{\mathrm{I}}(N_i)$. Therefore there exists some i with $\iota \in Red_{\mathrm{I}}(N_i)$, which implies $\iota \in Red_{\mathrm{I}}(N_*)$. □

Lemmas 6 and 7 can then be reproved for reduced static and reduced dynamic refutational completeness. Together with Theorem 14, we obtain this result:

thm:reduced-dyn-ref-compl

Theorem 16. *The properties* (i)–(iv) *of Theorem 14 and the following four properties are equivalent:*

(v) *(Inf, Red) is dynamically refutationally complete w.r.t. $\models$;*
(vi) *(Inf, Red) is reducedly dynamically refutationally complete w.r.t. $\models$;*
(vii) *(Inf, Red') is dynamically refutationally complete w.r.t. $\models$;*
(viii) *(Inf, Red') is reducedly dynamically refutationally complete w.r.t. $\models$.*

Summarizing, we see that there are some differences between the "reduced" and the "nonreduced" approach, but that these differences are restricted to the intermediate notions, notably saturation. As far as (static or dynamic) refutational completeness is concerned, both approaches agree. Furthermore, Theorem 16 demonstrates that we can mix and match definitions from both worlds. Consequently, when we want to build on an existing refutational completeness proof for some saturation calculus, it does not matter which approach has been used there.

Given that the "nonreduced" definitions in Sects. 2.1 and 2.2 are simpler that than the "reduced" ones in the current section, there is usually little reason to prefer the "reduced" ones. For our purposes, a major advantage of the "nonreduced" definitions is that Red_{F} and Red_{I} are separated as much as possible. In particular, our definitions of saturation and static refutational completeness do not depend on redundant formulas, but only on redundant inferences. This property will be crucial for the proof of Theorem 40 in Sect. 3.

Fairness in the Limit. Bachmair and Ganzinger define $(N_i)_i$ to be fair if $concl(Inf(N') \setminus Red_{\mathrm{I}}(N')) \subseteq N_\infty \cup Red_{\mathrm{F}}(N_\infty)$, where $N' = N_* \setminus Red_{\mathrm{F}}(N_*)$ [5, Sect. 4.1]. This is a quite peculiar property. First of all, it is overly complicated: If the conclusion of an inference $\iota \in Inf(N') \setminus Red_{\mathrm{I}}(N')$ is contained in $N_\infty \cup Red_{\mathrm{F}}(N_\infty)$, then $\iota \in Red_{\mathrm{I}}(N_\infty)$, and by Lemma 2, $\iota \in Red_{\mathrm{I}}(N_\infty) \subseteq Red_{\mathrm{I}}(N_\infty \setminus (N_\infty \setminus N_*)) = Red_{\mathrm{I}}(N_*) \subseteq Red_{\mathrm{I}}(N_* \setminus Red_{\mathrm{F}}(N_*)) = Red_{\mathrm{I}}(N')$. But this contradicts the assumption that $\iota \in Inf(N') \setminus Red_{\mathrm{I}}(N')$. So the condition can be simplified to

[2] The limit need not be saturated, though. For instance, in Example 12, the one-element sequence $(N_i)_{i=0}^0$ with $N_0 = N$ is reducedly fair w.r.t. *Red*, and its limit $N_* = N$ is reducedly saturated w.r.t. *Red*, but not saturated w.r.t. *Red*.

$Inf(N') \subseteq Red_{\mathrm{I}}(N')$, and since $Red_{\mathrm{I}}(N') = Red_{\mathrm{I}}(N_* \setminus Red_{\mathrm{F}}(N_*)) = Red_{\mathrm{I}}(N_*)$, this is equivalent to $Inf(N_* \setminus Red_{\mathrm{F}}(N_*)) \subseteq Red_{\mathrm{I}}(N_*)$.

Since $Inf(N_* \setminus Red_{\mathrm{F}}(N_*)) \subseteq Inf(N_* \setminus \bigcup_i Red_{\mathrm{F}}(N_i))$ and $\bigcup_i Red_{\mathrm{I}}(N_i) \subseteq Red_{\mathrm{I}}(N_*)$, the (simplified) condition is entailed by reduced fairness. There is a crucial difference, though: While reduced fairness requires that every inference from N_* is redundant or has a redundant premise at some finite step of the derivation, the Bachmair–Ganzinger definition also admits derivations where redundancy is achieved only in the limit.

Example 17. Consider a signature consisting of two unary predicate symbols P, Q, a unary function symbol f, and a constant b. Let *Inf* be the set of inferences of the ordered resolution calculus with selection over clauses over the signature.

Let N be the set of clauses (1) $\mathsf{P}(\mathsf{b})$, (2) $\neg\mathsf{P}(x) \vee \mathsf{P}(\mathsf{f}(x))$, (3) $\mathsf{Q}(\mathsf{b})$, (4) $\neg\mathsf{Q}(\mathsf{b}) \vee \mathsf{P}(\mathsf{f}(x))$, where the atom ordering is an LPO with precedence $\mathsf{P} > \mathsf{Q} > \mathsf{f} > \mathsf{b}$ and the first literals of (2) and (4) are selected. From (1) and (2), we obtain in the first derivation step $\mathsf{P}(\mathsf{f}(\mathsf{b}))$, in the second step $\mathsf{P}(\mathsf{f}(\mathsf{f}(\mathsf{b}))))$, and so on. The limit N_* consists of the four initial clauses (1)–(4) and all clauses of the form $\mathsf{P}(\mathsf{f}^i(\mathsf{b}))$ with $i \geq 1$. The resolution inference between (3) and (4), yielding $\mathsf{P}(\mathsf{f}(x))$, is therefore redundant w.r.t. N_*, since for each of its ground instances the conclusion $\mathsf{P}(\mathsf{f}^i(\mathsf{b}))$ is contained in N_*. However, it is not redundant w.r.t. any set N_j. Similarly, the premise (4) is redundant w.r.t. N_* but not w.r.t. any set N_j. Therefore, the sequence of clause sets is fair according to the definition in Bachmair and Ganzinger [5, Sect. 4.1], but neither fair nor reducedly fair according to our definitions.

Of course, a redundancy property that holds only for the limit of an infinite sequence can never be checked effectively. In other words, Bachmair and Ganzinger's definition is more permissive than our alternative definition, but the additional degree of freedom cannot be exploited in a theorem prover.

Nonstrict Redundancy. Nieuwenhuis and Rubio [20] and Peltier [22] define a ground clause C to be nonstrictly redundant w.r.t. a set N of ground clauses if C follows from smaller *or equal* clauses in N. This definition does not satisfy our condition (R3). Consequently, it can be used for proving the static completeness of a calculus, but it is insufficient to establish the connection between static and dynamic completeness (unless the notion of fairness is strengthened).

2.4 Intersections of Redundancy Criteria

In the sequel, it will be useful to define consequence relations and redundancy criteria as intersections of previously defined consequence relations or redundancy criteria.

Let Q be a set, and let $(\models^q)_{q \in Q}$ be a Q-indexed family of consequence relations over $\mathbf{F}$. Define $\models^{\cap} := \bigcap_{q \in Q} \models^q$.

lem:intersection-of-conseq-rel

Lemma 18. $\models^{\cap}$ *is a consequence relation.*

Proof. Obvious. □

Let Inf be an inference system, and let $(\mathit{Red}^q)_{q\in Q}$ be a Q-indexed family of redundancy criteria, where each $\mathit{Red}^q = (\mathit{Red}^q_{\mathrm{I}}, \mathit{Red}^q_{\mathrm{F}})$ is a redundancy criterion for Inf and $\models^q$. Let $\mathit{Red}^{\bigcap}_{\mathrm{I}}(N) := \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{I}}(N)$ and $\mathit{Red}^{\bigcap}_{\mathrm{F}}(N) := \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{F}}(N)$. Define $\mathit{Red}^{\cap} = (\mathit{Red}^{\bigcap}_{\mathrm{I}}, \mathit{Red}^{\bigcap}_{\mathrm{F}}) := \bigcap_{q\in Q} \mathit{Red}^q$.

lem:intersection-of-red-crit

Lemma 19. *$\mathit{Red}^{\cap} = \bigcap_{q\in Q} \mathit{Red}^q$ is a redundancy criterion for $\models^{\cap}$ and Inf.*

Proof. (R1) Assume that $N \models^{\cap} \{\bot\}$ for some $\bot \in \mathbf{F}_\bot$—i.e., $N \models^q \{\bot\}$ for every $q \in Q$. As $\mathit{Red}^{\bigcap}_{\mathrm{F}}(N) \subseteq \mathit{Red}^q_{\mathrm{F}}(N)$, we have $N \setminus \mathit{Red}^{\bigcap}_{\mathrm{F}}(N) \supseteq N \setminus \mathit{Red}^q_{\mathrm{F}}(N)$, and by (C2) $N \setminus \mathit{Red}^{\bigcap}_{\mathrm{F}}(N) \models^q N \setminus \mathit{Red}^q_{\mathrm{F}}(N)$. Furthermore, $N \setminus \mathit{Red}^q_{\mathrm{F}}(N) \models^q \{\bot\}$ by (R1) for Red^q. So $N \setminus \mathit{Red}^{\bigcap}_{\mathrm{F}}(N) \models^q \{\bot\}$ by (C4) for every $q \in Q$ and therefore $N \setminus \mathit{Red}_{\mathrm{F}}(N) \models^{\cap} \{\bot\}$.

(R2) Let $N \subseteq N'$. Since $\mathit{Red}^q_{\mathrm{F}}(N) \subseteq \mathit{Red}^q_{\mathrm{F}}(N')$ for every q, we have $\mathit{Red}^{\bigcap}_{\mathrm{F}}(N) = \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{F}}(N) \subseteq \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{F}}(N') = \mathit{Red}^{\bigcap}_{\mathrm{F}}(N')$ and analogously for $\mathit{Red}^{\bigcap}_{\mathrm{I}}$.

(R3) Let $N' \subseteq \mathit{Red}_{\mathrm{F}}(N)$. Since $\mathit{Red}^q_{\mathrm{F}}(N) \subseteq \mathit{Red}^q_{\mathrm{F}}(N \setminus N')$ for every q, we have $\mathit{Red}^{\bigcap}_{\mathrm{F}}(N) = \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{F}}(N) \subseteq \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{F}}(N \setminus N') = \mathit{Red}^{\bigcap}_{\mathrm{F}}(N \setminus N')$ and analogously for $\mathit{Red}^{\bigcap}_{\mathrm{I}}$.

(R4) If $\iota \in \mathit{Inf}$ and $\mathit{concl}(\iota) \in N$, then $\iota \in \mathit{Red}^q_{\mathrm{I}}(N)$ for every $q \in Q$; hence $\iota \in \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{I}}(N) = \mathit{Red}^{\bigcap}_{\mathrm{I}}(N)$. □

lem:satur-wrt-intersection-of-red

Lemma 20. *A set $N \subseteq \mathbf{F}$ is saturated w.r.t. Inf and $\mathit{Red}^{\cap}$ if and only if it is saturated w.r.t. Inf and Red^q for every $q \in Q$.*

Proof. If N is saturated w.r.t. Inf and $\mathit{Red}^{\cap}$, then $\mathit{Inf}(N) \subseteq \mathit{Red}^{\bigcap}_{\mathrm{I}}(N) = \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{I}}(N)$; hence $\mathit{Inf}(N) \subseteq \mathit{Red}^q_{\mathrm{I}}(N)$ for every $q \in Q$, implying that N is saturated w.r.t. Inf and Red^q.

Conversely, if N is saturated w.r.t. Inf and Red^q for every $q \in Q$, then $\mathit{Inf}(N) \subseteq \mathit{Red}^q_{\mathrm{I}}(N)$ for every $q \in Q$; hence $\mathit{Inf}(N) \subseteq \mathit{Red}^{\bigcap}_{\mathrm{I}}(N) = \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{I}}(N)$, which implies that N is saturated w.r.t. Inf and $\mathit{Red}^{\cap}$. □

In many cases where a redundancy criterion $\mathit{Red}^{\cap}$ is defined as the intersection of other criteria, the consequence relations $\models^q$ agree for all $q \in Q$. For calculi where they disagree, such as constraint superposition [19], one can typically demonstrate the static refutational completeness of $(\mathit{Inf}, \mathit{Red}^{\cap})$ in the following form:

lem:checking-static-ref-compl-for-intersections

Lemma 21. *If for every set $N \subseteq \mathbf{F}$ that is saturated w.r.t. Inf and $\mathit{Red}^{\cap}$ and does not contain any $\bot' \in \mathbf{F}_\bot$ there exists some $q \in Q$ such that $N \not\models^q \{\bot\}$ for some $\bot \in \mathbf{F}_\bot$, then $(\mathit{Inf}, \mathit{Red}^{\cap})$ is statically refutationally complete w.r.t. $\models^{\cap}$.*

Proof. Suppose that $N \subseteq \mathbf{F}$ is saturated w.r.t. Inf and $\mathit{Red}^{\cap}$ and $N \models^{\cap} \{\bot''\}$ for some $\bot'' \in \mathbf{F}_\bot$. Consequently, $N \models^q \{\bot''\}$ for every $q \in Q$. By (C1), $N \models^q \{\bot''\} \models^q \{\bot\}$ for every $\bot \in \mathbf{F}_\bot$. If the condition of the lemma holds, then N must contain some $\bot' \in \mathbf{F}_\bot$. Therefore, $(\mathit{Inf}, \mathit{Red}^{\cap})$ is statically refutationally complete w.r.t. $\models^{\cap}$. □

3 Lifting

A standard approach for establishing the refutational completeness of a calculus is to first concentrate on the ground case and then lift the results to the non-ground case. In this section, we show how to perform this lifting abstractly, given a suitable grounding function $\mathcal{G}$. The function maps every formula $C \in \mathbf{F}$ to a set $\mathcal{G}(C)$ of formulas from a class of formulas $\mathbf{G}$. Depending on the logic and the calculus, $\mathcal{G}(C)$ may be, for example, the set of all ground instances of C, a subset of the set of ground instances of C, or even a set of formulas from another logic. Similarly, *FInf*-inferences are mapped to sets of *GInf*-inferences, and saturation w.r.t. *FInf*-inferences is related to saturation w.r.t. *GInf*-inferences. (There are calculi where some *FInf*-inferences ι do not have a counterpart in *GInf*, such as the ArgCong or PosExt inferences of higher-order superposition [10, 11] and the Close inferences of hierarchic superposition [7,9]. In these cases, we set $\mathcal{G}(\iota) = \mathit{undef}$.)

3.1 Standard Lifting

Given two sets of formulas $\mathbf{F}$ and $\mathbf{G}$, an $\mathbf{F}$-inference system *FInf*, a $\mathbf{G}$-inference system *GInf*, and a redundancy criterion *Red* for *GInf*, let $\mathcal{G}$ be a function that maps every formula in $\mathbf{F}$ to a subset of $\mathbf{G}$ and every $\mathbf{F}$-inference in *FInf* to *undef* or to a subset of *GInf*. The function $\mathcal{G}$ is called a *grounding function* if

(G1) for every $\bot \in \mathbf{F}_\bot$, $\emptyset \neq \mathcal{G}(\bot) \subseteq \mathbf{G}_\bot$;
(G2) for every $C \in \mathbf{F}$, if $\bot \in \mathcal{G}(C)$ and $\bot \in \mathbf{G}_\bot$ then $C \in \mathbf{F}_\bot$;
(G3) for every $\iota \in \mathit{FInf}$, if $\mathcal{G}(\iota) \neq \mathit{undef}$, then $\mathcal{G}(\iota) \subseteq \mathit{Red}_\mathrm{I}(\mathcal{G}(\mathit{concl}(\iota)))$.

The function $\mathcal{G}$ is extended to sets $N \subseteq \mathbf{F}$ by defining $\mathcal{G}(N) := \bigcup_{C \in N} \mathcal{G}(C)$. Analogously, for a set $X \subseteq \mathit{FInf}$, $\mathcal{G}(X) := \bigcup_{\iota \in X,\, \mathcal{G}(\iota) \neq \mathit{undef}} \mathcal{G}(\iota)$.

Remark 22. Conditions (G1) and (G2) express that *false* formulas may only be mapped to sets of *false* formulas, and that only *false* formulas may be mapped to sets of *false* formulas. For most applications, it would be possible to replace condition (G3) by

(G3′) for every $\iota \in \mathit{FInf}$, if $\mathcal{G}(\iota) \neq \mathit{undef}$ then $\mathit{concl}(\mathcal{G}(\iota)) \subseteq \mathcal{G}(\mathit{concl}(\iota))$,

which implies (G3) by property (R4). There are some calculi, however, for which condition (G3′) is too strong. Typical examples are calculi where the $\mathbf{F}$-inferences include some normalization or abstraction step that does not have a counterpart in the $\mathbf{G}$-inferences. So an $\mathbf{F}$-inference ι may have a conclusion $C \vee t \not\approx t'$, where the literal $t \not\approx t'$ results from the normalization step, but the conclusions of the instances of ι have the form $C\theta$ for a substitution θ that unifies t and t'. In this case, (G3) is still satisfied, but (G3′) is not.

Example 23. In standard superposition, $\mathbf{F}$ is the set of all universally quantified first-order clauses over some signature Σ, $\mathbf{G}$ is the set of all ground first-order clauses over Σ, and $\mathcal{G}$ maps every clause C to the set of its ground instances $C\theta$ and every superposition inference ι to the set of its ground instances $\iota\theta$.

Let $\mathcal{G}$ be a grounding function from $\mathbf{F}$ and *FInf* to $\mathbf{G}$ and *GInf*, and let $\models \subseteq \mathcal{P}(\mathbf{G}) \times \mathcal{P}(\mathbf{G})$ be a consequence relation over $\mathbf{G}$. We define the relation $\models_{\mathcal{G}} \subseteq \mathcal{P}(\mathbf{F}) \times \mathcal{P}(\mathbf{F})$ such that $N_1 \models_{\mathcal{G}} N_2$ if and only if $\mathcal{G}(N_1) \models \mathcal{G}(N_2)$. We call $\models_{\mathcal{G}}$ the *$\mathcal{G}$-lifting* of $\models$.

lem:derived-consequence-relation

Lemma 24. *$\models_{\mathcal{G}}$ is a consequence relation over $\mathbf{F}$.*

Proof. (C1) Let $\bot \in \mathbf{F}_{\bot}$. Then $\mathcal{G}(\{\bot\})$ contains some $\bot' \in \mathbf{G}_{\bot}$. So $\mathcal{G}(\{\bot\}) \models \{\bot'\} \models \mathcal{G}(N_1)$ for every N_1, and hence $\{\bot\} \models_{\mathcal{G}} N_1$ as required.

(C2) Let $N_2 \subseteq N_1$, then $\mathcal{G}(N_2) \subseteq \mathcal{G}(N_1)$, so $\mathcal{G}(N_1) \models \mathcal{G}(N_2)$, and thus $N_1 \models_{\mathcal{G}} N_2$.

(C3) Suppose that $N_1 \models_{\mathcal{G}} \{C\}$ for every $C \in N_2$. Then $\mathcal{G}(N_1) \models \mathcal{G}(\{C\})$ for every $C \in N_2$ and therefore $\mathcal{G}(N_1) \models \bigcup_{C \in N_2} \mathcal{G}(\{C\}) = \mathcal{G}(N_2)$; hence $N_1 \models_{\mathcal{G}} N_2$.

(C4) Suppose that $N_1 \models_{\mathcal{G}} N_2$ and $N_2 \models_{\mathcal{G}} N_3$. Then $\mathcal{G}(N_1) \models \mathcal{G}(N_2)$ and $\mathcal{G}(N_2) \models \mathcal{G}(N_3)$; therefore $\mathcal{G}(N_1) \models \mathcal{G}(N_3)$, and therefore $N_1 \models_{\mathcal{G}} N_3$. □

Let $Red = (Red_{\mathrm{I}}, Red_{\mathrm{F}})$ be a redundancy criterion for *GInf* and $\models$. We define functions $Red_{\mathrm{I}}^{\mathcal{G}} : \mathcal{P}(\mathbf{F}) \to \mathcal{P}(FInf)$ and $Red_{\mathrm{F}}^{\mathcal{G}} : \mathcal{P}(\mathbf{F}) \to \mathcal{P}(\mathbf{F})$ by

$$\begin{aligned}
&\iota \in Red_{\mathrm{I}}^{\mathcal{G}}(N) \text{ if and only if} \\
&\quad \mathcal{G}(\iota) \neq \mathit{undef} \text{ and } \mathcal{G}(\iota) \subseteq Red_{\mathrm{I}}(\mathcal{G}(N)) \\
&\quad \text{or } \mathcal{G}(\iota) = \mathit{undef} \text{ and } \mathcal{G}(\mathit{concl}(\iota)) \subseteq \mathcal{G}(N) \cup Red_{\mathrm{F}}(\mathcal{G}(N)); \\
&C \in Red_{\mathrm{F}}^{\mathcal{G}}(N) \text{ if and only if} \\
&\quad \mathcal{G}(C) \subseteq Red_{\mathrm{F}}(\mathcal{G}(N)).
\end{aligned}$$

We call $Red^{\mathcal{G}} = (Red_{\mathrm{I}}^{\mathcal{G}}, Red_{\mathrm{F}}^{\mathcal{G}})$ the *$\mathcal{G}$-lifting* of *Red*.

thm:lifted-red-crit

Theorem 25. *$Red^{\mathcal{G}} = (Red_{\mathrm{I}}^{\mathcal{G}}, Red_{\mathrm{F}}^{\mathcal{G}})$ is a redundancy criterion for FInf and $\models_{\mathcal{G}}$.*

We omit the proof at this point since we will prove a more general result (Theorem 37) in Sect. 3.2.

We get the following folklore theorem.

lem:sat-wrt-finf

Lemma 26. *If $N \subseteq \mathbf{F}$ is saturated w.r.t. FInf and $Red^{\mathcal{G}}$ and $GInf(\mathcal{G}(N)) \subseteq \mathcal{G}(FInf(N)) \cup Red_{\mathrm{I}}(\mathcal{G}(N))$, then $\mathcal{G}(N)$ is saturated w.r.t. GInf and Red.*

Proof. Suppose that N is saturated w.r.t. *FInf* and $Red^{\mathcal{G}}$—i.e., $FInf(N) \subseteq Red_{\mathrm{I}}^{\mathcal{G}}(N)$. We must show that $\mathcal{G}(N)$ is saturated w.r.t. *GInf* and *Red*—i.e., $GInf(\mathcal{G}(N)) \subseteq Red_{\mathrm{I}}(\mathcal{G}(N))$.

Let $\iota \in GInf(\mathcal{G}(N))$. By assumption, ι is contained in $\mathcal{G}(FInf(N))$ or in $Red_{\mathrm{I}}(\mathcal{G}(N))$. In the second case, we are done immediately. In the first case, $\iota \in \mathcal{G}(\iota')$ for some $\iota' \in FInf(N) \subseteq Red_{\mathrm{I}}^{\mathcal{G}}(N)$ with $\mathcal{G}(\iota) \neq \mathit{undef}$, so by definition of $Red_{\mathrm{I}}^{\mathcal{G}}$ we have again $\iota \in Red_{\mathrm{I}}(\mathcal{G}(N))$. □

An inference in $GInf(\mathcal{G}(N))$ is called *liftable* if it contained in $\mathcal{G}(FInf(N))$. Using this terminology, we can rephrase the lemma as follows: If N is saturated and every unliftable inference from $\mathcal{G}(N)$ is redundant w.r.t. $\mathcal{G}(N)$, then $\mathcal{G}(N)$ is saturated.

thm:finf-complete

Theorem 27. *If* $(\mathit{GInf}, \mathit{Red})$ *is statically refutationally complete w.r.t.* $\models$*, and if we have* $\mathit{GInf}(\mathcal{G}(N)) \subseteq \mathcal{G}(\mathit{FInf}(N)) \cup \mathit{Red}_{\mathrm{I}}(\mathcal{G}(N))$ *for every* $N \subseteq \mathbf{F}$ *that is saturated w.r.t.* FInf *and* $\mathit{Red}^{\mathcal{G}}$*, then* $(\mathit{FInf}, \mathit{Red}^{\mathcal{G}})$ *is statically refutationally complete w.r.t.* $\models_{\mathcal{G}}$*.*

Proof. Assume $(\mathit{GInf}, \mathit{Red})$ is statically refutationally complete w.r.t. $\models$. Assume $N \subseteq \mathbf{F}$ is saturated w.r.t. FInf and $\mathit{Red}^{\mathcal{G}}$ and assume that $N \models_{\mathcal{G}} \bot$ for some $\bot \in \mathbf{F}_{\bot}$. We must show that $\bot' \in N$ for some $\bot' \in \mathbf{F}_{\bot}$.

By definition of $\models_{\mathcal{G}}$, we know that $\mathcal{G}(N) \models \mathcal{G}(\bot)$. Since $\mathcal{G}$ is a grounding function, $\mathcal{G}(\bot)$ is a nonempty subset of $\mathbf{G}_{\bot}$. Let $\bot_{\mathbf{G}} \in \mathcal{G}(\bot)$, then $\mathcal{G}(N) \models \mathcal{G}(\bot) \models \{\bot_{\mathbf{G}}\}$.

By the previous lemma, we know that $\mathcal{G}(N)$ is saturated w.r.t. GInf and Red, so there exists some $\bot'_{\mathbf{G}} \in \mathbf{G}_{\bot}$ such that $\bot'_{\mathbf{G}} \in \mathcal{G}(N)$. Hence $\bot'_{\mathbf{G}} \in \mathcal{G}(C)$ for some $C \in N$, which implies $C \in \mathbf{F}_{\bot}$. Now define $\bot' := C$. □

Example 28. In ordered resolution with selection, all inferences are liftable, as demonstrated below. Let Σ be a first-order signature containing at least one constant, let $\mathbf{F}$ be the set of all Σ-clauses without equality, and let $\mathbf{G}$ be the set of all ground Σ-clauses without equality. Let FInf and GInf be the sets of all resolution or factoring inferences from clauses in respectively $\mathbf{F}$ and $\mathbf{G}$ that satisfy the given ordering and selection restrictions, and let $\mathcal{G}$ be the function that maps every clause $C \in \mathbf{F}$ to the set of all its ground instances $C\theta$ and that maps every inference $(C_n, \ldots, C_0)$ in FInf to the set of all $(C_n\theta, \ldots, C_0\theta)$ in GInf. As usual, we assume that every clause D in $\mathbf{G}$ inherits the selection of at least one clause $C \in \mathbf{F}$ such that $D \in \mathcal{G}(C)$. Then every resolution inference in GInf from ground instances of clauses in N has the form

$$\frac{D'\theta \vee B\theta \quad C'\theta \vee \neg A\theta}{D'\theta \vee C'\theta}$$

with $A\theta = B\theta$ and is contained in $\mathcal{G}(\iota)$ for some inference ι in $\mathit{FInf}(N)$ of the form

$$\frac{D' \vee B \quad C' \vee \neg A}{(D' \vee C')\sigma}$$

with $\sigma = \mathrm{mgu}(A, B)$, and analogously for factoring inferences.

Thus, the static refutational completeness of GInf implies the static refutational completeness of FInf.

Example 29. In the superposition calculus, all inferences are liftable, except superpositions at or below a variable position. Let Σ be a first-order signature containing at least one constant and no predicate symbols except $\approx$, let $\mathbf{F}$ be the set of all Σ-clauses with equality, and let $\mathbf{G}$ be the set of all ground Σ-clauses with equality. Let FInf and GInf be the sets of all superposition, equality resolution, and equality factoring inferences from clauses in respectively $\mathbf{F}$ and $\mathbf{G}$ that satisfy the given ordering and selection restrictions, and let $\mathcal{G}$ be the function

that maps every clause $C \in \mathbf{F}$ to the set of all its ground instances $C\theta$ and that maps every inference $(C_n, \ldots, C_0)$ in *FInf* to the set of all $(C_n\theta, \ldots, C_0\theta)$ in *GInf*. As usual, we assume that every clause D in $\mathbf{G}$ inherits the selection of at least one clause $C \in \mathbf{F}$ such that $D \in \mathcal{G}(C)$. Then every equality resolution or equality factoring inference from ground instances of clauses in N is contained in $\mathcal{G}(\iota)$ for some inference ι in $\mathit{FInf}(N)$. The same applies to superposition inferences

$$\frac{D'\theta \vee t\theta \approx t'\theta \quad C'\theta \vee [\neg]s\theta \approx s'\theta}{D'\theta \vee C'\theta \vee [\neg]s\theta[t'\theta]_p \approx s'\theta}$$

with $s\theta|_p = t\theta$, provided that p is a position of s and $s|_p$ is not a variable. Otherwise, $p = p_1p_2$ for some variable x occurring in s at the position p_1, so $x\theta|_{p_2} = t\theta$. In this case, define θ' by $x\theta' = x\theta[t'\theta]_{p_2}$ and $y\theta' = y\theta$ for $y \neq x$. By congruence, the conclusion of the inference is entailed by the first premise (which is necessarily smaller than the second) and $C'\theta' \vee [\neg]s\theta' \approx s'\theta'$. The ordering restrictions of the calculus require that $t\theta \succ t'\theta$; hence the latter clause is also smaller than the second premise. By the usual redundancy criterion for superposition, this renders the inference redundant w.r.t. N.

As for ordered resolution, the static refutational completeness of *GInf* implies the static refutational completeness of *FInf*.

3.2 Adding Well-Founded Orderings

We now strengthen the $\mathcal{G}$-lifting of redundancy criteria introduced in the previous subsection so that subsumption deletion is also supported. Let $\sqsupset = (\sqsupset_D)_{D \in \mathbf{G}}$ be a $\mathbf{G}$-indexed family of well-founded strict partial orderings on $\mathbf{F}$. We define functions $\mathit{Red}_{\mathrm{I}}^{\mathcal{G}} : \mathcal{P}(\mathbf{F}) \to \mathcal{P}(\mathit{FInf})$ and $\mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset} : \mathcal{P}(\mathbf{F}) \to \mathcal{P}(\mathbf{F})$ as follows:

$\iota \in \mathit{Red}_{\mathrm{I}}^{\mathcal{G}}(N)$ if and only if
  $\mathcal{G}(\iota) \neq \mathit{undef}$ and $\mathcal{G}(\iota) \subseteq \mathit{Red}_{\mathrm{I}}(\mathcal{G}(N))$
  or $\mathcal{G}(\iota) = \mathit{undef}$ and $\mathcal{G}(\mathit{concl}(\iota)) \subseteq \mathcal{G}(N) \cup \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$;

$C \in \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$ if and only if
  for every $D \in \mathcal{G}(C)$,
    $D \in \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$ or there exists $C' \in N$ such that $C \sqsupset_D C'$ and $D \in \mathcal{G}(C')$.

We call $\mathit{Red}^{\mathcal{G},\sqsupset} = (\mathit{Red}_{\mathrm{I}}^{\mathcal{G}}, \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset})$ the $(\mathcal{G}, \sqsupset)$-*lifting* of *Red*.

For nearly all applications, the orderings $\sqsupset_D$ agree for all $D \in \mathbf{G}$. In these cases, we may take $\sqsupset$ as a single well-founded strict partial ordering, rather than as a $\mathbf{G}$-indexed family of such orderings. Note that we get the previously defined $\mathit{Red}^{\mathcal{G}} = (\mathit{Red}_{\mathrm{I}}^{\mathcal{G}}, \mathit{Red}_{\mathrm{F}}^{\mathcal{G}})$ as a special case of $\mathit{Red}^{\mathcal{G},\sqsupset} = (\mathit{Red}_{\mathrm{I}}^{\mathcal{G}}, \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset})$ by setting $\sqsupset_D := \emptyset$—i.e., the empty strict partial ordering on $\mathbf{F}$—for every $D \in \mathbf{G}$.

As demonstrated by the following lemma, we may assume without loss of generality that the formula C' in the definition of $\mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}$ is contained in $N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$:

lem:wolog-C'-nonredundant

Lemma 30. *$C \in \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$ if and only if for every $D \in \mathcal{G}(C)$ we have $D \in \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$ or there exists $C' \in N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$ such that $C \sqsupset_D C'$ and $D \in \mathcal{G}(C')$.*

Proof. The "if" direction is trivial. For the "only if" direction, assume that $C \in \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$ and $D \in \mathcal{G}(C)$. By definition, $D \in \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$ or there exists $C' \in N$ such that $C \sqsupset_D C'$ and $D \in \mathcal{G}(C')$. If $D \in \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$, we are done. Let $D \notin \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$. By well-foundedness of $\sqsupset_D$, there exists a minimal formula $C' \in N$ w.r.t. $\sqsupset_D$ such that $C \sqsupset_D C'$ and $D \in \mathcal{G}(C')$. Assume that C' were contained in $\mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$. Since $D \notin \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$, there exists $C'' \in N$ such that $C' \sqsupset_D C''$ and $D \in \mathcal{G}(C'')$. But then $C \sqsupset_D C''$, contradicting the minimality of C'. So $C' \in N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$. □

Next, we show that $(\mathit{Red}_{\mathrm{I}}^{\mathcal{G}}, \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset})$ is a redundancy criterion. We start with a technical lemma:

lem:lifting-main-technical

Lemma 31. *$\mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N)) \subseteq \mathcal{G}(N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N))$.*

Proof. Let $D \in \mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$. Since $D \in \mathcal{G}(N)$, there exists $C \in N$ with $D \in \mathcal{G}(C)$. Let C be a minimal formula with this property w.r.t. $\sqsupset_D$.

Assume that $C \in \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$. Then, by definition, $D \in \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$ or there exists $C' \in N$ such that $C \sqsupset_D C'$ and $D \in \mathcal{G}(C')$. The first property contradicts our initial assumption, whereas the second property contradicts the minimality of C. So $C \notin \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$ and thus $D \in \mathcal{G}(N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N))$. □

We can now show that $(\mathit{Red}_{\mathrm{I}}^{\mathcal{G}}, \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset})$ satisfies the properties (R1)–(R4) of redundancy criteria:

lem:nonredundant-entails-redundant

Lemma 32. *If $N \models_{\mathcal{G}} \{\bot\}$ for some $\bot \in \mathbf{F}_{\bot}$, then $N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N) \models_{\mathcal{G}} \{\bot\}$.*

Proof. Let $\bot \in \mathbf{F}_{\bot}$ and suppose that $N \models_{\mathcal{G}} \{\bot\}$—i.e., $\mathcal{G}(N) \models \mathcal{G}(\{\bot\})$. Since $\mathcal{G}(\{\bot\})$ contains some $\bot' \in \mathbf{G}_{\bot}$, $\mathcal{G}(N) \models \mathcal{G}(\{\bot\}) \models \{\bot'\}$. By property (R1) of redundancy criteria, this implies $\mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N)) \models \{\bot'\}$. Furthermore, by Lemma 31, $\mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N)) \subseteq \mathcal{G}(N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N))$, and therefore $\mathcal{G}(N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)) \models \mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$. Combining both relations, we obtain $\mathcal{G}(N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)) \models \{\bot'\} \models \mathcal{G}(\{\bot\})$. By definition of $\models_{\mathcal{G}}$, this means $N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N) \models_{\mathcal{G}} \{\bot\}$, as required. □

lem:redundancy-monotonic-addition

Lemma 33. *If $N \subseteq N'$, then $\mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N) \subseteq \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N')$ and $\mathit{Red}_{\mathrm{I}}^{\mathcal{G}}(N) \subseteq \mathit{Red}_{\mathrm{I}}^{\mathcal{G}}(N')$*

Proof. Obvious. □

lem:redundancy-monotonic-deletion-forms

Lemma 34. *If $N' \subseteq \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$, then $\mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N) \subseteq \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N \setminus N')$.*

Proof. Let $N' \subseteq \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$, let $C \in \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$. Then for every $D \in \mathcal{G}(C)$ we have $D \in \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$ or there exists $C' \in N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$ such that $C \sqsupset_D C'$ and $D \in \mathcal{G}(C')$.

Case 1: $D \in \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$. By property (R3), $D \in \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N)))$. Since $\mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N)) \subseteq \mathcal{G}(N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)) \subseteq \mathcal{G}(N \setminus N')$, this implies $D \in \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N \setminus N'))$.

Case 2: $D \notin \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$ and there exists $C' \in N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$ such that $C \sqsupset_D C'$ and $D \in \mathcal{G}(C')$. Since $N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N) \subseteq N \setminus N'$, we get $C' \in N \setminus N'$.

Since every $D \in \mathcal{G}(C)$ is either contained in $\mathit{Red}_{\mathrm{F}}(\mathcal{G}(N \setminus N'))$ or in $\mathcal{G}(C')$ for some $C' \in N \setminus N'$ with $C \sqsupset_D C'$, we conclude that $C \in \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N \setminus N')$. □

lem:redundancy-monotonic-deletion-infs

Lemma 35. *If $N' \subseteq \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$, then $\mathit{Red}_{\mathrm{I}}^{\mathcal{G}}(N) \subseteq \mathit{Red}_{\mathrm{I}}^{\mathcal{G}}(N \setminus N')$.*

Proof. Let $N' \subseteq \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$, let $\iota \in \mathit{Red}_{\mathrm{I}}^{\mathcal{G}}(N)$.

If $\mathcal{G}(\iota) \neq \mathit{undef}$, then every $\iota' \in \mathcal{G}(\iota)$ is contained in $\mathit{Red}_{\mathrm{I}}(\mathcal{G}(N))$, and by property (R3) also in $\mathit{Red}_{\mathrm{I}}(\mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N)))$. Furthermore, since $\mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N)) \subseteq \mathcal{G}(N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)) \subseteq \mathcal{G}(N \setminus N')$, this implies $\iota' \in \mathit{Red}_{\mathrm{I}}(\mathcal{G}(N \setminus N'))$.

Since every $\iota' \in \mathcal{G}(\iota)$ is contained in $\mathit{Red}_{\mathrm{I}}(\mathcal{G}(N \setminus N'))$, we conclude that $\iota \in \mathit{Red}_{\mathrm{I}}^{\mathcal{G}}(N \setminus N')$.

Otherwise $\mathcal{G}(\iota) = \mathit{undef}$. Then $\mathcal{G}(\mathit{concl}(\iota)) \subseteq \mathcal{G}(N) \cup \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N)) = (\mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))) \cup \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$. Let $D \in \mathcal{G}(\mathit{concl}(\iota))$. We consider two cases: If $D \in \mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$, then by Lemma 31, $D \in \mathcal{G}(N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)) \subseteq \mathcal{G}(N \setminus N')$. Otherwise $D \in \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N))$, then by (R3) $D \in \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N)))$. Since $\mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N)) \subseteq \mathcal{G}(N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)) \subseteq \mathcal{G}(N \setminus N')$, this implies $D \in \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N \setminus N'))$. Combining both cases, we obtain $\mathcal{G}(\mathit{concl}(\iota)) \in \mathcal{G}(N \setminus N') \cup \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N \setminus N'))$, hence $\iota \in \mathit{Red}_{\mathrm{I}}^{\mathcal{G}}(N \setminus N')$. □

lem:concl-contained-implies-red-inf

Lemma 36. *If $\iota \in \mathit{FInf}$ and $\mathit{concl}(\iota) \in N$, then $\iota \in \mathit{Red}_{\mathrm{I}}^{\mathcal{G}}(N)$.*

Proof. Let $\iota \in \mathit{FInf}$ such that $\mathit{concl}(\iota) \in N$. If $\mathcal{G}(\iota) \neq \mathit{undef}$, then $\mathcal{G}(\iota)$ is a subset of $\mathit{Red}_{\mathrm{I}}(\mathcal{G}(\mathit{concl}(\iota)))$, which in turn is a subset of $\mathit{Red}_{\mathrm{I}}(\mathcal{G}(N))$. So $\iota \in \mathit{Red}_{\mathrm{I}}^{\mathcal{G}}(N)$.

Otherwise, $\mathcal{G}(\iota) = \mathit{undef}$. Then $\mathit{concl}(\iota) \in N$ implies $\mathcal{G}(\mathit{concl}(\iota)) \in \mathcal{G}(N)$, so again $\iota \in \mathit{Red}_{\mathrm{I}}^{\mathcal{G}}(N)$. □

By combining Lemmas 32–36, we obtain our first main result, generalizing Theorem 25:

thm:FRedsqsubset-is-red-crit

Theorem 37. *Let $\mathit{Red} = (\mathit{Red}_{\mathrm{I}}, \mathit{Red}_{\mathrm{F}})$ be a redundancy criterion for GInf and $\models$, let $\mathcal{G}$ be a grounding function from $\mathbf{F}$ and FInf to $\mathbf{G}$ and GInf, and let $\sqsupset = (\sqsupset_D)_{D \in \mathbf{G}}$ be a $\mathbf{G}$-indexed family of well-founded strict partial orderings on $\mathbf{F}$. Then the $(\mathcal{G}, \sqsupset)$-lifting $(\mathit{Red}_{\mathrm{I}}^{\mathcal{G}}, \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset})$ of Red is a redundancy criterion for FInf and $\models_{\mathcal{G}}$.*

Note that the first component of $\mathit{Red}^{\mathcal{G},\sqsupset} = (\mathit{Red}_{\mathrm{I}}^{\mathcal{G}}, \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset})$ does not depend on $\sqsupset$ and that the definitions of a saturated set and of static refutational completeness do not depend on the second component of a redundancy criterion. The following lemmas are immediate consequences of these observations:

lem:saturation-indep-of-sqsubset

Lemma 38. *A set $N \subseteq \mathbf{F}$ is saturated w.r.t. FInf and $(\mathit{Red}_{\mathrm{I}}^{\mathcal{G}}, \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset})$ if and only if it is saturated w.r.t. FInf and $(\mathit{Red}_{\mathrm{I}}^{\mathcal{G}}, \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\emptyset})$.*

lem:static-ref-compl-indep-of-sqsubset

Lemma 39. *$(FInf, Red^{\mathcal{G},\sqsupset})$ is statically refutationally complete w.r.t. $\models_{\mathcal{G}}$ if and only if $(FInf, Red^{\mathcal{G},\emptyset})$ is statically refutationally complete w.r.t. $\models_{\mathcal{G}}$.*

Combining Lemmas 6 and 39, we obtain our second main result:

thm:FRedsqsubset-is-dyn-ref-compl

Theorem 40. *Let $Red = (Red_{\mathrm{I}}, Red_{\mathrm{F}})$ be a redundancy criterion for GInf and $\models$, let $\mathcal{G}$ be a grounding function from $\mathbf{F}$ and FInf to $\mathbf{G}$ and GInf, and let $\sqsupset = (\sqsupset_D)_{D \in \mathbf{G}}$ be a $\mathbf{G}$-indexed family of well-founded strict partial orderings on $\mathbf{F}$. If $(FInf, Red^{\mathcal{G},\emptyset})$ is statically refutationally complete w.r.t. $\models_{\mathcal{G}}$, then $(FInf, Red^{\mathcal{G},\sqsupset})$ is dynamically refutationally complete w.r.t. $\models_{\mathcal{G}}$.*

Example 41. For resolution or superposition in standard first-order logic, we can define the subsumption quasi-ordering $\succeq$ on clauses by $C \succeq C'$ if and only if $C = C'\sigma$ for some substitution σ. The subsumption ordering $\succ := \succeq \setminus \preceq$ is well founded. By choosing $\sqsupset := \succ$, we obtain a criterion $Red^{\mathcal{G},\sqsupset}$ that includes standard redundancy and also supports subsumption deletion.[3]

Similarly, for proof calculi modulo commutativity (C) or associativity and commutativity (AC), we can let $C \succeq C'$ if there exists a substitution σ such that C equals $C'\sigma$ up to the equational theory (C or AC). The relation $\succ = \succeq \setminus \preceq$ is then again well founded.

Example 42. Constraint superposition with ordering constraints [19] is an example of a calculus where the subsumption ordering is not well founded. A ground instance of a constrained clause $C \,[\![K]\!]$ is a ground clause $C\theta$ for which the constraint $K\theta$ evaluates to true. We can then define $\succeq$ on constrained clauses by stating that $C \,[\![K]\!] \succeq C' \,[\![K']\!]$ if and only if every ground instance of $C \,[\![K]\!]$ is a ground instance of $C' \,[\![K']\!]$. The subsumption ordering $\succ$ defined by $\succ := \succeq \setminus \preceq$ is not well founded, though, since

$$\mathsf{P}(x) \,[\![\, x \prec \mathsf{b} \,]\!] \succ \mathsf{P}(x) \,[\![\, x \prec \mathsf{f}(\mathsf{b}) \,]\!] \succ \mathsf{P}(x) \,[\![\, x \prec \mathsf{f}(\mathsf{f}(\mathsf{b})) \,]\!] \succ \cdots$$

if $\succ$ is a simplification ordering.

Example 43. For higher-order calculi such as higher-order resolution [16] and clausal higher-order superposition [10], subsumption is also not well founded, as witnessed by the chain

$$\mathsf{p}\, x\, x \;\succ\; \mathsf{p}\,(x\,\mathsf{a})\,(x\,\mathsf{b}_1) \;\succ\; \mathsf{p}\,(x\,\mathsf{a}\,\mathsf{a})\,(x\,\mathsf{b}_1\,\mathsf{b}_2) \;\succ\; \cdots .$$

Even if the subsumption ordering for some logic is not well founded, as in the two examples above, we can always define $\sqsupset$ as the intersection of the subsumption ordering and an appropriate ordering based on formula sizes or weights.

[3] It is common to define $\succeq$ by $C \succeq C'$ if $C = C'\sigma \vee C''$ for some substitution σ and some possibly empty clause C'', but since the cases where C'' is nonempty are already handled by the standard redundancy criterion, the easier definition is sufficient.

Example 44. There are a few applications, notably for superposition-based decision procedures [6], where one would like to define $Red_{\mathrm{F}}^{\mathcal{G},\sqsupset}$ using the reverse subsumption ordering $\preccurlyeq$. This ordering is not well founded on the set of all first-order clauses: $\mathsf{P}(x) \preccurlyeq \mathsf{P}(\mathsf{f}(x)) \preccurlyeq \mathsf{P}(\mathsf{f}(\mathsf{f}(x)) \preccurlyeq \cdots$. However, it is well founded if we restrict it to the set of generalizations $gen(D) := \{C \mid D = C\theta \text{ for some } \theta\}$ of a fixed ground clause D, so that we may in fact define $\sqsupset = (\sqsupset_D)_D$ where $\sqsupset_D := \preccurlyeq \cap (gen(D) \times gen(D))$.

3.3 Intersections of Liftings

The results of the previous subsection can be extended in a straightforward way to intersections of lifted redundancy criteria. As before, let $\mathbf{F}$ and $\mathbf{G}$ be two sets of formulas, and let *FInf* be an $\mathbf{F}$-inference system. In addition, let Q be a set. For every $q \in Q$, let $\models^q$ be a consequence relation over $\mathbf{G}$, let $GInf^q$ be a $\mathbf{G}$-inference system, let $Red^q = (Red_{\mathrm{I}}^q, Red_{\mathrm{F}}^q)$ be a redundancy criterion for $GInf^q$ and $\models^q$, and let $\mathcal{G}^q$ be a grounding function from $\mathbf{F}$ and *FInf* to $\mathbf{G}$ and $GInf^q$. Let $\sqsupset = (\sqsupset_D)_{D \in \mathbf{G}}$ be a $\mathbf{G}$-indexed family of well-founded strict partial orderings on $\mathbf{F}$.[4]

For each $q \in Q$, we know by Theorem 37 that the $(\mathcal{G}^q, \emptyset)$-lifting $Red^{q,\mathcal{G}^q,\emptyset} = (Red_{\mathrm{I}}^{q,\mathcal{G}^q}, Red_{\mathrm{F}}^{q,\mathcal{G}^q,\emptyset})$ and the $(\mathcal{G}^q, \sqsupset)$-lifting $Red^{q,\mathcal{G}^q,\sqsupset} = (Red_{\mathrm{I}}^{q,\mathcal{G}^q}, Red_{\mathrm{F}}^{q,\mathcal{G}^q,\sqsupset})$ of Red^q are redundancy criteria for *FInf* and $\models_{\mathcal{G}^q}^q$. Consequently, by Lemma 19 the intersections

$$Red^{\cap} := \bigcap_{q \in Q} Red^{q,\mathcal{G}^q,\emptyset} = \Big(\bigcap_{q \in Q} Red_{\mathrm{I}}^{q,\mathcal{G}^q}, \bigcap_{q \in Q} Red_{\mathrm{F}}^{q,\mathcal{G}^q,\emptyset} \Big)$$

and

$$Red^{\cap,\sqsupset} := \bigcap_{q \in Q} Red^{q,\mathcal{G}^q,\sqsupset} = \Big(\bigcap_{q \in Q} Red_{\mathrm{I}}^{q,\mathcal{G}^q}, \bigcap_{q \in Q} Red_{\mathrm{F}}^{q,\mathcal{G}^q,\sqsupset} \Big)$$

are redundancy criteria for $\models^{\cap} := \bigcap_{q \in Q} \models_{\mathcal{G}^q}^q$.

We get the following analogue of Theorem 27:

thm:intersect-finf-complete

Theorem 45. *If $(GInf^q, Red^q)$ is statically refutationally complete w.r.t. $\models^q$ for every $q \in Q$, and if for every $N \subseteq \mathbf{F}$ that is saturated w.r.t. FInf and $Red^{\cap}$ there exists a q such that $GInf^q(\mathcal{G}^q(N)) \subseteq \mathcal{G}^q(FInf(N)) \cup Red_{\mathrm{I}}^q(\mathcal{G}^q(N))$, then $(FInf, Red^{\cap})$ is statically refutationally complete w.r.t. $\models^{\cap}$.*

Proof. Assume that $(GInf^q, Red^q)$ is statically refutationally complete w.r.t. $\models^q$ for every $q \in Q$ and that for every $N \subseteq \mathbf{F}$ that is saturated w.r.t. *FInf* and $Red^{\cap}$ there exists a q such that $GInf^q(\mathcal{G}^q(N)) \subseteq \mathcal{G}^q(FInf(N)) \cup Red_{\mathrm{I}}^q(\mathcal{G}^q(N))$.

Let $N \subseteq \mathbf{F}$ be saturated w.r.t. *FInf* and $Red^{\cap}$ and assume that $N \models^{\cap} \{\bot\}$ for some $\bot \in \mathbf{F}_{\bot}$. We must show that $\bot' \in N$ for some $\bot' \in \mathbf{F}_{\bot}$. First, we know

[4] We could also use a Q-indexed family of sets $(\mathbf{G}^q)_{q \in Q}$ instead of a single set $\mathbf{G}$, and a $(Q, \mathbf{G}^q)$-indexed family of well-founded strict partial orderings on $\mathbf{F}$ instead of a $\mathbf{G}$-indexed family, but we are not aware of applications where this is necessary.

that there exists a q such that $\mathit{GInf}^q(\mathcal{G}^q(N)) \subseteq \mathcal{G}^q(\mathit{FInf}(N)) \cup \mathit{Red}^q_{\mathrm{I}}(\mathcal{G}^q(N))$. Since $\mathit{Red}^{\cap} = \bigcap_{q \in Q} \mathit{Red}^{q,\mathcal{G}^q,\emptyset}$, we know by Lemma 20 that N is saturated w.r.t. FInf and the $(\mathcal{G}^q, \emptyset)$-lifting $\mathit{Red}^{q,\mathcal{G}^q,\emptyset}$ of Red^q. Therefore, by Lemma 26, $\mathcal{G}^q(N)$ is saturated w.r.t. GInf and Red^q.

Furthermore, $N \models^{\cap} \{\bot\}$ implies $N \models^q_{\mathcal{G}^q} \{\bot\}$, and since $\models^q_{\mathcal{G}^q}$ is the $\mathcal{G}^q$-lifting of $\models^q$, this is equivalent to $\mathcal{G}^q(N) \models^q \mathcal{G}^q(\bot)$. Since $\mathcal{G}^q$ is a grounding function, $\mathcal{G}^q(\bot)$ is a nonempty subset of $\mathbf{G}_\bot$. Let $\bot_{\mathbf{G}} \in \mathcal{G}^q(\bot)$, then $\mathcal{G}^q(N) \models \mathcal{G}^q(\bot) \models \{\bot_{\mathbf{G}}\}$.

Since $\mathcal{G}^q(N)$ is saturated w.r.t. GInf and Red^q, there must exist some $\bot'_{\mathbf{G}} \in \mathbf{G}_\bot$ such that $\bot'_{\mathbf{G}} \in \mathcal{G}^q(N)$. Hence $\bot'_{\mathbf{G}} \in \mathcal{G}^q(C)$ for some $C \in N$, which implies $C \in \mathbf{F}_\bot$. Now define $\bot' := C$. □

Since the first components of $\mathit{Red}^{\cap}$ and $\mathit{Red}^{\cap,\sqsupset}$ agree, we obtain the analogues of Lemmas 38 and 39 and Theorem 40:

lem:intersect-saturation-indep-of-sqsubset

Lemma 46. *A set $N \subseteq \mathbf{F}$ is saturated w.r.t. FInf and $\mathit{Red}^{\cap}$ if and only if it is saturated w.r.t. FInf and $\mathit{Red}^{\cap,\sqsupset}$.*

lem:intersect-static-ref-compl-indep-of-sqsubset

Lemma 47. *$(\mathit{FInf}, \mathit{Red}^{\cap,\sqsupset})$ is statically refutationally complete w.r.t. $\models^{\cap}$ if and only if $(\mathit{FInf}, \mathit{Red}^{\cap})$ is statically refutationally complete w.r.t. $\models^{\cap}$.*

thm:intersect-static-ref-compl-is-dyn-ref-compl-with-order

Theorem 48. *If $(\mathit{FInf}, \mathit{Red}^{\cap})$ is statically refutationally complete w.r.t. $\models^{\cap}$, then $(\mathit{FInf}, \mathit{Red}^{\cap,\sqsupset})$ is dynamically refutationally complete. w.r.t. $\models^{\cap}$.*

Example 49. Intersections of liftings are needed to support selection functions in ordered resolution. The calculus FInf is parameterized by a function *fsel* on the set $\mathbf{F}$ of first-order clauses that selects a subset of the negative literals in each $C \in \mathbf{F}$. There are several ways to extend *fsel* to a selection function *gsel* on the set $\mathbf{G}$ of ground clauses such that for every $D \in \mathbf{G}$ there exists some $C \in \mathbf{F}$ such that $D = C\theta$ and D and C have corresponding selected literals. For every such *gsel*, $\models^{\mathit{gsel}}$ is first-order entailment, $\mathit{GInf}^{\mathit{gsel}}$ is the set of ground inferences satisfying *gsel*, and $\mathit{Red}^{\mathit{gsel}} = (\mathit{Red}^{\mathit{gsel}}_{\mathrm{I}}, \mathit{Red}^{\mathit{gsel}}_{\mathrm{F}})$ is the redundancy criterion for $\mathit{GInf}^{\mathit{gsel}}$. The grounding function $\mathcal{G}^{\mathit{gsel}}$ maps $C \in \mathbf{F}$ to $\{C\theta \in \mathbf{G} \mid \theta \text{ is a substitution}\}$ and $\iota \in \mathit{FInf}$ to the set of ground instances of ι in $\mathit{GInf}^{\mathit{gsel}}$ with corresponding literals selected in the premises. In the static refutational completeness proof, only one *gsel* is needed, but this *gsel* is not known during a derivation, so fairness must be guaranteed w.r.t. $\mathit{Red}^{\mathit{gsel},\mathcal{G}^{\mathit{gsel}}}_{\mathrm{I}}$ for every possible extension *gsel* of *fsel*.

Example 50. Intersections of liftings are also necessary for constraint superposition calculi. In this case, the calculus FInf operates on the set $\mathbf{F}$ of first-order clauses with (ordering and equality) constraints. For a convergent rewrite system R, $\models^R$ is first-order entailment up to R on the set $\mathbf{G}$ of unconstrained ground clauses, GInf^R is the set of ground superposition inferences, and $\mathit{Red}^R = (\mathit{Red}^R_{\mathrm{I}}, \mathit{Red}^R_{\mathrm{F}})$ is redundancy up to R. The grounding function $\mathcal{G}^R$ maps $C[\![K]\!] \in \mathbf{F}$ to $\{D \in \mathbf{G} \mid D = C\theta,\ K\theta = \text{true},\ x\theta \text{ is } R\text{-irreducible for all } x\}$ and $\iota \in \mathit{FInf}$ to the set of ground instances of ι where the premises and conclusion of $\mathcal{G}^R(\iota)$ are the $\mathcal{G}^R$-ground instances of the premises and conclusion of ι. In the static

refutational completeness proof, only one particular R is needed, but this R is not known during a derivation, so fairness must be guaranteed w.r.t. $Red_{\mathrm{I}}^{R,\mathcal{G}^R}$ for every convergent rewrite system R.

Almost every redundancy criterion for a nonground inference system *FInf* that can be found in the literature can be written as $Red^{\mathcal{G},\emptyset}$ for some grounding function $\mathcal{G}$ from $\mathbf{F}$ and *FInf* to $\mathbf{G}$ and *GInf*, and some redundancy criterion *Red* for *GInf*, or as an intersection $Red^{\cap}$ of such criteria. As Theorem 48 demonstrates, every static refutational completeness result for *FInf* and $Red^{\cap}$—which does not permit the deletion of subsumed formulas during a run—yields immediately a dynamic refutational completeness result for *FInf* and $Red^{\cap,\sqsupset}$—which permits the deletion of subsumed formulas during a run, provided they are larger w.r.t. $\sqsupset$.

3.4 Adding Labels

In practice, the orderings $\sqsupset_D$ used in $(\mathcal{G},\sqsupset)$-lifting often depend on meta-information about a formula, such as its age or the way in which it has been processed so far during a derivation. To capture this meta-information, we extend formulas and inference systems in a rather trivial way with labels.

As before, let $\mathbf{F}$ and $\mathbf{G}$ be two sets of formulas, let *FInf* be an $\mathbf{F}$-inference system, let *GInf* be a $\mathbf{G}$-inference system, let $\models \subseteq \mathcal{P}(\mathbf{G}) \times \mathcal{P}(\mathbf{G})$ be a consequence relation over $\mathbf{G}$, let $Red = (Red_{\mathrm{I}}, Red_{\mathrm{F}})$ be a redundancy criterion for *GInf* and $\models$, and let $\mathcal{G}$ be a grounding function from $\mathbf{F}$ and *FInf* to $\mathbf{G}$ and *GInf*.

Let $\mathbf{L}$ be a nonempty set of *labels*. Define $\mathbf{FL} := \mathbf{F}\times\mathbf{L}$ and $\mathbf{FL}_{\bot} := \mathbf{F}_{\bot}\times\mathbf{L}$. We use $\mathcal{M},\mathcal{N}$ to denote labeled formula sets. Given a set $\mathcal{N} \subseteq \mathbf{FL}$, let $\lfloor\mathcal{N}\rfloor := \{\, C \mid (C,l) \,\}$ denote the set of formulas without their labels. Given a set $\mathcal{N}$ and a label l, we define the projection $\mathcal{N}{\downarrow_l}$ as consisting only of the formulas labeled by l.

We call an $\mathbf{FL}$-inference system *FLInf* a *labeled version* of *FInf* if it has the following properties:

(i) for every inference $(C_n,\ldots,C_0) \in \mathit{FInf}$ and every tuple $(l_1,\ldots,l_n) \in \mathbf{L}^n$, there exists an $l_0 \in \mathbf{L}$ and an inference $((C_n,l_n),\ldots,(C_0,l_0)) \in \mathit{FLInf}$;

(ii) if $\iota = ((C_n,l_n),\ldots,(C_0,l_0))$ is an inference in *FLInf*, then $(C_n,\ldots,C_0)$ is an inference in *FInf*, denoted by $\lfloor\iota\rfloor$.

In other words, whenever there is an *FInf*-inference from some premises, there is a corresponding *FLInf*-inference from the labeled premises (regardless of the labeling), and whenever there is an *FLInf*-inference from labeled premises, there is a corresponding *FInf*-inference from the unlabeled premises.

Let *FLInf* be a labeled version of *FInf*. We define the function $\mathcal{G}_{\mathbf{L}}$ by $\mathcal{G}_{\mathbf{L}}((C,l)) := \mathcal{G}(C)$ for every $(C,l) \in \mathbf{FL}$ and by $\mathcal{G}_{\mathbf{L}}(\iota) := \mathcal{G}(\lfloor\iota\rfloor)$ for every $\iota \in \mathit{FLInf}$. The following lemmas are then obvious:

lem:labeled-grounding-function

Lemma 51. *$\mathcal{G}_{\mathbf{L}}$ is a grounding function from $\mathbf{FL}$ and FLInf to $\mathbf{G}$ and GInf.*

Let $\models_{\mathcal{G}_{\mathbf{L}}}$ be the $\mathcal{G}_{\mathbf{L}}$-lifting of $\models$. Let $Red^{\mathcal{G}_{\mathbf{L}},\emptyset} = (Red_{\mathrm{I}}^{\mathcal{G}_{\mathbf{L}}}, Red_{\mathrm{F}}^{\mathcal{G}_{\mathbf{L}},\emptyset})$ be the $(\mathcal{G}_{\mathbf{L}},\emptyset)$-lifting of *Red*.

lem:labeled-consequence

Lemma 52. *$\mathcal{N} \models_{\mathcal{G}_\mathbf{L}} \mathcal{N}'$ if and only if $\lfloor \mathcal{N} \rfloor \models_{\mathcal{G}} \lfloor \mathcal{N}' \rfloor$.*

lem:labeled-saturation

Lemma 53. *If a set $\mathcal{N} \subseteq \mathbf{FL}$ is saturated w.r.t. FLInf and $Red^{\mathcal{G}_\mathbf{L},\emptyset}$, then $\lfloor \mathcal{N} \rfloor \subseteq \mathbf{F}$ is saturated w.r.t. FInf and $Red^{\mathcal{G},\emptyset}$.*

lem:labeled-static-ref-compl

Lemma 54. *If $(FInf, Red^{\mathcal{G},\emptyset})$ is statically refutationally complete w.r.t. $\models_{\mathcal{G}}$, then $(FLInf, Red^{\mathcal{G}_\mathbf{L},\emptyset})$ is statically refutationally complete w.r.t. $\models_{\mathcal{G}_\mathbf{L}}$.*

The extension to intersections of redundancy criteria is also straightforward. Let $\mathbf{F}$ and $\mathbf{G}$ be two sets of formulas, and let *FInf* be an $\mathbf{F}$-inference system. Let Q be a set. For every $q \in Q$ let $\models^q$ be a consequence relation over $\mathbf{G}$, let $GInf^q$ be a $\mathbf{G}$-inference system, let $Red^q = (Red^q_{\mathrm{I}}, Red^q_{\mathrm{F}})$ be a redundancy criterion for $GInf^q$ and $\models^q$, and let $\mathcal{G}^q$ be a grounding function from $\mathbf{F}$ and *FInf* to $\mathbf{G}$ and $GInf^q$. Then for every $q \in Q$, the $(\mathcal{G}^q, \emptyset)$-lifting $Red^{q,\mathcal{G}^q,\emptyset} = (Red^{q,\mathcal{G}^q}_{\mathrm{I}}, Red^{q,\mathcal{G}^q,\emptyset}_{\mathrm{F}})$ of Red^q is a redundancy criterion for *FInf* and the $\mathcal{G}^q$-lifting $\models^q_{\mathcal{G}^q}$ of $\models^q$, and so, the intersection

$$Red^{\cap} = (Red^{\cap}_{\mathrm{I}}, Red^{\cap}_{\mathrm{F}}) := \Big(\bigcap_{q \in Q} Red^{q,\mathcal{G}^q}_{\mathrm{I}},\ \bigcap_{q \in Q} Red^{q,\mathcal{G}^q,\emptyset}_{\mathrm{F}} \Big)$$

is a redundancy criterion for $\models^{\cap} := \bigcap_{q \in Q} \models^q_{\mathcal{G}^q}$.

Now let $\mathbf{L}$ be a nonempty set of labels, and define $\mathbf{FL}$, $\mathbf{FL}_\bot$, and *FLInf* as above. For every $q \in Q$, define the function $\mathcal{G}^q_\mathbf{L}$ by $\mathcal{G}^q_\mathbf{L}((C, l)) := \mathcal{G}^q(C)$ for every $(C, l) \in \mathbf{FL}$ and by $\mathcal{G}^q_\mathbf{L}(\iota) := \mathcal{G}^q(\lfloor \iota \rfloor)$ for every $\iota \in FLInf$. By Lemma 51, every $\mathcal{G}^q_\mathbf{L}$ is a grounding function from $\mathbf{FL}$ and *FLInf* to $\mathbf{G}$ and $GInf^q$. Then for every $q \in Q$, the $(\mathcal{G}^q_\mathbf{L}, \emptyset)$-lifting $Red^{q,\mathcal{G}^q_\mathbf{L}} = (Red^{q,\mathcal{G}^q_\mathbf{L}}_{\mathrm{I}}, Red^{q,\mathcal{G}^q_\mathbf{L},\emptyset}_{\mathrm{F}})$ of Red^q is a redundancy criterion for *FLInf* and the $\mathcal{G}^q_\mathbf{L}$-lifting $\models^q_{\mathcal{G}^q_\mathbf{L}}$ of $\models^q$, and so, the intersection

$$Red^{\mathbf{L},\cap} = (Red^{\mathbf{L},\cap}_{\mathrm{I}}, Red^{\mathbf{L},\cap}_{\mathrm{F}}) := \Big(\bigcap_{q \in Q} Red^{q,\mathcal{G}^q_\mathbf{L}}_{\mathrm{I}},\ \bigcap_{q \in Q} Red^{q,\mathcal{G}^q_\mathbf{L},\emptyset}_{\mathrm{F}} \Big)$$

is a redundancy criterion for $\models^{\cap}_\mathbf{L} := \bigcap_{q \in Q} \models^q_{\mathcal{G}^q_\mathbf{L}}$.

Analogously to Lemmas 52–54, we obtain the following results:

lem:labeled-consequence-intersection

Lemma 55. *$\mathcal{N} \models^{\mathbf{L},\cap} \mathcal{N}'$ if and only if $\lfloor \mathcal{N} \rfloor \models^{\cap} \lfloor \mathcal{N}' \rfloor$.*

lem:labeled-saturation-intersection

Lemma 56. *If a set $\mathcal{N} \subseteq \mathbf{FL}$ is saturated w.r.t. FLInf and $Red^{\mathbf{L},\cap}$, then $\lfloor \mathcal{N} \rfloor \subseteq \mathbf{F}$ is saturated w.r.t. FInf and $Red^{\cap}$.*

thm:labeled-static-ref-compl-intersection

Theorem 57. *If $(FInf, Red^{\cap})$ is statically refutationally complete w.r.t. $\models^{\cap}$, then $(FLInf, Red^{\mathbf{L},\cap})$ is statically refutationally complete w.r.t. $\models^{\mathbf{L},\cap}$.*

4 Prover Architectures

We now use the above results to prove the refutational completeness of a popular prover architecture: the given clause procedure [18]. The architecture is parameterized by an inference system and a redundancy criterion. A generalization of the architecture decouples scheduling and computation of inferences, which has several benefits.

4.1 Given Clause Procedure

For this section, we assume the following. Let $\mathbf{F}$ and $\mathbf{G}$ be two sets of formulas, and let *FInf* be an $\mathbf{F}$-inference system without premise-free inferences. Let Q be a set. For every $q \in Q$ let $\models^q$ be a consequence relation over $\mathbf{G}$, let $GInf^q$ be a $\mathbf{G}$-inference system, let $Red^q = (Red^q_{\mathrm{I}}, Red^q_{\mathrm{F}})$ be a redundancy criterion for $GInf^q$ and $\models^q$, and let $\mathcal{G}^q$ be a grounding function from $\mathbf{F}$ and *FInf* to $\mathbf{G}$ and $GInf^q$. Let $(FInf, Red^{\cap})$ be statically refutationally complete w.r.t. $\models^{\cap}$, where $\models^{\cap} := \bigcap_{q\in Q} \models^q_{\mathcal{G}^q}$ and

$$Red^{\cap} = (Red^{\cap}_{\mathrm{I}}, Red^{\cap}_{\mathrm{F}}) := \Big(\bigcap_{q\in Q} Red^{q,\mathcal{G}^q}_{\mathrm{I}},\ \bigcap_{q\in Q} Red^{q,\mathcal{G}^q,\emptyset}_{\mathrm{F}}\Big).$$

Let $\mathbf{L}$ be a nonempty set of *labels*, let $\mathbf{FL} := \mathbf{F} \times \mathbf{L}$, and let the $\mathbf{FL}$-inference system *FLInf* be a labeled version of *FInf*. By Theorem 57, $(FLInf, Red^{\mathbf{L},\cap})$ is statically refutationally complete w.r.t. $\models^{\mathbf{L},\cap}$, where

$$Red^{\mathbf{L},\cap} = (Red^{\mathbf{L},\cap}_{\mathrm{I}}, Red^{\mathbf{L},\cap}_{\mathrm{F}}) := \Big(\bigcap_{q\in Q} Red^{q,\mathcal{G}^q_{\mathbf{L}}}_{\mathrm{I}},\ \bigcap_{q\in Q} Red^{q,\mathcal{G}^q_{\mathbf{L}},\emptyset}_{\mathrm{F}}\Big).$$

Let $\doteq$ be an equivalence relation on $\mathbf{F}$, let $\succ$ be a well-founded strict partial ordering on $\mathbf{F}$ such that $\succ$ is compatible with $\doteq$ (i.e., $C \succ D$, $C \doteq C'$, $D \doteq D'$ implies $C' \succ D'$), such that $C \doteq D$ implies $\mathcal{G}^q(C) = \mathcal{G}^q(D)$ for all $q \in Q$, and such that $C \succ D$ implies $\mathcal{G}^q(C) \subseteq \mathcal{G}^q(D)$ for all $q \in Q$. We define $\succeq$ as $\succ \cup \doteq$. In practice, $\doteq$ is typically α-renaming, $\succ$ is either the subsumption ordering $\succcurlyeq$ (provided it is well founded) or some well-founded ordering included in $\succcurlyeq$, and for every $q \in Q$, $\mathcal{G}^q$ maps every formula $C \in \mathbf{F}$ to the set of ground instances of C.

Let $\sqsupset$ be a well-founded strict partial ordering on $\mathbf{L}$. We define the ordering $\sqsupset$ on $\mathbf{FL}$ by $(C, l) \sqsupset (C', l')$ if either $C \succ C'$ or else $C \doteq C'$ and $l \sqsupset l'$. By Lemma 47, the static refutational completeness of $(FLInf, Red^{\mathbf{L},\cap})$ w.r.t. $\models^{\mathbf{L},\cap}$ implies the static refutational completeness of $(FLInf, Red^{\mathbf{L},\cap,\sqsupset})$, where

$$Red^{\mathbf{L},\cap,\sqsupset} = (Red^{\mathbf{L},\cap}_{\mathrm{I}}, Red^{\mathbf{L},\cap,\sqsupset}_{\mathrm{F}}) := \Big(\bigcap_{q\in Q} Red^{q,\mathcal{G}^q_{\mathbf{L}}}_{\mathrm{I}},\ \bigcap_{q\in Q} Red^{q,\mathcal{G}^q_{\mathbf{L}},\sqsupset}_{\mathrm{F}}\Big),$$

and by Lemma 6, the latter implies the dynamic refutational completeness of $(FLInf, Red^{\mathbf{L},\cap,\sqsupset})$.

This result may look a bit intimidating, so let us try to unroll it a bit. The $\mathbf{FL}$-inference system *FLInf* is a labeled version of *FInf*, which means that we get an *FLInf*-inference by first omitting the labels of the premises $(C_n, l_n), \ldots, (C_1, l_1)$, then performing an *FInf*-inference $(C_n, \ldots, C_0)$, and finally attaching an arbitrary label l_0 to the conclusion C_0. Since the labeled grounding functions $\mathcal{G}^q_{\mathbf{L}}$ differ from the corresponding unlabeled grounding functions $\mathcal{G}^q$ only by the omission of the labels and since the first components of $Red^{\mathbf{L},\cap,\sqsupset}$ and $Red^{\mathbf{L},\cap}$ agree, we get this result:

lem:redundant-labeled-inferences

Lemma 58. *An FLInf-inference ι is redundant w.r.t. $Red^{\mathbf{L},\cap,\sqsupset}$ and $\mathcal{N}$ if and only if the underlying FInf-inference $\lfloor\iota\rfloor$ is redundant w.r.t. $Red^{\cap}$ and $\lfloor\mathcal{N}\rfloor$.*

For $Red_{\mathrm{F}}^{\mathbf{L},\cap,\sqsupset}$, we can show that a labeled formula (C, l) is redundant if (i) C itself is redundant w.r.t. $Red_{\mathrm{F}}^{\cap}$, or if (ii) C is $\succ$-subsumed, or if (iii) C is a variant of another formula that occurs with a $\sqsupset$-smaller label. More formally:

lem:redundant-labeled-formulas

Lemma 59. *Let $\mathcal{N} \subseteq \mathbf{FL}$, and let (C, l) be a labeled formula. Then $(C, l) \in Red_{\mathrm{F}}^{\mathbf{L},\cap,\sqsupset}(\mathcal{N})$ if one of the following conditions hold:*

(i) $C \in Red_{\mathrm{F}}^{\cap}(\lfloor \mathcal{N} \rfloor)$;
(ii) $C \succ C'$ *for some* $C' \in \lfloor \mathcal{N} \rfloor$;
(iii) $C \succeq C'$ *for some* $(C', l') \in \mathcal{N}$ *with* $l \sqsupset l'$.

Proof. (i) Let $C \in Red_{\mathrm{F}}^{\cap}(\lfloor \mathcal{N} \rfloor)$. Then $C \in Red_{\mathrm{F}}^{q,\mathcal{G}^q,\emptyset}(\lfloor \mathcal{N} \rfloor)$ for every $q \in Q$, which means that $\mathcal{G}^q(C) \subseteq Red_{\mathrm{F}}^{q}(\mathcal{G}^q(\lfloor \mathcal{N} \rfloor))$. Now $\mathcal{G}_{\mathbf{L}}^{q}((C, l)) = \mathcal{G}^q(C)$ and $\mathcal{G}^q(\lfloor \mathcal{N} \rfloor) = \mathcal{G}_{\mathbf{L}}^{q}(\mathcal{N})$; hence $\mathcal{G}_{\mathbf{L}}^{q}((C, l)) \subseteq Red_{\mathrm{F}}^{q}(\mathcal{G}_{\mathbf{L}}^{q}(\mathcal{N}))$, which implies $(C, l) \in Red_{\mathrm{F}}^{q,\mathcal{G}_{\mathbf{L}}^{q},\sqsupset}(\mathcal{N})$ for every $q \in Q$ and thus $(C, l) \in Red_{\mathrm{F}}^{\mathbf{L},\cap,\sqsupset}(\mathcal{N})$.

(ii) Assume that $C \succ C'$ for some $C' \in \lfloor \mathcal{N} \rfloor$. Then there exists a label l' such that $(C', l') \in \mathcal{N}$. By the definition of $\sqsupset$, we have $(C, l) \sqsupset (C', l')$. Furthermore, $\mathcal{G}^q(C) \subseteq \mathcal{G}^q(C')$ for all $q \in Q$. Therefore $\mathcal{G}_{\mathbf{L}}^{q}((C, l)) = \mathcal{G}^q(C) \subseteq \mathcal{G}^q(C') = \mathcal{G}_{\mathbf{L}}^{q}((C', l'))$, which implies $(C, l) \in Red_{\mathrm{F}}^{q,\mathcal{G}_{\mathbf{L}}^{q},\sqsupset}(\mathcal{N})$ for every $q \in Q$ and thus $(C, l) \in Red_{\mathrm{F}}^{\mathbf{L},\cap,\sqsupset}(\mathcal{N})$.

(iii) If $C \succ C'$, the result follows from (ii). Otherwise $C \doteq C'$ for some $(C', l') \in \mathcal{N}$ with $l \sqsupset l'$. Then $(C, l) \sqsupset (C', l')$ and $\mathcal{G}^q(C) = \mathcal{G}^q(C')$, so $\mathcal{G}_{\mathbf{L}}^{q}((C, l)) = \mathcal{G}^q(C) = \mathcal{G}^q(C') = \mathcal{G}_{\mathbf{L}}^{q}((C', l'))$. This implies $(C, l) \in Red_{\mathrm{F}}^{q,\mathcal{G}_{\mathbf{L}}^{q},\sqsupset}(\mathcal{N})$ for every $q \in Q$; therefore, $(C, l) \in Red_{\mathrm{F}}^{\mathbf{L},\cap,\sqsupset}(\mathcal{N})$. □

The given clause procedure that lies at the heart of saturation provers can be presented and studied abstractly.[5] We assume that the set of labels $\mathbf{L}$ contains at least two values, one of which is a distinguished $\sqsupset$-smallest value denoted by active, and that the labeled version *FLInf* of *FInf* never assigns the label active to a conclusion.

The state of a prover is a set of labeled formulas (i.e., of formula–label pairs). The label identifies to which formula set each formula belongs. The active label identifies the active formula set from the familiar given clause procedure. The other, unspecified formula sets are considered passive.

The given clause prover GC is defined as the following transition system:

PROCESS $\mathcal{N} \uplus \mathcal{M} \Longrightarrow_{\mathsf{GC}} \mathcal{N} \cup \mathcal{M}'$
  where $\mathcal{M} \subseteq Red_{\mathrm{F}}^{\mathbf{L},\cap,\sqsupset}(\mathcal{N} \cup \mathcal{M}')$ and $\mathcal{M}'{\downarrow_{\mathsf{active}}} = \emptyset$

INFER $\mathcal{N} \uplus \{(C, l)\} \Longrightarrow_{\mathsf{GC}} \mathcal{N} \cup \{(C, \mathsf{active})\} \cup \mathcal{M}$
  where $l \neq \mathsf{active}$, $\mathcal{M}{\downarrow_{\mathsf{active}}} = \emptyset$, and
  $FInf(\lfloor \mathcal{N}{\downarrow_{\mathsf{active}}} \rfloor, \{C\}) \subseteq Red_{\mathrm{I}}^{\cap}(\lfloor \mathcal{N} \rfloor \cup \{C\} \cup \lfloor \mathcal{M} \rfloor)$

[5] We keep the traditional term "given clause procedure" even though our framework is not restricted to clauses.

INFER is the only rule that puts a formula in the active set. It relabels a passive formula C to active and ensures that all inferences between C and the active formulas (including C itself) become redundant. Recall that by Lemma 58, $\mathit{FLInf}(\mathcal{N}{\downarrow_{\mathsf{active}}}, \{(C, \mathsf{active})\}) \subseteq \mathit{Red}_{\mathrm{I}}^{\mathbf{L},\cap}(\mathcal{N} \cup \{(C, \mathsf{active})\} \cup \mathcal{M})$ if and only if $\mathit{FInf}(\lfloor \mathcal{N}{\downarrow_{\mathsf{active}}} \rfloor, \{C\}) \subseteq \mathit{Red}_{\mathrm{I}}^{\cap}(\lfloor \mathcal{N} \rfloor \cup \{C\} \cup \lfloor \mathcal{M} \rfloor)$. By property (R4) of redundancy criteria, every inference is redundant if its conclusion is contained in the set of formulas, and typically, inferences are in fact made redundant by adding their conclusions to any of the passive sets. Then, $\lfloor \mathcal{M} \rfloor$ equals $\mathit{concl}(\mathit{FInf}(\lfloor \mathcal{N}{\downarrow_{\mathsf{active}}} \rfloor, \{C\}))$. There are some techniques commonly implemented in theorem provers, however, for which we need INFER's side condition in full generality.

The PROCESS rule covers the other operations performed in a theorem prover. By Lemma 59, this includes

- deleting $\mathit{Red}_{\mathrm{F}}^{\cap}$-redundant formulas with arbitrary labels and adding formulas that make other formulas $\mathit{Red}_{\mathrm{F}}^{\cap}$-redundant (i.e., simplifying w.r.t. $\mathit{Red}_{\mathrm{F}}^{\cap}$), by (i);
- deleting formulas that are $\dot{\succ}$-subsumed by other formulas with arbitrary labels, by (ii);
- deleting formulas that are $\dot{\succeq}$-subsumed by other formulas with smaller labels, by (iii);
- replacing the label of a formula by a smaller label different from active, again by (iii).

lem:gc-derivations-are-red-derivations

Lemma 60. *Every $\Longrightarrow_{\mathsf{GC}}$-derivation is a $\rhd_{\mathit{Red}^{\mathbf{L},\cap,\sqsupset}}$-derivation.*

Proof. We must show that every labeled formula that is deleted in a $\Longrightarrow_{\mathsf{GC}}$-step is $\mathit{Red}^{\mathbf{L},\cap,\sqsupset}$-redundant w.r.t. the remaining labeled formulas. For PROCESS, this is trivial. For INFER, the only deleted formula is (C, l), which is $\mathit{Red}^{\mathbf{L},\cap,\sqsupset}$-redundant w.r.t. (C, active) by part (iii) of Lemma 59, since $l \sqsupset \mathsf{active}$. □

Since $(\mathit{FLInf}, \mathit{Red}^{\mathbf{L},\cap,\sqsupset})$ is dynamically refutationally complete, it now suffices to show fairness to prove the refutational completeness of GC.

lem:fair-gc-derivations

Lemma 61. *Let $(\mathcal{N}_i)_i$ be a $\Longrightarrow_{\mathsf{GC}}$-derivation. If $\mathcal{N}_0{\downarrow_{\mathsf{active}}} = \emptyset$ and $\mathcal{N}_*{\downarrow_l} = \emptyset$ for all $l \neq \mathsf{active}$, then $(\mathcal{N}_i)_i$ is a fair $\rhd_{\mathit{Red}^{\mathbf{L},\cap,\sqsupset}}$-derivation.*

Proof. We must show that $\mathit{FLInf}(\mathcal{N}_*) \subseteq \bigcup_i \mathit{Red}_{\mathrm{I}}^{\mathbf{L},\cap}(\mathcal{N}_i)$. First observe that $\mathcal{N}_* = \bigcup_{l \in \mathbf{L}} \mathcal{N}_*{\downarrow_l}$, so if $\mathcal{N}_*{\downarrow_l} = \emptyset$ for all $l \neq \mathsf{active}$, then $\mathcal{N}_* = \mathcal{N}_*{\downarrow_{\mathsf{active}}}$. Let ι' be an arbitrary inference in $\mathit{FLInf}(\mathcal{N}_*{\downarrow_{\mathsf{active}}})$, and let (C_j, active) for $1 \leq j \leq m$ be the finitely many premises of ι'. Since each premise is contained in $\mathcal{N}_*{\downarrow_{\mathsf{active}}}$ and $\mathcal{N}_0{\downarrow_{\mathsf{active}}} = \emptyset$, we know that for each j there exists some n_j such that $(C_j, \mathsf{active}) \in \mathcal{N}_k{\downarrow_{\mathsf{active}}}$ for all $k \geq n_j$ and $(C_j, \mathsf{active}) \notin \mathcal{N}_{n_j - 1}{\downarrow_{\mathsf{active}}}$. Let $n = \max\{n_j \mid 1 \leq j \leq m\}$ and assume that $n = n_{j_0}$. Since in every $\Longrightarrow_{\mathsf{GC}}$-step at most one formula can have its label changed to active, we know that the step $\mathcal{N}_{n-1} \Longrightarrow_{\mathsf{GC}} \mathcal{N}_n$ must be an INFER step

$$\mathcal{N}_{n-1} = \mathcal{N} \uplus \{(C, l)\} \Longrightarrow_{\mathsf{GC}} \mathcal{N} \cup \{(C, \mathsf{active})\} \cup \mathcal{M} = \mathcal{N}_n,$$

where $C = C_{j_0}$ and all other premises of ι' are contained in $\mathcal{N}{\downarrow_{\mathsf{active}}} \cup \{(C, \mathsf{active})\}$. By Infer's side condition, $\iota = \lfloor \iota' \rfloor \in \mathit{FInf}(\lfloor \mathcal{N}{\downarrow_{\mathsf{active}}} \rfloor, \{C\}) \subseteq \mathit{Red}_{\mathrm{I}}^{\cap}(\lfloor \mathcal{N}_n \rfloor)$, hence $\iota' \in \mathit{Red}_{\mathrm{I}}^{\mathbf{L},\cap}(\mathcal{N}_n) \subseteq \bigcup_i \mathit{Red}_{\mathrm{I}}^{\mathbf{L},\cap}(\mathcal{N}_i)$, as required. □

thm:gc-completeness

Theorem 62. *Let $(\mathcal{N}_i)_i$ be a $\Longrightarrow_{\mathsf{GC}}$-derivation, where $\mathcal{N}_0{\downarrow_{\mathit{active}}} = \emptyset$ and $\mathcal{N}_*{\downarrow_l} = \emptyset$ for all $l \neq \mathsf{active}$. If $\lfloor \mathcal{N}_0 \rfloor \models^{\cap} \{\bot\}$ for some $\bot \in \mathbf{F}_\bot$, then some $\mathcal{N}_i$ contains $(\bot', l)$ for some $\bot' \in \mathbf{F}_\bot$ and $l \in \mathbf{L}$.*

Proof. By Lemma 55, $\lfloor \mathcal{N}_0 \rfloor \models^{\cap} \{\bot\}$ is equivalent to $\mathcal{N}_0 \models^{\mathbf{L},\cap} \{(\bot, \mathsf{active})\}$. By Lemma 61, we know that $(\mathcal{N}_i)_i$ is a fair $\rhd_{\mathit{Red}^{\mathbf{L},\cap,\sqsupset}}$-derivation. Since $(\mathit{FLInf}, \mathit{Red}^{\mathbf{L},\cap,\sqsupset})$ is dynamically refutationally complete, we can conclude that some $\mathcal{N}_i$ contains $(\bot', l)$ for some $\bot' \in \mathbf{F}_\bot$ and $l \in \mathbf{L}$. □

Example 63. The following Otter loop [18, Sect. 2.3.1] prover OL is an instance of the given clause prover GC. This loop design is inspired by Weidenbach's prover without splitting from his *Handbook* chapter [28, Tables 4–6]. The prover's state is a five-tuple $N \mid X \mid P \mid Y \mid A$ of formula sets. The N, P, and A sets store the new, passive, and active formulas, respectively. The X and Y sets are subsingletons (i.e., sets of at most one element) that can store a chosen new or passive formula, respectively. Initial states are of the form $N \mid \emptyset \mid \emptyset \mid \emptyset \mid \emptyset$.

ChooseN $N \uplus \{C\} \mid \emptyset \mid P \mid \emptyset \mid A \Longrightarrow_{\mathsf{OL}} N \mid \{C\} \mid P \mid \emptyset \mid A$

DeleteFwd $N \mid \{C\} \mid P \mid \emptyset \mid A \Longrightarrow_{\mathsf{OL}} N \mid \emptyset \mid P \mid \emptyset \mid A$
if $C \in \mathit{Red}_{\mathrm{F}}^{\cap}(P \cup A)$ or $C \mathrel{\dot{\succeq}} C'$ for some $C' \in P \cup A$

SimplifyFwd $N \mid \{C\} \mid P \mid \emptyset \mid A \Longrightarrow_{\mathsf{OL}} N \mid \{C'\} \mid P \mid \emptyset \mid A$
if $\{C\} \cup P \cup A \mathrel{\dot{\approx}} \{C'\}$ and $C \in \mathit{Red}_{\mathrm{F}}^{\cap}(P \cup A \cup \{C'\})$

DeleteBwdP $N \mid \{C\} \mid P \uplus \{C'\} \mid \emptyset \mid A \Longrightarrow_{\mathsf{OL}} N \mid \{C\} \mid P \mid \emptyset \mid A$
if $C' \in \mathit{Red}_{\mathrm{F}}^{\cap}(\{C\})$ or $C' \mathrel{\dot{\succ}} C$

SimplifyBwdP $N \mid \{C\} \mid P \uplus \{C'\} \mid \emptyset \mid A \Longrightarrow_{\mathsf{OL}} N \cup \{C''\} \mid \{C\} \mid P \mid \emptyset \mid A$
if $\{C, C'\} \mathrel{\dot{\approx}} \{C''\}$ and $C' \in \mathit{Red}_{\mathrm{F}}^{\cap}(\{C, C''\})$

DeleteBwdA $N \mid \{C\} \mid P \mid \emptyset \mid A \uplus \{C'\} \Longrightarrow_{\mathsf{OL}} N \mid \{C\} \mid P \mid \emptyset \mid A$
if $C' \in \mathit{Red}_{\mathrm{F}}^{\cap}(\{C\})$ or $C' \mathrel{\dot{\succ}} C$

SimplifyBwdA $N \mid \{C\} \mid P \mid \emptyset \mid A \uplus \{C'\} \Longrightarrow_{\mathsf{OL}} N \cup \{C''\} \mid \{C\} \mid P \mid \emptyset \mid A$
if $\{C, C'\} \mathrel{\dot{\approx}} \{C''\}$ and $C' \in \mathit{Red}_{\mathrm{F}}^{\cap}(\{C, C''\})$

Transfer $N \mid \{C\} \mid P \mid \emptyset \mid A \Longrightarrow_{\mathsf{OL}} N \mid \emptyset \mid P \cup \{C\} \mid \emptyset \mid A$

ChooseP $\emptyset \mid \emptyset \mid P \uplus \{C\} \mid \emptyset \mid A \Longrightarrow_{\mathsf{OL}} \emptyset \mid \emptyset \mid P \mid \{C\} \mid A$

Infer $\emptyset \mid \emptyset \mid P \mid \{C\} \mid A \Longrightarrow_{\mathsf{OL}} M \mid \emptyset \mid P \mid \emptyset \mid A \cup \{C\}$
if $A \cup \{C\} \mathrel{\dot{\approx}} M$ and $\mathit{FInf}(A, \{C\}) \subseteq \mathit{Red}_{\mathrm{I}}^{\cap}(A \cup \{C\} \cup M)$

Weidenbach identifies the X and Y components of OL's five-tuples; this is possible since the former is used only in his inner loop, whereas the latter is used only in his outer loop.

A reasonable strategy for applying the OL rules is described below. It relies on a well-founded ordering $\succ$ on formulas to make sure that the backward simplification rules, SimplifyBwdP and SimplifyBwdA, actually "simplify" their target, preventing nontermination of the inner loop. It also assumes that *FInf* is sound w.r.t. $\mathrel{\dot{\approx}}$ and that $\mathit{FInf}(N, \{C\})$ is finite if N is finite.

1. Repeat while $N \cup P \neq \emptyset$ and $\bot \notin N \cup P \cup A$:
   1.1. Repeat while $N \neq \emptyset$:
      1.1.1. Apply ChooseN to retrieve the next formula C from the state's N component, which is organized as a queue.
      1.1.2. Apply SimplifyFwd as long as the simplified formula C' is $\succ$-smaller than the original formula C.
      1.1.3. If DeleteFwd is applicable, apply it.
      1.1.4. Otherwise:
         1.1.4.1. Apply DeleteBwdP and DeleteBwdA exhaustively.
         1.1.4.2. Apply SimplifyBwdP and SimplifyBwdA as long as the simplified formula C'' is $\succ$-smaller than the original formula C'.
         1.1.4.3. Apply Transfer.
   1.2. If $P \neq \emptyset$:
      1.2.1. Apply ChooseP. Make sure that the choice of C is fair.
      1.2.2. Apply Infer with $M = \mathit{concl}(\mathit{FInf}(A, \{C\}))$.

Let $(N_i \mid X_i \mid P_i \mid Y_i \mid A_i)_i$ be a $\Longrightarrow_{\mathsf{OL}}$-derivation that follows the strategy, where N_0 is finite and $X_0 = P_0 = Y_0 = A_0 = \emptyset$. If the outer loop terminates because $\bot \in N \cup P \cup A$, the condition of dynamic refutational completeness is trivially satisfied. Otherwise, the argument is as follows. With each application of a rule other than Infer, the state, viewed as a multiset of labeled formulas, decreases w.r.t. the multiset extension of a relation that compares formulas using $\succ$ and breaks ties using $\sqsupset$ on the labels. This ensures no formula is left in N or X forever. The fair choice of C ensures that that no formula is left in P forever, and the application of Infer following ChooseP ensures the same about Y. As a result, we have $N_* = X_* = P_* = Y_* = \emptyset$. Therefore, by Theorem 62, OL is dynamically refutationally complete.

In most saturation calculi, *Red* is defined in terms of some total and well-founded ordering $\succ_{\mathbf{G}}$ on $\mathbf{G}$. We can then define $\succ$ so that $C \succ C'$ if the smallest element of $\mathcal{G}^q(C)$ is greater than the smallest element of $\mathcal{G}^q(C')$ w.r.t. $\succ_{\mathbf{G}}$, for some arbitrary fixed $q \in Q$. This allows a wide range of simplifications typically implemented in resolution or superposition provers.

To ensure fairness when applying ChooseP, one approach is to use an $\mathbb{N}$-valued weight function that is strictly antimonotone in the age of the formula [23, Sect. 4]. Another option is to alternate between heuristically choosing n formulas and taking the oldest formula [18, Sect. 2.3.1].

Example 64. Bachmair and Ganzinger's resolution prover RP [5, Sect. 4.3] is another instance of GC. It embodies both a concrete prover architecture and a concrete inference system: ordered resolution with selection ($\mathsf{O}^{\succ}_{S}$). States are triples $N \mid P \mid O$ of finite clause sets consisting of new, processed (passive), and old (active) clauses, respectively. The instantiation relies on three labels $l_1 \sqsupset l_2 \sqsupset l_3 = \mathsf{active}$. Subsumption can be supported as described in Example 41.

TAUTO $N \cup \{C\} \mid P \mid O \Longrightarrow_{\mathsf{RP}} N \mid P \mid O$
if C is a tautology

DELETEFWD $N \cup \{C\} \mid P \mid O \Longrightarrow_{\mathsf{RP}} N \mid P \mid O$
if some clause in $P \cup O$ subsumes C

REDUCEFWD $N \cup \{C \vee L\} \mid P \mid O \Longrightarrow_{\mathsf{RP}} N \cup \{C\} \mid P \mid O$
if there is a clause $D \vee L'$ in $P \cup O$ such that $\bar{L} = L'\sigma$ and $D\sigma \subseteq C$

DELETEBWDP $N \mid P \cup \{C\} \mid O \Longrightarrow_{\mathsf{RP}} N \mid P \mid O$
if some clause in N properly subsumes C

REDUCEBWDP $N \mid P \cup \{C \vee L\} \mid O \Longrightarrow_{\mathsf{RP}} N \mid P \cup \{C\} \mid O$
if there is a clause $D \vee L'$ in N such that $\bar{L} = L'\sigma$ and $D\sigma \subseteq C$

DELETEBWDO $N \mid P \mid O \cup \{C\} \Longrightarrow_{\mathsf{RP}} N \mid P \mid O$
if some clause in N properly subsumes C

REDUCEBWDO $N \mid P \mid O \cup \{C \vee L\} \Longrightarrow_{\mathsf{RP}} N \mid P \cup \{C\} \mid O$
if there is a clause $D \vee L'$ in N such that $\bar{L} = L'\sigma$ and $D\sigma \subseteq C$

CHOOSE $N \cup \{C\} \mid P \mid O \Longrightarrow_{\mathsf{RP}} N \mid P \cup \{C\} \mid O$

INFER $\emptyset \mid P \cup \{C\} \mid O \Longrightarrow_{\mathsf{RP}} N \mid P \mid O \cup \{C\}$
if $N = \mathit{concl}(\mathsf{O}_S^{\succ}(O, C))$

Let $(N_i \mid P_i \mid O_i)_i$ be a full $\Longrightarrow_{\mathsf{RP}}$-derivation, where $P_0 = O_0 = \emptyset$. Since the rule system excluding INFER terminates [23, Sect. 4] and we can always apply CHOOSE to empty N, we have $N_* = \emptyset$. The only restriction that is needed to ensure fairness is that the choice of C in INFER must be fair. This ensures $P_* = \emptyset$. As a result, by Theorem 62, RP is dynamically refutationally complete. Incidentally, our version of RP repairs a subtle flaw in Bachmair and Ganzinger's definition of the notation $\mathit{Inf}(N, \{C\})$, used in the INFER rule [24, Sect. 7].

4.2 Delayed Inferences

An *orphan* is a passive formula that was generated by an inference for which at least one premise is no longer active. The given clause prover GC presented in the previous section is sufficient to describe an Otter loop prover as well as a basic DISCOUNT loop prover, but to describe a DISCOUNT loop prover with orphan deletion, we need to decouple the scheduling of inferences and their computation. The same scheme can be used to model provers based on inference systems that contain premise-free inferences or that may generate infinitely many conclusions from finitely many premises. Yet another use of the scheme is to save memory: A delayed inference can be stored compactly as a tuple of premises together with instructions on how to compute the conclusion.

The lazy given clause prover LGC is defined as the following transition system on pairs $(T, \mathcal{N})$, where T ("to do") is a set of inferences and $\mathcal{N}$ is a set of labeled formulas. We use the same assumptions as in Section 4.1, except that we permit premise-free inferences in FInf.

PROCESS $(T, \mathcal{N} \uplus \mathcal{M}) \Longrightarrow_{\mathsf{LGC}} (T, \mathcal{N} \cup \mathcal{M}')$
where $\mathcal{M} \subseteq \mathit{Red}_{\mathrm{F}}^{\mathbf{L},\cap,\sqsupset}(\mathcal{N} \cup \mathcal{M}')$ and $\mathcal{M}'{\downarrow_{\mathsf{active}}} = \emptyset$

ScheduleInfer $(T, \mathcal{N} \uplus \{(C, l)\}) \Longrightarrow_{\mathsf{LGC}} (T \cup T', \mathcal{N} \cup \{(C, \mathsf{active})\})$
where $l \neq \mathsf{active}$ and $T' = \mathit{FInf}(\lfloor \mathcal{N}{\downarrow_{\mathsf{active}}} \rfloor, \{C\})$

ComputeInfer $(T \uplus \{\iota\}, \mathcal{N}) \Longrightarrow_{\mathsf{LGC}} (T, \mathcal{N} \cup \mathcal{M})$
where $\mathcal{M}{\downarrow_{\mathsf{active}}} = \emptyset$ and $\iota \in \mathit{Red}_{\mathrm{I}}^{\cap}(\lfloor \mathcal{N} \cup \mathcal{M} \rfloor)$

DeleteOrphans $(T \uplus T', \mathcal{N}) \Longrightarrow_{\mathsf{LGC}} (T, \mathcal{N})$
where $T' \cap \mathit{FInf}(\lfloor \mathcal{N}{\downarrow_{\mathsf{active}}} \rfloor) = \emptyset$

Process mimics the behavior of the corresponding GC rule, except for the additional T component. The Infer rule of GC is split into two parts in LGC: ScheduleInfer relabels a passive formula C to active and puts all inferences between C and the active formulas, including C itself, into the set T. ComputeInfer removes an inference from T and ensures that it becomes redundant by adding appropriate labeled formulas to $\mathcal{N}$ (typically the conclusion of the inference). DeleteOrphans can delete scheduled inferences from T if some of their premises have been deleted from $\mathcal{N}{\downarrow_{\mathsf{active}}}$ in the meantime by an application of Process. Note that DeleteOrphans cannot delete premise-free inferences.

Abstractly, the T component of the state is a set of inferences $(C_n, \ldots, C_0)$. In an actual implementation, it can be represented in different ways: as a set of compactly encoded recipes for computing the conclusion C_0 from the premises $(C_n, \ldots, C_1)$ as in Waldmeister [15], or as a set of explicit formulas C_0 with information about their ancestors $(C_n, \ldots, C_1)$ as in E [25]. In the latter case, some presimplifications may be performed on C_0; this could be modeled more faithfully by defining T as a set of pairs $(\iota, \mathit{simp}(C_0))$.

lem:lgc-derivations-are-red-derivations

Lemma 65. *If $(T_i, \mathcal{N}_i)_i$ is a $\Longrightarrow_{\mathsf{LGC}}$-derivation, then $(\mathcal{N}_i)_i$ is a $\rhd_{\mathit{Red}^{\mathbf{L},\cap,\sqsupset}}$-derivation.*

Proof. We must show that every labeled formula that is deleted in a $\Longrightarrow_{\mathsf{LGC}}$-step from the $\mathcal{N}$ component is $\mathit{Red}^{\mathbf{L},\cap,\sqsupset}$-redundant w.r.t. the remaining labeled formulas. For Process this is trivial. For ScheduleInfer, the only deleted formula is (C, l), which is $\mathit{Red}^{\mathbf{L},\cap,\sqsupset}$-redundant w.r.t. (C, active) by part (iii) of Lemma 59, since $l \sqsupset \mathsf{active}$. Finally, the rules ComputeInfer and DeleteOrphans do not delete any formulas. □

lem:fair-lgc-derivations

Lemma 66. *Let $(T_i, \mathcal{N}_i)_i$ be a $\Longrightarrow_{\mathsf{LGC}}$-derivation. If $\mathcal{N}_0{\downarrow_{\mathsf{active}}} = \emptyset$, $\mathcal{N}_*{\downarrow_l} = \emptyset$ for all $l \neq \mathsf{active}$, T_0 contains all premise-free inferences of FInf, and $T_* = \emptyset$, then $(\mathcal{N}_i)_i$ is a fair $\rhd_{\mathit{Red}^{\mathbf{L},\cap,\sqsupset}}$-derivation.*

Proof. We must show that $\mathit{FLInf}(\mathcal{N}_*) \subseteq \bigcup_i \mathit{Red}_{\mathrm{I}}^{\mathbf{L},\cap}(\mathcal{N}_i)$. Since $\mathcal{N}_*{\downarrow_l} = \emptyset$ for all $l \neq \mathsf{active}$, we have $\mathcal{N}_* = \mathcal{N}_*{\downarrow_{\mathsf{active}}}$. Let ι' be an arbitrary inference in $\mathit{FLInf}(\mathcal{N}_*{\downarrow_{\mathsf{active}}})$. We first prove that there exists some index n such that $\iota = \lfloor \iota' \rfloor \in T_n$. We distinguish two cases: If ι' has no premises, then ι has no premises either. So let $n = 0$, then $\iota \in T_n$ follows be assumption. Otherwise, let (C_j, active) for $1 \leq j \leq m$ be the finitely many premises of ι'. Since each premise is contained in $\mathcal{N}_*{\downarrow_{\mathsf{active}}}$ and $\mathcal{N}_0{\downarrow_{\mathsf{active}}} = \emptyset$, we know that for each j there exists some n_j such that $(C_j, \mathsf{active}) \in \mathcal{N}_k{\downarrow_{\mathsf{active}}}$ for all $k \geq n_j$ and $(C_j, \mathsf{active}) \notin \mathcal{N}_{n_j-1}{\downarrow_{\mathsf{active}}}$. Let $n = \max\{n_j \mid 1 \leq j \leq m\}$ and assume that $n = n_{j_0}$. Since in every $\Longrightarrow_{\mathsf{LGC}}$-step

at most one formula can have its label changed to active, we know that the step $\mathcal{N}_{n-1} \Longrightarrow_{\mathsf{LGC}} \mathcal{N}_n$ must be a ScheduleInfer step

$$(T_{n-1}, \mathcal{N}_{n-1}) = (T, \mathcal{N} \uplus \{(C, l)\}) \Longrightarrow_{\mathsf{LGC}} (T \cup T', \mathcal{N} \cup \{(C, \mathsf{active})\}) = (T_n, \mathcal{N}_n),$$

where $C = C_{j_0}$ and all other premises of ι' are contained in $\mathcal{N}{\downarrow_{\mathsf{active}}} \cup \{(C, \mathsf{active})\}$. By ScheduleInfer's side condition, $\iota = \lfloor \iota' \rfloor \in \mathit{FInf}(\lfloor \mathcal{N}{\downarrow_{\mathsf{active}}} \rfloor, \{C\}) = T' \subseteq T_n$.

In both cases, since $T_* = \emptyset$, there must exist some $p > n$ such that $\iota \in T_{p-1}$ and $\iota \notin T_p$. There are two rules that can be used to remove inferences from the first component—namely, ComputeInfer and DeleteOrphans—but the step $(T_{p-1}, \mathcal{N}_{p-1}) \Longrightarrow_{\mathsf{LGC}} (T_p, \mathcal{N}_p)$ cannot be a DeleteOrphans step, since all premises of ι are contained in $\lfloor \mathcal{N}_{p-1}{\downarrow_{\mathsf{active}}} \rfloor$. So ι is deleted by a ComputeInfer step

$$(T_{p-1}, \mathcal{N}_{p-1}) = (T \uplus \{\iota\}, \mathcal{N}) \Longrightarrow_{\mathsf{LGC}} (T, \mathcal{N} \cup \mathcal{M}) = (T_p, \mathcal{N}_p),$$

and by ComputeInfer's side condition, $\lfloor \iota' \rfloor = \iota \in \mathit{Red}_{\mathrm{I}}^{\cap}(\lfloor \mathcal{N}_p \rfloor)$, hence $\iota' \in \mathit{Red}_{\mathrm{I}}^{\mathbf{L},\cap}(\mathcal{N}_p) \subseteq \bigcup_i \mathit{Red}_{\mathrm{I}}^{\mathbf{L},\cap}(\mathcal{N}_i)$, as required. □

thm:lgc-completeness

Theorem 67. *Let $(T_i, \mathcal{N}_i)_i$ be a $\Longrightarrow_{\mathsf{LGC}}$-derivation, where $\mathcal{N}_0{\downarrow_{\mathsf{active}}} = \emptyset$, $\mathcal{N}_*{\downarrow_l} = \emptyset$ for all $l \neq \mathsf{active}$, T_0 contains all premise-free inferences of FInf, and $T_* = \emptyset$. If $\lfloor \mathcal{N}_0 \rfloor \models^{\cap} \{\bot\}$ for some $\bot \in \mathbf{F}_\bot$, then some $\mathcal{N}_i$ contains $(\bot', l)$ for some $\bot' \in \mathbf{F}_\bot$ and $l \in \mathbf{L}$.*

Proof. By Lemma 55, $\lfloor \mathcal{N}_0 \rfloor \models^{\cap} \{\bot\}$ is equivalent to $\mathcal{N}_0 \models^{\mathbf{L},\cap} \{(\bot, \mathsf{active})\}$. By Lemma 66, we know that $(\mathcal{N}_i)_i$ is a fair $\rhd_{\mathit{Red}^{\mathbf{L},\cap,\sqsupset}}$-derivation. Since $(\mathit{FLInf}, \mathit{Red}^{\mathbf{L},\cap,\sqsupset})$ is dynamically refutationally complete, we can conclude that some $\mathcal{N}_i$ contains $(\bot', l)$ for some $\bot' \in \mathbf{F}_\bot$ and $l \in \mathbf{L}$. □

Example 68. The following DISCOUNT loop [1] prover DL is an instance of the lazy given clause prover LGC. This loop design is inspired by the description of E [25] but omits E's presimplification of $\mathit{concl}(\iota)$. The prover's state is a four-tuple $T \mid P \mid Y \mid A$, where T is a set of inferences and P, Y, A are sets of formulas. The T, P, and A sets correspond to the scheduled inferences and passive and active formulas, respectively. The Y set is a subsingleton that can store a chosen passive formula. Initial states have the form $T \mid P \mid \emptyset \mid \emptyset$, where T contains all premise-free inferences of *FInf*.

ComputeInfer $T \uplus \{\iota\} \mid P \mid \emptyset \mid A \Longrightarrow_{\mathsf{DL}} T \mid P \mid \{C\} \mid A$
if $\{\mathit{concl}(\iota)\} \cup A \models\!\!\!\approx \{C\}$ and $\iota \in \mathit{Red}_{\mathrm{I}}^{\cap}(A \cup \{C\})$

ChooseP $T \mid P \uplus \{C\} \mid \emptyset \mid A \Longrightarrow_{\mathsf{DL}} T \mid P \mid \{C\} \mid A$

DeleteFwd $T \mid P \mid \{C\} \mid A \Longrightarrow_{\mathsf{DL}} T \mid P \mid \emptyset \mid A$
if $C \in \mathit{Red}_{\mathrm{F}}^{\cap}(A)$ or $C \mathrel{\dot{\succeq}} C'$ for some $C' \in A$

SimplifyFwd $T \mid P \mid \{C\} \mid A \Longrightarrow_{\mathsf{DL}} T \mid P \mid \{C'\} \mid A$
if $\{C\} \cup A \models\!\!\!\approx \{C'\}$ and $C \in \mathit{Red}_{\mathrm{F}}^{\cap}(A \cup \{C'\})$

DeleteBwd $T \mid P \mid \{C\} \mid A \uplus \{C'\} \Longrightarrow_{\mathsf{DL}} T \mid P \mid \{C\} \mid A$
if $C' \in \mathit{Red}_{\mathrm{F}}^{\cap}(\{C\})$ or $C' \succ\!\!- C$

SimplifyBwd $T \mid P \mid \{C\} \mid A \uplus \{C'\} \Longrightarrow_{\mathsf{DL}} T \mid P \cup \{C''\} \mid \{C\} \mid A$
if $\{C, C'\} \mathrel{\mathrel{\approx}\!\!\succ} \{C''\}$ and $C' \in \mathit{Red}_{\mathrm{F}}^{\cap}(\{C, C''\})$

ScheduleInfer $T \mid P \mid \{C\} \mid A \Longrightarrow_{\mathsf{DL}} T \cup T' \mid P \mid \emptyset \mid A \cup \{C\}$
if $T' = \mathit{FInf}(A, \{C\})$

DeleteOrphans $T \uplus T' \mid P \mid Y \mid A \Longrightarrow_{\mathsf{DL}} T \mid P \mid Y \mid A$
if $T' \cap \mathit{FInf}(A) = \emptyset$

A reasonable strategy for applying the DL rules is presented below. It relies on a well-founded ordering $\succ$ on formulas to make sure that the simplification rules actually simplify their target in some sense, preventing infinite looping. It assumes that *FInf* is sound and that $\mathit{FInf}(N, \{C\})$ is finite whenever N is finite.

1. Repeat while $T \cup P \neq \emptyset$ and $\bot \notin Y \cup A$:
   1.1. Apply ComputeInfer or ChooseP to retrieve the next conclusion of an inference from T or the next formula from P, where T and P are organized as a single queue.
   1.2. Apply SimplifyFwd as long as the simplified formula C' is $\succ$-smaller than the original formula C.
   1.3. If DeleteFwd is applicable, apply it.
   1.4. Otherwise:
      1.4.1. Apply DeleteBwd exhaustively.
      1.4.2. Apply SimplifyBwd as long as the simplified formula C'' is $\succ$-smaller than the original formula C'.
      1.4.3. Apply DeleteOrphans.
      1.4.4. Apply ScheduleInfer.

The instantiation of LGC relies on three labels $l_1 \sqsupset \cdots \sqsupset l_3 = \mathsf{active}$ corresponding to the sets P, Y, A, respectively. The T set is ignored when mapping DL states to LGC states.

Example 69. Higher-order unification can give rise to infinitely many incomparable unifiers. As a result, in clausal higher-order superposition [10], performing all inferences between two clauses can lead to infinitely many conclusions, which need to be enumerated fairly. The Zipperposition prover [10], which implements the calculus, performs this enumeration in an extended DISCOUNT loop.

Another instance of infinitary inferences arises in conjunction with the theory of datatypes and codatatypes. Superposition with (co)datatypes [13] includes n-ary Acycl and Uniq rules, which had to be restricted and complemented with axioms so that they could be implemented in Vampire [17]. In Zipperposition, it would have been possible to support the rules in full generality, eliminating the need for the axioms.

Abstractly, a Zipperposition loop prover ZL operates on states $T \mid P \mid Y \mid A$, where T is organized as a finite set of possibly infinite sequences $(\iota_i)_i$ of inferences and the other components are as in DL (Example 68). The ChooseP, DeleteFwd, SimplifyFwd, DeleteBwd, and SimplifyBwd rules are as in DL. The other rules follow:

ComputeInfer $T \uplus \{(\iota_i)_i\} \mid P \mid \emptyset \mid A \Longrightarrow_{\mathsf{ZL}} T \cup \{(\iota_i)_{i\geq 1}\} \mid P \cup \{C\} \mid \emptyset \mid A$
if $\{\mathit{concl}(\iota_0)\} \cup A \models\!\!\approx \{C\}$ and $\iota_0 \in \mathit{Red}_{\mathrm{I}}^{\cap}(A \cup \{C\})$

ScheduleInfer $T \mid P \mid \{C\} \mid A \Longrightarrow_{\mathsf{ZL}} T \cup T' \mid P \mid \emptyset \mid A \cup \{C\}$
if T' is a finite set of sequences $(\iota_i^j)_i$ of inferences such that the set of all ι_i^j equals $\mathit{FInf}(A, \{C\})$

DeleteOrphan $T \uplus \{(\iota_i)_i\} \mid P \mid Y \mid A \Longrightarrow_{\mathsf{ZL}} T \mid P \mid Y \mid A$
if $\iota_i \notin \mathit{FInf}(A)$ for all i

ComputeInfer works on the first element of sequences. ScheduleInfer adds new sequences to T. Typically, these sequences store $\mathit{FInf}(A, \{C\})$, which may be countably infinite, in such a way that all inferences in one sequence have identical premises and can be removed together by DeleteOrphan.

A subtle difference with DL is that ComputeInfer puts the formula C in P instead of Y. This gives more flexibility for scheduling; for example, a prover can pick several formulas from the same sequence and then choose the most suitable one—not necessarily the first one—to move to the active set.

To produce fair derivations, a prover needs to choose the sequence in ComputeInfer fairly and to choose the formula in ChooseP fairly. In combination, this achieves a form of dovetailing. The prover could use a simple algorithm, such as round-robin, for ComputeInfer and employ more sophisticated heuristics for ChooseP.

The implementation in Zipperposition uses a slightly more complicated representation for T, with sequences of subsingletons of inferences. Thus, each sequence element is either a single inference ι or the empty set, which signifies that no new unifier was found up to a certain depth.

4.3 Making Saturation Calculi Fit

The prover architectures described above can be instantiated with saturation calculi that use a redundancy criterion obtained as an intersection of lifted redundancy criteria. Some saturation calculi are defined in such a way that this requirement is trivially satisfied. For other, some reformulation of the redundancy criterion may be necessary.

Example 70. Waldmann [27] considers a superposition calculus modulo Ψ-torsion-free cancellative abelian monoids. Redundant clauses and inferences are defined in the standard way by lifting, except for the Abstraction inference rule: According to Waldmann's definition, a ground instance of an Abstraction inference $\iota = (C_2, C_1, C_0)$ is an Abstraction inference $(C_2\theta, C_1\theta, C_0\theta)$

where $C_2\theta$ and $C_1\theta$ are ground. But the conclusion of an ABSTRACTION inference is never ground, and this applies also to $C_0\theta$. When defining redundancy for ABSTRACTION inferences, it is therefore necessary to further instantiate the abstraction variable y in $C_0\theta$ using a substitution ρ that maps y to a sufficiently small ground term. To obtain a grounding function $\mathcal{G}$ as defined in Sect. 3.1, we need to redefine $\mathcal{G}(\iota)$ as the set of all inferences $(C_2\theta, C_1\theta, C_0\theta\rho)$, rather than the set of all $(C_2\theta, C_1\theta, C_0\theta)$.

Example 71. As explained in Example 50, the refutational completeness of constraint superposition calculi [19, 20] can be shown using intersections $Red^\cap$ of lifted redundancy criteria. The descriptions of these calculi make often use of criteria that imply every Red^R, and therefore $Red^\cap$, but that can be checked more easily. As one can see from the proofs, however, using $Red^\cap$ itself would work as well.

Example 72. The definition of redundancy for Bachmair, Ganzinger, and Waldmann's hierarchic superposition calculus [7] is mostly standard, using a grounding function that maps every clause C to a subset $\mathcal{G}(C)$ of the set of its ground instances and every hierarchic superposition inference ι to a set $\mathcal{G}(\iota)$ of ground standard superposition inferences. There is one exception, namely, CLOSE inferences, which derive $\bot$ from a list of premises that are inconsistent w.r.t. some base (background) theory. For CLOSE inferences, we have $\mathcal{G}(\iota) = \mathit{undef}$.

Baumgartner and Waldmann's variant of the hierarchic superposition calculus [9] relies on a slightly different definition of redundancy: A clause C is redundant if $\mathcal{G}(C) \subseteq \mathit{Red}_\mathrm{F}(\mathcal{G}(N) \cup T) \cup T$; a non-CLOSE inference ι is redundant if $\mathcal{G}(\iota) \subseteq \mathit{Red}_\mathrm{I}(\mathcal{G}(N \cup T))$, where T is a fixed set of ground base clauses and Red is the usual redundancy criterion for ground standard superposition. To convert this into the format required in Sect. 3.1, we can define $\mathit{Red}'_\mathrm{F}(M) := \mathit{Red}_\mathrm{F}(M \cup T) \cup T$, and $\mathit{Red}'_\mathrm{I}(M) := \mathit{Red}_\mathrm{I}(M \cup T)$. It is easy to check that $\mathit{Red}' = (\mathit{Red}'_\mathrm{I}, \mathit{Red}'_\mathrm{F})$ is also a redundancy criterion and that the properties above are equivalent to $\mathcal{G}(C) \subseteq \mathit{Red}'_\mathrm{F}(\mathcal{G}(N))$ and $\mathcal{G}(\iota) \subseteq \mathit{Red}'_\mathrm{I}(\mathcal{G}(N))$. For CLOSE inferences, we have again $\mathcal{G}(\iota) = \mathit{undef}$.

Example 73. For saturation calculi whose refutational completeness proof is based on some kind of lifting of ground instances, the requirement to use a redundancy criterion obtained as an intersection of lifted redundancy criteria is rather natural. The outlier is unfailing completion [2].

Unfailing completion predates the introduction of Bachmair–Ganzinger-style redundancy, but it can be incorporated into that framework by defining that formulas (i.e., rewrite rules and equations) and inferences (i.e., orientation and critical pair computation[6]) are redundant if for every rewrite proof using that rewrite rule, equation, or critical peak, there exists a smaller rewrite proof. The requirement that the redundancy criterion must be obtained by lifting (which

[6] The other inferences of the unfailing completion calculus, such as simplifications of equations or rules, must be considered as simplifications in our framework, rather than as inferences.

is necessary to introduce the labeling) can then be trivially fulfilled by "self-lifting"—i.e., by defining $\mathbf{G} := \mathbf{F}$ and $\mathrel{\not\succ} := \emptyset$ and by taking $\mathcal{G}$ as the function that maps every formula or inference to the set of its α-renamings.

Note that this definition of redundancy differs from the usual definition of redundancy for superposition. For instance, with a term ordering satisfying $\mathsf{f}(\mathsf{b}) \succ \mathsf{f}(\mathsf{c}) \succ \mathsf{f}(\mathsf{d}) \succ \mathsf{b} \succ \mathsf{c} \succ \mathsf{d}$, the equations $\mathsf{b} \approx \mathsf{c}$ and $\mathsf{b} \approx \mathsf{d}$ make $\mathsf{f}(\mathsf{c}) \approx \mathsf{f}(\mathsf{d})$ redundant in the superposition calculus (since they are smaller in the induced clause ordering), but they do not make $\mathsf{f}(\mathsf{c}) \approx \mathsf{f}(\mathsf{d})$ redundant in unfailing completion (since the rewrite proof $\mathsf{f}(\mathsf{c}) \leftrightarrow \mathsf{f}(\mathsf{b}) \leftrightarrow \mathsf{f}(\mathsf{d})$ using $\mathsf{b} \approx \mathsf{c}$ and $\mathsf{b} \approx \mathsf{d}$ is larger than the rewrite proof $\mathsf{f}(\mathsf{c}) \leftrightarrow \mathsf{f}(\mathsf{d})$ using $\mathsf{f}(\mathsf{c}) \approx \mathsf{f}(\mathsf{d})$).

5 Isabelle Development

Most of the framework described in the previous sections has been formalized in Isabelle/HOL [21]. The development is available as part of the IsaFoL (Isabelle Formalization of Logic) [12] repository.[7] IsaFoL is an effort that aims at developing a reusable formal library of results about automated reasoning.

It currently consists of five theory files that build on each other:

- `Consequence_Relations_and_Inferences_Systems.thy` collects basic definitions and lemmas about consequence relations and inference systems.
- `Calculi.thy` contains the definition of inference systems with redundancy criteria together with corresponding results, including the equivalence of static and dynamic refutational completeness.
- `Lifting_to_Non_Ground_Calculi.thy` gathers the results on nonground liftings of calculi without and with well-founded orderings $\sqsupset_D$.
- `Labeled_Lifting_to_Non_Ground_Calculi.thy` contains the labeled extensions of the previous liftings.
- `Prover_Architectures.thy` includes results about the given clause procedure. This file is still under development.

Our development uses Isabelle locales [8] extensively. These are contexts that fix variables and make assumptions about these. Definitions and lemmas occurring inside the locale may then refer to them. With locales, the definitions and lemmas look similar to or even simpler than how they are stated on paper, but the proofs often become more complicated: Layers of locales may hide definitions, and often these need to be manually unfolded in several steps before the desired lemma can be proved. A pathological example is Lemma 58, which obviously holds by construction from a human perspective but whose Isabelle proof required more than a hundred lines of code.

We chose to represent basic nonempty sets such as $\mathbf{F}$, $\mathbf{L}$, and Q by types. This lightened the development in two ways. First, it relieved us from having to thread through nonemptiness conditions. Second, objects are automatically

[7] `https://bitbucket.org/isafol/isafol/src/master/Saturation_Framework`

typed appropriately based on the context, meaning that lemmas could be stated without explicit hypotheses that given objects are formulas, labels, or indices. On the other hand, for sets such as $\mathbf{F}_\bot$ and *FInf* that are subsets of other sets, it was natural to use simply typed sets.

Derivations, which are introduced in `Calculi.thy` to describe the dynamic behavior of a calculus, are represented by the same lazy list codatatype and auxiliary definitions that were used in the mechanization of the ordered resolution prover RP by Schlichtkrull et al. [24]. Their theory files are available in the Archive of Formal Proofs.[8]

The framework's design and its mechanization were carried out largely in parallel. This resulted in more work on the mechanization side because changes had to be propagated, but it also helped detect missing conditions and shape the theory itself. For example, an earlier version of the framework considered only single lifted redundancy criteria instead of intersections of lifted redundancy criteria (Section 3.3). An attempt at verifying RP (Example 64) in Isabelle using the framework made it clear that the theory was not quite general enough to support selection functions (Example 49). In ongoing work, we are completing the RP proof and are developing a verified superposition prover.

6 Conclusion

We presented a formal framework for saturation theorem proving inspired by Bachmair and Ganzinger's *Handbook* chapter [5]. Users can conveniently derive a dynamic refutational completeness result for a concrete prover based on a statically refutationally complete calculus. The key was to strengthen the standard redundancy criterion so that all prover operations, including subsumption deletion, can be justified by inference or redundancy. The framework is mechanized in Isabelle/HOL, where it can be instantiated to verify concrete provers.

The main missing piece of the framework is a generic treatment of clause splitting. The only formal treatment of splitting we are aware of, by Fietzke and Weidenbach [14], hard-codes both the underlying calculus and the strategy for performing splits. Voronkov's AVATAR architecture [26] is more flexible and yields impressive empirical results, but it offers no completeness guarantees.

Acknowledgment. We thank Alexander Bentkamp for discussions about prover architectures for higher-order logic and for feedback from instantiating the framework, notably Theorem 45, Mathias Fleury and Christian Sternagel for their help with the Isabelle development, and Robert Lewis, Visa Nummelin, and Dmitriy Traytel for their comments and suggestions. Blanchette's research has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 713999, Matryoshka). He also benefited from the Netherlands Organization for Scientific Research (NWO) Incidental Financial Support scheme and he has received funding from the NWO under the Vidi program (project No. 016.Vidi.189.037, Lean Forward).

[8] `https://www.isa-afp.org/entries/Ordered_Resolution_Prover.html`

References

[1] Avenhaus, J., Denzinger, J., Fuchs, M.: DISCOUNT: A system for distributed equational deduction. In: Hsiang, J. (ed.) RTA-95. LNCS, vol. 914, pp. 397–402. Springer (1995)

[2] Bachmair, L., Dershowitz, N., Plaisted, D.A.: Completion without failure. In: Aït-Kaci, H., Nivat, M. (eds.) Rewriting Techniques—Resolution of Equations in Algebraic Structures, vol. 2, pp. 1–30. Academic Press (1989)

[3] Bachmair, L., Ganzinger, H.: On restrictions of ordered paramodulation with simplification. In: Stickel, M.E. (ed.) CADE-10. LNCS, vol. 449, pp. 427–441. Springer (1990)

[4] Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. J. Log. Comput. 4(3), 217–247 (1994)

[5] Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. I, pp. 19–99. Elsevier and MIT Press (2001)

[6] Bachmair, L., Ganzinger, H., Waldmann, U.: Superposition with simplification as a desision procedure for the monadic class with equality. In: Gottlob, G., Leitsch, A., Mundici, D. (eds.) KGC '93. LNCS, vol. 713, pp. 83–96. Springer (1993)

[7] Bachmair, L., Ganzinger, H., Waldmann, U.: Refutational theorem proving for hierarchic first-order theories. Appl. Algebra Eng. Commun. Comput. 5, 193–212 (1994)

[8] Ballarin, C.: Locales: A module system for mathematical theories. J. Autom. Reason. 52(2), 123–153 (2014)

[9] Baumgartner, P., Waldmann, U.: Hierarchic superposition revisited. In: Lutz, C., Sattler, U., Tinelli, C., Turhan, A., Wolter, F. (eds.) Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday, Lecture Notes in Computer Science, vol. 11560, pp. 15–56. Springer (2019), https://doi.org/10.1007/978-3-030-22102-7_2

[10] Bentkamp, A., Blanchette, J., Tourret, S., Vukmirović, P., Waldmann, U.: Superposition with lambdas. In: Fontaine, P. (ed.) CADE-27. vol. 11716, pp. 55–73. Springer (2019)

[11] Bentkamp, A., Blanchette, J.C., Cruanes, S., Waldmann, U.: Superposition for lambda-free higher-order logic. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNCS, vol. 10900, pp. 28–46. Springer (2018)

[12] Blanchette, J.C.: Formalizing the metatheory of logical calculi and automatic provers in Isabelle/HOL (invited talk). In: Mahboubi, A., Myreen, M.O. (eds.) CPP 2019. pp. 1–13. ACM (2019)

[13] Blanchette, J.C., Peltier, N., Robillard, S.: Superposition with datatypes and codatatypes. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNCS, vol. 10900, pp. 370–387. Springer (2018)

[14] Fietzke, A., Weidenbach, C.: Labelled splitting. Ann. Math. Artif. Intell. 55(1–2), 3–34 (2009)

[15] Hillenbrand, T., Löchner, B.: The next Waldmeister loop. In: Voronkov, A. (ed.) CADE-18. LNCS, vol. 2392, pp. 486–500. Springer (2002)

[16] Huet, G.P.: A mechanization of type theory. In: Nilsson, N.J. (ed.) IJCAI-73. pp. 139–146. William Kaufmann (1973)

[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 (2013)

[18] McCune, W., Wos, L.: Otter—the CADE-13 competition incarnations. J. Autom. Reason. 18(2), 211–220 (1997)
[19] Nieuwenhuis, R., Rubio, A.: Theorem proving with ordering and equality constrained clauses. J. Symb. Comput. 19(4), 321–351 (1995)
[20] Nieuwenhuis, R., Rubio, A.: Paramodulation-based theorem proving. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. I, pp. 371–443. Elsevier and MIT Press (2001)
[21] Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)
[22] Peltier, N.: A variant of the superposition calculus. Archive of Formal Proofs 2016 (2016), `https://www.isa-afp.org/entries/SuperCalc.shtml`
[23] Schlichtkrull, A., Blanchette, J.C., Traytel, D.: A verified prover based on ordered resolution. In: Mahboubi, A., Myreen, M.O. (eds.) CPP 2019. pp. 152–165. ACM (2019)
[24] Schlichtkrull, A., Blanchette, J.C., Traytel, D., Waldmann, U.: Formalizing Bachmair and Ganzinger's ordered resolution prover. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNCS, vol. 10900, pp. 89–107. Springer (2018)
[25] Schulz, S.: E—a brainiac theorem prover. AI Commun. 15(2–3), 111–126 (2002)
[26] Voronkov, A.: AVATAR: The architecture for first-order theorem provers. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 696–710. Springer (2014)
[27] Waldmann, U.: Cancellative abelian monoids and related structures in refutational theorem proving (part I). J. Symb. Comput. 33(6), 777–829 (2002)
[28] Weidenbach, C.: Combining superposition, sorts and splitting. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. II, pp. 1965–2013. Elsevier and MIT Press (2001)