Journal of Automated Reasoning (2022) 66:499–539
https://doi.org/10.1007/s10817-022-09621-7

A Comprehensive Framework for Saturation Theorem Proving

Uwe Waldmann[1] · **Sophie Tourret**[1,2] · **Simon Robillard**[3] · **Jasmin Blanchette**[1,2,4]

Received: 28 May 2021 / Accepted: 21 March 2022 / Published online: 7 June 2022

Abstract

A crucial operation of saturation theorem provers is deletion of subsumed formulas. Designers of proof calculi, however, usually discuss this only informally, and the rare formal expositions tend to be clumsy. This is because the equivalence of dynamic and static refutational completeness holds only for derivations where all deleted formulas are redundant, but the standard notion of redundancy is too weak: 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 and superposition. The framework modularly extends redundancy criteria derived via a familiar ground-to-nonground lifting. It allows us to extend redundancy criteria so that they cover subsumption, and also to model entire prover architectures so that the static refutational completeness of a calculus immediately implies the dynamic refutational completeness of a prover implementing the calculus within, for instance, an Otter or DISCOUNT loop. Our framework is mechanized in Isabelle/HOL.

Keywords Automated theorem proving · Saturation · Resolution calculus · Superposition calculus · Redundancy · Prover architectures

✉ Uwe Waldmann
uwe@mpi-inf.mpg.de

Sophie Tourret
stourret@mpi-inf.mpg.de; sophie.tourret@inria.fr

Simon Robillard
simon.robillard@umontpellier.fr

Jasmin Blanchette
jblanche@mpi-inf.mpg.de; jasmin.blanchette@inria.fr; j.c.blanchette@vu.nl

1 Max-Planck-Institut für Informatik, Saarland Informatics Campus, Saarbrücken, Germany

2 Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

3 LIRMM, Université de Montpellier, CNRS, Montpellier, France

4 Vrije Universiteit Amsterdam, Amsterdam, the Netherlands

1 Introduction

Saturation is one of the most successful approaches to automatic theorem proving. Provers based on resolution, superposition, and related proof calculi all implement some saturation procedure that systematically derives conclusions of inferences up to a redundancy criterion. More specifically, a saturation prover starts with a problem to refute, typically given as a set of clauses, and draws inferences from the clauses, adding the conclusions to the set. The prover may remove redundant clauses at any point. The refutation attempt ends as soon as the empty clause $\bot$, denoting a contradiction, has been derived. The proof calculus specifies which inferences need to be performed and which clauses are redundant.

In their *Handbook of Automated Reasoning* chapter [6, 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 $\bot$, then N_0 has a model. We will refer to the calculus' refutational completeness as *static* as opposed to a prover's *dynamic* completeness.

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: It will delete a clause C if it is subsumed by another clause C', meaning that $C = C'\sigma \vee D$ for some substitution σ and some clause D. Yet the case where $D = \bot$ 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. Concretely, we would like $\mathsf{P}(x)$ to make $\mathsf{P}(\mathsf{a})$ redundant, but this fails because the instance of $\mathsf{P}(x)$ that entails $\mathsf{P}(\mathsf{a})$, namely $\mathsf{P}(\mathsf{a})$ itself, is not strictly smaller than $\mathsf{P}(\mathsf{a})$. As a result, RP-derivations are *not* $\rhd$-derivations, and Bachmair and Ganzinger's saturation 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. For this reason, 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. 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 that can be refined to obtain an Otter [28] or DISCOUNT [1] loop, and we prove it refutationally complete. We also present a generalization that decouples scheduling and computation of inferences, to support *orphan formula deletion* and *inference dovetailing*. A formula is an orphan when it has lost one of its parents to the redundancy criterion [24, 39]; removing such formulas reduces the search space. Dovetailing makes it possible to support infinitary inference rules, such as λ-superposition and its possibly infinite sequences of higher-order unifiers [14].

When users of the framework instantiate these prover architectures with a concrete saturation calculus, they obtain the dynamic refutational completeness of the combination from the properties of the prover architecture and the static refutational completeness proof for the calculus. The framework is applicable to a wide range of calculi, including ordered resolution [6], unfailing completion [2], standard superposition [5], constraint superposition [29], theory superposition [44], and hierarchic superposition [8], It is already used in several published and ongoing works on combinatory superposition [15], λ-free superposition [12], λ-superposition [13, 14], superposition with interpreted Booleans [33], AVATAR-style splitting [21], and superposition with SAT-inspired inprocessing [43].

When Schlichtkrull, Blanchette, Traytel, and Waldmann [36] mechanized Bachmair and Ganzinger's chapter using the Isabelle/HOL proof assistant [32], they found quite a few mistakes, including one that compromised RP's dynamic refutational completeness [36, Sect. 7.1]. This motivated us to mechanize our framework as well.

This article is structured as follows:

– Section 2 introduces the basic notions of Bachmair–Ganzinger-style saturation theorem proving, which also form the basis of our framework. These include the notions of inferences, redundancy, and static and dynamic refutational completeness.
– Section 3 shows how to lift the refutational completeness of a ground calculus to the nonground level. This step adds support for subsumption and formula labels.
– Section 4 presents several prover architectures that are variants of the given clause procedure, showing how to cope with multiple formula sets.
– Section 5 briefly describes the Isabelle mechanization of the framework.

An earlier version of this work was presented at IJCAR 2020 [45]. This article extends the IJCAR paper with detailed proofs, more explanations and examples, and an index. The Isabelle mechanization is described in more detail in a CPP 2021 paper [41].

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

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 $N_1 \models N_2$ 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: 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). Perhaps unexpectedly, 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). Typically, the consequence relation $\mathrel{|\!\!\approx}$ 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 $\models$ used for (2) implies refutational completeness w.r.t. $\mathrel{|\!\!\approx}$.

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. We write $\mathit{Inf}(N, M) := \mathit{Inf}(N \cup M) \setminus \mathit{Inf}(N \setminus M)$ for the set of all inferences in *Inf* such that one premise is in M (and possibly in N as well) and the other premises are contained in $N \cup M$.

One can find several slightly differing definitions for redundancy criteria, fairness, and saturation in the literature [6, 8, 44]. We discuss the differences in Sect. 2.3. Here we mostly follow Waldmann [44].

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. In a prover, $\mathit{Red}_{\mathrm{I}}$ indicates which inferences need not be performed (e.g., because they have already been performed), whereas $\mathit{Red}_{\mathrm{F}}$ justifies the deletion and simplification of formulas. Intuitively, (R1) states that deleting redundant formulas preserves inconsistency. (R2) and (R3) state that formulas or inferences that are redundant w.r.t. a set N remain redundant if arbitrary formulas are added to N or redundant formulas are deleted from N. (R4) ensures that computing an inference makes it redundant. Note that $C \in \mathit{Red}_{\mathrm{F}}(\{C\})$ generally does not hold.

The connection between redundant inferences and redundant formulas is given by the second part of (R3). Together with (R4) it implies that inferences with a redundant conclusion are themselves redundant:

Lemma 1 *If $\iota \in \mathit{Inf}$ and $\mathit{concl}(\iota) \in \mathit{Red}_F(N)$, then $\iota \in \mathit{Red}_I(N)$.*

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 $\iota \in \mathit{Red}_{\mathrm{I}}(N \cup \mathit{Red}_{\mathrm{F}}(N)) \subseteq \mathit{Red}_{\mathrm{I}}((N \cup \mathit{Red}_{\mathrm{F}}(N)) \setminus (\mathit{Red}_{\mathrm{F}}(N) \setminus N)) = \mathit{Red}_{\mathrm{I}}(N)$. □

Redundant inferences will play a central role in the definition of saturation and refutational completeness in Sect. 2.2.

Example 2 The *trivial redundancy criterion* defined by $\mathit{Red}_{\mathrm{I}}(N) = \{\iota \in \mathit{Inf} \mid \mathit{concl}(\iota) \in N\}$ and $\mathit{Red}_{\mathrm{F}}(N) = \emptyset$ clearly meets requirements (R1)–(R4). According to this criterion, an inference becomes redundant once its conclusion has been added to the set of formulas (e.g., by performing the inference), whereas formulas are never redundant.

Example 3 Assume $\models$ to be compact—i.e., $N \models \{C\}$ implies that $M \models \{C\}$ for some finite $M \subseteq N$.

Given a well-founded ordering $\succ$ on $\mathbf{F}$ and an inference system *Inf* that satisfies $C_1 \succ C_0$ for all $(C_n, \ldots, C_1, C_0) \in \mathit{Inf}$, the *standard redundancy criterion* defined by

$$\mathit{Red}_{\mathrm{I}}(N) := \{(C_n, \ldots, C_1, C_0) \in \mathit{Inf} \mid \{C_n, \ldots, C_2\} \cup \{D \in N \mid C_1 \succ D\} \models \{C_0\}\}$$
$$\mathit{Red}_{\mathrm{F}}(N) := \{C \in \mathbf{F} \mid \{D \in N \mid C \succ D\} \models \{C\}\}$$

also meets requirements (R1)–(R4) [6, Sect. 4.2]. Intuitively, a formula is redundant if it is entailed by $\succ$-smaller formulas in N, whereas an inference is redundant if it is entailed by the side premises $C_n, \ldots, C_2$ together with some formulas $M \subseteq N$ that are $\succ$-smaller than the main premise C_1. This criterion is used in conjunction with ordered resolution and superposition. Some authors require the subsets M to be finite but this is not necessary if $\models$ is compact. *Inf* often satisfies $C_1 \succ C_k$ for all $k > 1$; the definition of $\mathit{Red}_{\mathrm{I}}$ can then be simplified to $\mathit{Red}_{\mathrm{I}}(N) := \{(C_n, \ldots, C_1, C_0) \in \mathit{Inf} \mid \{D \in N \mid C_1 \succ D\} \models \{C_0\}\}$.

Example 4 It is always possible to conflate redundant inferences and formulas by defining $\mathit{Red}_{\mathrm{I}}$ such that $\iota \in \mathit{Red}_{\mathrm{I}}(N)$ if and only if $\iota \in \mathit{Inf}$ and $\mathit{concl}(\iota) \in N \cup \mathit{Red}_{\mathrm{F}}(N)$ for all ι and N. This is, however, needlessly weak for many calculi. Consider superposition, which is based on the standard redundancy criterion. Let $\mathsf{d} > \mathsf{c} > \mathsf{b} > \mathsf{a}$ be the atom ordering, and assume the premises of the superposition inference

$$\frac{\mathsf{d} \approx \mathsf{b} \qquad \mathsf{d} \approx \mathsf{c} \vee \mathsf{d} \approx \mathsf{a}}{\mathsf{b} \approx \mathsf{c} \vee \mathsf{d} \approx \mathsf{a}}$$

are in the clause set. Instead of performing the inference, we would like to exhaustively rewrite the second premise using the first premise, to $\mathsf{b} \approx \mathsf{c} \vee \mathsf{b} \approx \mathsf{a}$, using either demodulation or a simultaneous superposition inference [11]. Notice that the conclusion does not become redundant in this way: It is entailed by $\mathsf{b} \approx \mathsf{c} \vee \mathsf{b} \approx \mathsf{a}$ and $\mathsf{d} \approx \mathsf{b}$, but $\mathsf{d} \approx \mathsf{b}$ is $\succ$-larger than the conclusion and cannot be used. Nevertheless, with the standard redundancy criterion, the superposition inference becomes redundant, since the clause $\mathsf{d} \approx \mathsf{b}$ is smaller than the larger premise of the inference.

2.2 Refutational Completeness

Let $\models$ be a consequence relation, let *Inf* be an inference system, and let *Red* be a redundancy criterion for $\models$ and *Inf*.

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 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 a relation $\rhd \subseteq A \times A$, a $\rhd$-*derivation* is a nonempty A-sequence such that $a_i \rhd a_{i+1}$ for all valid indices.

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 $N \setminus N' \subseteq \mathit{Red}_{\mathrm{F}}(N')$.

In other words, when taking a transition, we may add arbitrary formulas $(N' \setminus N)$ and remove redundant formulas $(N \setminus N')$. In practice, the added formulas would normally be entailed by N. But since our framework is designed to establish only dynamic refutational completeness, we impose no soundness restrictions on added formulas. If dynamic soundness of a prover is desired, it can be derived immediately from the soundness of the inferences and does not require its own framework.

Let $(N_i)_i$ be a $\mathcal{P}(\mathbf{F})$-sequence. Its *limit* (*inferior*) is the set $N_\infty := \bigcup_i \bigcap_{j\geq i} N_j$. The limit consists of the *persistent formulas*: These are formulas that eventually emerge in some N_i and remain present in all the following sets N_j with $j \geq i$. The sequence is called (*weakly*) *fair* if $\mathit{Inf}(N_\infty) \subseteq \bigcup_i \mathit{Red}_{\mathrm{I}}(N_i)$. 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 $\bot' \in \mathbf{F}_\bot$.

Example 5 Let Σ be a signature consisting of a binary predicate P, a unary function f, and a constant a. Let *Inf* be the set of inferences of the ordered resolution calculus with selection on Σ-clauses, and let *Red* be the trivial redundancy criterion (Example 2). This calculus is refutationally complete [6, Sect. 3].

Consider the initial set $N = \{\mathsf{P}(a, a), \neg\mathsf{P}(x, x), \mathsf{P}(\mathsf{f}(x), y) \vee \neg\mathsf{P}(x, y)\}$, where the second literal of the third clause is selected and no other literals are selected. The inference $\iota = (\neg\mathsf{P}(x, x), \mathsf{P}(a, a), \bot)$ belongs to $\mathit{Inf}(N)$ and derives $\bot$, so N is clearly unsatisfiable. However, without fairness, a derivation may never perform this inference. One such example is the infinite derivation

$$N \;\rhd_{\mathit{Red}}\; N \cup \{\mathsf{P}(\mathsf{f}(\mathsf{a}), \mathsf{a})\} \;\rhd_{\mathit{Red}}\; N \cup \{\mathsf{P}(\mathsf{f}(\mathsf{a}), \mathsf{a}), \mathsf{P}(\mathsf{f}(\mathsf{f}(\mathsf{a})), \mathsf{a})\} \;\rhd_{\mathit{Red}}\; \cdots$$

where the clause $\neg\mathsf{P}(x, x)$ is always ignored in favor of inferences between $\mathsf{P}(\mathsf{f}(x), y) \vee \neg\mathsf{P}(x, y)$ and a clause of the form $\mathsf{P}(\mathsf{f}^i(\mathsf{a}), \mathsf{a})$. This derivation is not fair because $\iota \in \mathit{Inf}(N_\infty)$, where $N_\infty = N \cup \{\mathsf{P}(\mathsf{f}^i(\mathsf{a}), \mathsf{a})\}_{i\in\mathbb{N}}$, but $\iota \notin \bigcup_i \mathit{Red}_{\mathrm{I}}(N_i)$. In contrast, any derivation that is similar to the previous one but that performs the inference ι at some step would be fair.

Now consider the satisfiable set $N' = \{\mathsf{P}(a, a), \mathsf{P}(\mathsf{f}(x), y) \vee \neg\mathsf{P}(x, y)\}$. The derivation

$$N' \;\rhd_{\mathit{Red}}\; N' \cup \{\mathsf{P}(\mathsf{f}(\mathsf{a}), \mathsf{a})\} \;\rhd_{\mathit{Red}}\; N' \cup \{\mathsf{P}(\mathsf{f}(\mathsf{a}), \mathsf{a}), \mathsf{P}(\mathsf{f}(\mathsf{f}(\mathsf{a})), \mathsf{a})\} \;\rhd_{\mathit{Red}}\; \cdots$$

is clearly fair.

Using properties (R1)–(R3), it is possible to show that static and dynamic refutational completeness agree [6]:

Lemma 6 *If $(N_i)_i$ is a $\rhd_{\mathit{Red}}$-derivation, then $(\bigcup_i N_i) \setminus N_\infty \subseteq \mathit{Red}_F(\bigcup_i N_i)$.*

Proof If $C \in (\bigcup_i N_i) \setminus N_\infty$, 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), $C \in \mathit{Red}_{\mathrm{F}}(\bigcup_i N_i)$. ◻

Lemma 7 *$(N_i)_i$ is a $\rhd_{Red}$-derivation, then $Red_I(N_i) \subseteq Red_I(N_\infty)$ and $Red_F(N_i) \subseteq Red_F(N_\infty)$ for every i.*

Proof By property (R2), $Red_{\mathrm{I}}(N_i) \subseteq Red_{\mathrm{I}}(\bigcup_i N_i)$; by Lemma 6 and property (R3), $Red_{\mathrm{I}}(\bigcup_i N_i) \subseteq Red_{\mathrm{I}}((\bigcup_i N_i) \setminus ((\bigcup_i N_i) \setminus N_\infty)) = Red_{\mathrm{I}}(N_\infty)$. Analogously, $Red_{\mathrm{F}}(N_i) \subseteq Red_{\mathrm{F}}(\bigcup_i N_i) \subseteq Red_{\mathrm{F}}((\bigcup_i N_i) \setminus ((\bigcup_i N_i) \setminus N_\infty)) = Red_{\mathrm{F}}(N_\infty)$. □

Lemma 8 *If $(N_i)_i$ is a $\rhd_{Red}$-derivation, then $N_i \subseteq N_\infty \cup Red_F(N_\infty)$ for every i.*

Proof Let $C \in N_i$. If $C \notin N_\infty$, then there exists some $j \geq i$ such that $C \in N_j \setminus N_{j+1}$. Consequently, $C \in Red_{\mathrm{F}}(N_{j+1})$ and therefore $C \in Red_{\mathrm{F}}(N_\infty)$ by Lemma 7. □

Lemma 9 *If $(N_i)_i$ is a fair $\rhd_{Red}$-derivation, then the limit N_∞ is saturated w.r.t. Inf and Red.*

Proof By fairness, every $\iota \in Inf(N_\infty)$ is contained in $\bigcup_i Red_{\mathrm{I}}(N_i)$, so there exists some i such that $\iota \in Red_{\mathrm{I}}(N_i)$, and thus $\iota \in Red_{\mathrm{I}}(N_\infty)$ by Lemma 7. □

Since fairness implies saturation of the limit, we could imagine simplifying the theory by defining dynamic refutational completeness in terms of derivations with saturated limits, instead of in terms of fair derivations. However, fairness more closely captures how provers work and is therefore easier to establish. A prover may delete formulas at different moments, without knowing the limit. This intuition is captured by the right-hand side $\bigcup_i Red_{\mathrm{I}}(N_i)$ of the definition of fairness.

Lemma 10 *If (Inf, Red) is statically refutationally complete w.r.t. $\models$, then it is dynamically refutationally complete w.r.t. $\models$.*

Proof Assume (Inf, Red) is statically refutationally complete w.r.t. $\models$, and let $(N_i)_i$ be a $\rhd_{Red}$-derivation. Assume that $N_0 \models \{\bot\}$ for some $\bot \in \mathbf{F}_\bot$. Since $N_0 \subseteq \bigcup_i N_i$, we get $\bigcup_i N_i \models N_0 \models \{\bot\}$, and by property (R1), this implies $(\bigcup_i N_i) \setminus Red_{\mathrm{F}}(\bigcup_i N_i) \models \{\bot\}$. By Lemma 6, we know that $(\bigcup_i N_i) \setminus N_\infty \subseteq Red_{\mathrm{F}}(\bigcup_i N_i)$, or equivalently, $(\bigcup_i N_i) \setminus Red_{\mathrm{F}}(\bigcup_i N_i) \subseteq N_\infty$; hence $N_\infty \models (\bigcup_i N_i) \setminus Red_{\mathrm{F}}(\bigcup_i N_i) \models \{\bot\}$.

If the sequence is fair, then N_∞ is saturated, so by static refutational completeness, $\bot' \in N_\infty$ 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:

Lemma 11 *If (Inf, Red) is dynamically refutationally complete w.r.t. $\models$, then it is statically refutationally complete w.r.t. $\models$.*

Proof Assume (Inf, 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 $N_\infty = N_0$ and N_0 is saturated, we know that $Inf(N_\infty) = Inf(N_0) \subseteq Red_{\mathrm{I}}(N_0) = \bigcup_i Red_{\mathrm{I}}(N_i)$, so the sequence is fair. By dynamic refutational completeness, this implies $\bot' \in N_0$ for some $\bot' \in \mathbf{F}_\bot$. Therefore (Inf, Red) is statically refutationally complete. □

Example 12 Some popular preprocessing techniques, such as pure literal elimination and blocked clause elimination [26], are not covered by redundancy. These require inspecting the entire formula set and are not compatible with (R2). For example, P is pure in $\{\mathsf{P}\}$ but not in the larger set $\{\mathsf{P}, \neg\mathsf{P}\}$.

In some situations, we may wish to apply techniques that delete clauses even if they are not redundant. If such techniques are used at most finitely many times, we can incorporate them in the framework by letting the initial set N_0 correspond to the set of formulas *after* all inprocessing has been performed, effectively considering all the transitions that happened before N_0 as one big preprocessing step. For example, a first-order prover might discover a clause that contains a pure predicate symbol (a predicate symbol that always occurs with the same polarity in the clause set) and delete it. If the signature is finite, this can be done only finitely many times and is hence compatible with the framework. We only need to show that inprocessing preserves unsatisfiability.

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.

2.3.1 Redundancy Criteria

As in Bachmair and Ganzinger's chapter [6, 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., $N \setminus Red_{\mathrm{F}}(N) \models Red_{\mathrm{F}}(N)$—leading to some obvious changes in Lemmas 10 and 37.

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 $Red_{\mathrm{I}}(N)$ whenever $\mathit{concl}(\iota) \in N \cup Red_{\mathrm{F}}(N)$. As demonstrated by Lemma 1, that condition is equivalent to our condition (R4).

2.3.2 Inferences from Redundant Premises

Inferences from redundant premises are sometimes excluded in the definitions of saturation, fairness, and refutational completeness [6], and sometimes not [5, 10, 30, 44].[1] Similarly, redundancy of inferences is sometimes defined in such a way that inferences from redundant premises are necessarily redundant themselves [5, 10], and sometimes not [6, 30, 44]. There are good arguments for each of these choices. 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, and that inferences from redundant formulas should be redundant as well. 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?

To clarify how the different definitions found in the literature relate to each other, 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 $\mathit{Inf}(N \setminus Red_{\mathrm{F}}(N)) \subseteq Red_{\mathrm{I}}(N)$. 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 $(N_i)_i$ is called *reducedly fair* if $\mathit{Inf}(N_\infty \setminus \bigcup_i Red_{\mathrm{F}}(N_i)) \subseteq \bigcup_i Red_{\mathrm{I}}(N_i)$. The pair

[1] Note that Bachmair and Ganzinger's *J. Log. Comput.* article [5] uses a terminology that differs from most later publications in this area: Their "composite" corresponds to "redundant," and their "complete" corresponds to "saturated."

$(\mathit{Inf}, \mathit{Red})$ 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 $\bot' \in \mathbf{F}_\bot$. A *reduced redundancy criterion* for $\models$ and *Inf* 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 $N \subseteq \mathbf{F}$. Recall that $\mathit{Inf}(N, M)$ denotes the set of *Inf*-inferences with at least one premise in M and the others in $N \cup M$.

For reduced redundancy criteria, saturation and reduced saturation agree:

Lemma 13 *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 $\iota \in \mathit{Red}_{\mathrm{I}}(N)$. □

Corollary 14 *If Red is a reduced redundancy criterion, then* $(\mathit{Inf}, \mathit{Red})$ *is statically refutationally complete if and only if it is reducedly statically refutationally complete.*

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

Lemma 15 *Red' is a reduced redundancy criterion.*

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

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

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

Lemma 16 *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 $\mathit{Red}_{\mathrm{I}}(N) \subseteq \mathit{Red}'_{\mathrm{I}}(N)$, $\mathit{Inf}(N) \subseteq \mathit{Red}_{\mathrm{I}}(N)$ implies $\mathit{Inf}(N) \subseteq \mathit{Red}'_{\mathrm{I}}(N)$. □

The converse does not hold:

Example 17 Consider a signature consisting of the four propositional variables (or nullary predicate symbols) $\mathsf{P}, \mathsf{Q}, \mathsf{R}, \mathsf{S}$. Let *Inf* be the set of inferences of the ordered resolution calculus with selection over clauses over the signature. Define $\mathit{Red}_{\mathrm{F}}$ such that a clause C is contained in $\mathit{Red}_{\mathrm{F}}(N)$ if it is entailed by clauses in N that are smaller than C. Define $\mathit{Red}_{\mathrm{I}}$ such that an inference is contained in $\mathit{Red}_{\mathrm{I}}(N)$ if its conclusion is entailed by clauses in N that are smaller than its largest premise. Then $\mathit{Red} := (\mathit{Red}_{\mathrm{I}}, \mathit{Red}_{\mathrm{F}})$ 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.

Lemma 18 *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) *$N \setminus Red_{\mathrm{F}}(N)$ 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., $Inf(N \setminus Red_{\mathrm{F}}(N)) \subseteq Red_{\mathrm{I}}(N)$. We must show that $Inf(N) \subseteq Red'_{\mathrm{I}}(N)$. Let $\iota \in Inf(N)$. If no premise of ι is contained in $Red_{\mathrm{F}}(N)$, then $\iota \in Inf(N \setminus Red_{\mathrm{F}}(N)) \subseteq Red_{\mathrm{I}}(N)$. Otherwise, $\iota \in Inf(\mathbf{F}, Red_{\mathrm{F}}(N))$. In both cases, we conclude $\iota \in Red'_{\mathrm{I}}(N)$.

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

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

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

Theorem 19 *The following properties are equivalent:*

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

Proof To show that (iii) implies (i), assume that (Inf, 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 16, 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 (Inf, 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 18, $N \setminus Red_{\mathrm{F}}(N)$ is saturated w.r.t. Inf and Red. Furthermore, by (R1), $N \setminus Red_{\mathrm{F}}(N) \models \{\bot\}$. So the static refutational completeness of (Inf, Red) implies that $\bot' \in N \setminus Red_{\mathrm{F}}(N)$ for some $\bot' \in \mathbf{F}_\bot$; hence $\bot' \in N$. Thus, (Inf, Red') is statically refutationally complete.

The equivalence of (iii) and (iv) follows from Lemma 15 and Corollary 14.

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 18, 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 9:

Lemma 20 *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 $Red_{\mathrm{F}}(N_i) \subseteq Red_{\mathrm{F}}(N_\infty)$ for every i, we have $Inf(N_\infty \setminus Red_{\mathrm{F}}(N_\infty)) \subseteq Inf(N_\infty \setminus \bigcup_i Red_{\mathrm{F}}(N_i))$. By reduced fairness, every inference $\iota \in Inf(N_\infty \setminus Red_{\mathrm{F}}(N_\infty))$ is contained in $\bigcup_i Red_{\mathrm{I}}(N_i)$. Therefore there exists some i with $\iota \in Red_{\mathrm{I}}(N_i)$, implying $\iota \in Red_{\mathrm{I}}(N_\infty)$. $\square$

Lemmas 10 and 11 can then be reproved for reduced static and reduced dynamic refutational completeness. Together with Theorem 19, we obtain this result:

Theorem 21 *The properties* (i)–(iv) *of Theorem* 19 *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 21 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 45 in Sect. 3.

2.3.3 Fairness in the Limit

Bachmair and Ganzinger consider a sequence $(N_i)_i$ fair if $concl(Inf(N') \setminus Red_{\mathrm{I}}(N')) \subseteq (\bigcup_i N_i) \cup Red_{\mathrm{F}}(\bigcup_i N_i)$, where $N' = N_\infty \setminus Red_{\mathrm{F}}(N_\infty)$ [6, 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 $(\bigcup_i N_i) \cup Red_{\mathrm{F}}(\bigcup_i N_i)$, then $\iota \in Red_{\mathrm{I}}(\bigcup_i N_i)$ by Lemma 1 and (R4), and by Lemma 6 and (R3) we have $\iota \in Red_{\mathrm{I}}(\bigcup_i N_i) \subseteq Red_{\mathrm{I}}((\bigcup_i N_i) \setminus ((\bigcup_i N_i) \setminus N_\infty)) = Red_{\mathrm{I}}(N_\infty) \subseteq Red_{\mathrm{I}}(N_\infty \setminus Red_{\mathrm{F}}(N_\infty)) = Red_{\mathrm{I}}(N')$. But this contradicts the assumption that $\iota \in Inf(N') \setminus Red_{\mathrm{I}}(N')$. So the condition can be simplified to $Inf(N') \subseteq Red_{\mathrm{I}}(N')$, and since $Red_{\mathrm{I}}(N') = Red_{\mathrm{I}}(N_\infty \setminus Red_{\mathrm{F}}(N_\infty)) = Red_{\mathrm{I}}(N_\infty)$, this is equivalent to $Inf(N_\infty \setminus Red_{\mathrm{F}}(N_\infty)) \subseteq Red_{\mathrm{I}}(N_\infty)$.

[2] The limit need not be saturated, though. In Example 17, the one-element sequence $(N_i)_{i=0}^{0}$ with $N_0 = N$ is reducedly fair w.r.t. *Red*, and its limit $N_\infty = N$ is reducedly saturated w.r.t. *Red*, but not saturated w.r.t. *Red*.

Since $Inf(N_\infty \setminus Red_{\mathrm{F}}(N_\infty)) \subseteq Inf(N_\infty \setminus \bigcup_i Red_{\mathrm{F}}(N_i))$ and $\bigcup_i Red_{\mathrm{I}}(N_i) \subseteq Red_{\mathrm{I}}(N_\infty)$, 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 22 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 a lexicographic path ordering 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 [6, 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 generally cannot be checked effectively. In other words, Bachmair and Ganzinger's definition is more permissive than our alternative definition, but the additional degree of freedom can hardly be exploited in a theorem prover.

2.3.4 Semi-redundancy

Bachmair, Ganzinger, and Waldmann [8] use a definition of redundancy criteria that requires (R2) only for formulas and (R3) only for inferences. With their definition of fairness, this is sufficient to show that the limit of a fair $\rhd_{Red}$-derivation is saturated, and thus, to show that static refutational completeness implies dynamic refutational completeness. Their definition of fairness, however, requires essentially that inferences from formulas in the limit N_∞ are redundant w.r.t. the limit, and since they do not enforce that an inference that is redundant at some step of the derivation is redundant w.r.t. the limit, this cannot be checked effectively in a theorem prover.

2.3.5 Nonstrict Redundancy

Nieuwenhuis and Rubio [29, 30] and Peltier [34] define a ground clause C to be nonstrictly redundant w.r.t. a set N of ground clauses if C is entailed by 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 descriptions of concrete saturation calculi, we frequently encounter the situation that the calculus is parameterized in some way and that exactly one value of the parameter is used to show that every saturated $N \subseteq \mathbf{F}$ with $N \models \{\bot\}$ contains $\bot$, but that this value is still unknown during the actual saturation process. Consequently, inferences and formulas may be considered as redundant during the saturation only if they are redundant for every possible value of the parameter. To model this situation in our framework, it is useful to define consequence relations and redundancy criteria as intersections of previously defined consequence relations or redundancy criteria.

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

Lemma 23 $\models^\cap$ *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 $\mathit{Red}^\cap_{\mathrm{I}}(N) := \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{I}}(N)$ and $\mathit{Red}^\cap_{\mathrm{F}}(N) := \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{F}}(N)$ for all N. Define $\mathit{Red}^\cap := (\mathit{Red}^\cap_{\mathrm{I}}, \mathit{Red}^\cap_{\mathrm{F}})$.

Lemma 24 *$\mathit{Red}^\cap$ is a redundancy criterion for $\models^\cap$ and Inf.*

Proof (R1) Assume that $N \models^\cap_G \{\bot\}$ for some $\bot \in \mathbf{F}_\bot$—i.e., $N \models^q \{\bot\}$ for every $q \in Q$. As $\mathit{Red}^\cap_{\mathrm{F}}(N) \subseteq \mathit{Red}^q_{\mathrm{F}}(N)$, we have $N \setminus \mathit{Red}^\cap_{\mathrm{F}}(N) \supseteq N \setminus \mathit{Red}^q_{\mathrm{F}}(N)$, and by (C2) $N \setminus \mathit{Red}^\cap_{\mathrm{F}}(N) \models^q N \setminus \mathit{Red}^q_{\mathrm{F}}(N)$. Furthermore, $N \setminus \mathit{Red}^q_{\mathrm{F}}(N) \models^q \{\bot\}$ by (R1) for Red^q. So $N \setminus \mathit{Red}^\cap_{\mathrm{F}}(N) \models^q \{\bot\}$ by (C4) for every $q \in Q$ and therefore $N \setminus \mathit{Red}_{\mathrm{F}}(N) \models^\cap_G \{\bot\}$.
(R2) Let $N \subseteq N'$. Since $\mathit{Red}^q_{\mathrm{F}}(N) \subseteq \mathit{Red}^q_{\mathrm{F}}(N')$ for every q, we have $\mathit{Red}^\cap_{\mathrm{F}}(N) = \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{F}}(N) \subseteq \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{F}}(N') = \mathit{Red}^\cap_{\mathrm{F}}(N')$ and analogously for $\mathit{Red}^\cap_{\mathrm{I}}$.
(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 $\mathit{Red}^\cap_{\mathrm{F}}(N) = \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{F}}(N) \subseteq \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{F}}(N \setminus N') = \mathit{Red}^\cap_{\mathrm{F}}(N \setminus N')$ and analogously for $\mathit{Red}^\cap_{\mathrm{I}}$.
(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 $\iota \in \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{I}}(N) = \mathit{Red}^\cap_{\mathrm{I}}(N)$. □

Lemma 25 *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 $\mathit{Inf}(N) \subseteq \mathit{Red}^\cap_{\mathrm{I}}(N) = \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{I}}(N)$; 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 $\mathit{Inf}(N) \subseteq \mathit{Red}^\cap_{\mathrm{I}}(N) = \bigcap_{q\in Q} \mathit{Red}^q_{\mathrm{I}}(N)$, 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 $q \in Q$.

There are some exceptions, though, for example constraint superposition [29], where the parameter q is a convergent rewrite system R and $\models^q$ is entailment modulo R. For such calculi, one can typically demonstrate the static refutational completeness of $(\mathit{Inf}, \mathit{Red}^\cap)$ in the following form:

Lemma 26 *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 nonground 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 set 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 PosExt inferences of λ-free superposition [12]. 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 $\mathcal{G}(N) := \bigcup_{C \in N} \mathcal{G}(C)$ for all N. Analogously, for a set $I \subseteq \mathit{FInf}$, $\mathcal{G}(I) := \bigcup_{\iota \in I,\, \mathcal{G}(\iota) \neq \mathit{undef}} \mathcal{G}(\iota)$.

Remark 27 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 (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 28 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 $\models$. It corresponds to Herbrand entailment. If Tarski entailment (i.e., $N_1 \models_{\mathrm{T}} N_2$ if and only if any model of N_1 is also a model of N_2) is desired, the mismatch can be repaired by showing that the two notions of entailment are equivalent as far as refutations are concerned.

Lemma 29 *$\models_{\mathcal{G}}$ is a consequence relation over* $\mathbf{F}$.

Proof (C1) Let $\bot \in \mathbf{F}_\bot$. Then by property (G1) of grounding functions, $\mathcal{G}(\{\bot\})$ 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 $\models$ and *GInf*. 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

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

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

Theorem 30 *$Red^{\mathcal{G}}$ is a redundancy criterion for $\models_{\mathcal{G}}$ and FInf.*

We omit the proof at this point since we will prove a more general result (Theorem 42) in Sect. 3.2. The following folklore lemma connects a nonground calculus with a ground calculus it overapproximates.

Lemma 31 *If $N \subseteq \mathbf{F}$ is saturated w.r.t. FInf and $Red^{\mathcal{G}}$ and $GInf(\mathcal{G}(N)) \subseteq \mathcal{G}(FInf(N)) \cup Red_I(\mathcal{G}(N))$, 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 $Red_{\mathrm{I}}(\mathcal{G}(N))$. 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 $\mathcal{G}(\iota) \neq undef$, 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 is 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.

Theorem 32 *If $(GInf, Red)$ is statically refutationally complete w.r.t. $\models$, and if we have $GInf(\mathcal{G}(N)) \subseteq \mathcal{G}(FInf(N)) \cup Red_I(\mathcal{G}(N))$ for every $N \subseteq \mathbf{F}$ that is saturated w.r.t. FInf and $Red^{\mathcal{G}}$, then $(FInf, Red^{\mathcal{G}})$ 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 $\mathcal{G}(N) \models \mathcal{G}(\bot)$. By property (G1) of grounding functions, $\mathcal{G}(\bot)$ 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 $C \in \mathbf{F}_\bot$ by property (G2) of grounding functions. Now define $\bot' := C$. □

Example 33 In ordered binary resolution without selection [6, 35], 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 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 $(C_n, \ldots, C_0) \in \mathit{FInf}$ to the set of all $(C_n\theta, \ldots, C_0\theta) \in \mathit{GInf}$. Then every resolution inference in *GInf* from ground instances of clauses in N has the form

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

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

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

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

Therefore, the static refutational completeness of *GInf* implies the static refutational completeness of *FInf*.

The liftability result above holds also for ordered binary resolution with selection, provided that the selection function *fsel* on $\mathbf{F}$ and the selection function *gsel* on $\mathbf{G}$ are such that every clause $D \in \mathcal{G}(N)$ inherits the selection of at least one clause $C \in N$ for which $D \in \mathcal{G}(C)$. One can show that for every $N \subseteq \mathbf{G}$ and *fsel*, such a *gsel* exists. However, this *gsel* depends on N, and therefore Theorem 32 is not applicable. We will discuss this issue further in Sect. 3.3.

Example 34 In the superposition calculus without selection [5], 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 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 $(C_n, \ldots, C_0) \in \mathit{FInf}$ to the set of all $(C_n\theta, \ldots, C_0\theta) \in \mathit{GInf}$. Then every equality resolution or equality factoring inference from ground instances of clauses in N is contained in $\mathcal{G}(\iota)$ for some inference $\iota \in \mathit{FInf}(N)$. The same applies to superposition inferences

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

with $s\theta|_p = t\theta$, provided that p is a position of s and $s|_p$ is not a variable. Otherwise, $p = p_1 p_2$ for some variable x occurring in s at the position p_1, so $x\theta|_{p_2} = t\theta$. In this case, define θ' such that $x\theta' = x\theta[t'\theta]_{p_2}$ 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.

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

3.2 Adding Tiebreaker Orderings

We now strengthen the $\mathcal{G}$-lifting of redundancy criteria introduced in the previous subsection to also support subsumption deletion. Let $\sqsupset = (\sqsupset_D)_{D\in\mathbf{G}}$ be a $\mathbf{G}$-indexed family of strict partial orderings on $\mathbf{F}$ that are well founded (i.e., for every D, $\sqsupset_D$ there exists no infinite descending chain $C_0 \sqsupset_D C_1 \sqsupset_D \cdots$). We define $Red_{\mathrm{F}}^{\mathcal{G},\sqsupset} : \mathcal{P}(\mathbf{F}) \to \mathcal{P}(\mathbf{F})$ as follows:

$$
\begin{aligned}
&C \in Red_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N) \ \text{ if and only if}\\
&\quad \text{for every } D \in \mathcal{G}(C),\\
&\qquad D \in Red_{\mathrm{F}}(\mathcal{G}(N)) \ \text{ or there exists } C' \in N \ \text{ such that } C \sqsupset_D C' \text{ and } D \in \mathcal{G}(C').
\end{aligned}
$$

Notice how $\sqsupset_D$ is used to break ties between C and C', possibly making C redundant. We call $Red^{\mathcal{G},\sqsupset} := (Red_{\mathrm{I}}^{\mathcal{G}}, Red_{\mathrm{F}}^{\mathcal{G},\sqsupset})$ the $(\mathcal{G}, \sqsupset)$*-lifting* of *Red*.

For nearly all applications (with a notable exception in Example 49 below), 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. We get the previously defined $Red^{\mathcal{G}} = (Red_{\mathrm{I}}^{\mathcal{G}}, Red_{\mathrm{F}}^{\mathcal{G}})$ as a special case of $Red^{\mathcal{G},\sqsupset} = (Red_{\mathrm{I}}^{\mathcal{G}}, Red_{\mathrm{F}}^{\mathcal{G},\sqsupset})$ 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 $Red_{\mathrm{F}}^{\mathcal{G},\sqsupset}$ is contained in $N \setminus Red_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$:

Lemma 35 *$C \in Red_F^{\mathcal{G},\sqsupset}(N)$ if and only if for every $D \in \mathcal{G}(C)$ we have $D \in Red_F(\mathcal{G}(N))$ or there exists $C' \in N \setminus Red_F^{\mathcal{G},\sqsupset}(N)$ 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 $C \in Red_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$ and $D \in \mathcal{G}(C)$. By definition, $D \in Red_{\mathrm{F}}(\mathcal{G}(N))$ or there exists $C' \in N$ such that $C \sqsupset_D C'$ and $D \in \mathcal{G}(C')$. If $D \in Red_{\mathrm{F}}(\mathcal{G}(N))$, we are done. Let $D \notin Red_{\mathrm{F}}(\mathcal{G}(N))$. 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 $Red_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$. Since $D \notin Red_{\mathrm{F}}(\mathcal{G}(N))$, 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 $C' \in N \setminus Red_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$. □

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

Lemma 36 $\mathcal{G}(N) \setminus Red_F(\mathcal{G}(N)) \subseteq \mathcal{G}(N \setminus Red_F^{\mathcal{G},\sqsupset}(N))$.

Proof Let $D \in \mathcal{G}(N) \setminus Red_{\mathrm{F}}(\mathcal{G}(N))$. 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 $C \in Red_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$. Then, by definition, $D \in Red_{\mathrm{F}}(\mathcal{G}(N))$ 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:

Lemma 37 *If $N \models_{\mathcal{G}} \{\bot\}$ for some $\bot \in \mathbf{F}_\bot$, then $N \setminus \mathit{Red}_{F}^{\mathcal{G},\sqsupset}(N) \models_{\mathcal{G}} \{\bot\}$.*

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 $\bot' \in \mathbf{G}_\bot$ by property (G1) of grounding functions, we have $\mathcal{G}(N) \models \mathcal{G}(\{\bot\}) \models \{\bot'\}$. By property (R1) of redundancy criteria, this implies $\mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N)) \models \{\bot'\}$. Furthermore, by Lemma 36, $\mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N)) \subseteq \mathcal{G}(N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N))$, 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 the two relations, we obtain $\mathcal{G}(N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)) \models \{\bot'\} \models \mathcal{G}(\{\bot\})$. By definition of $\models_{\mathcal{G}}$ and property (G1) of grounding functions, this means $N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N) \models_{\mathcal{G}} \{\bot\}$, as required. □

Lemma 38 *If $N \subseteq N'$, then $\mathit{Red}_{F}^{\mathcal{G},\sqsupset}(N) \subseteq \mathit{Red}_{F}^{\mathcal{G},\sqsupset}(N')$ and $\mathit{Red}_{I}^{\mathcal{G}}(N) \subseteq \mathit{Red}_{I}^{\mathcal{G}}(N')$.*

Proof Obvious. □

Lemma 39 *If $N' \subseteq \mathit{Red}_{F}^{\mathcal{G},\sqsupset}(N)$, then $\mathit{Red}_{F}^{\mathcal{G},\sqsupset}(N) \subseteq \mathit{Red}_{F}^{\mathcal{G},\sqsupset}(N \setminus N')$.*

Proof Let $N' \subseteq \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$, let $C \in \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)$. Then, by Lemma 35, 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 $C \in \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N \setminus N')$. □

Lemma 40 *If $N' \subseteq \mathit{Red}_{F}^{\mathcal{G},\sqsupset}(N)$, then $\mathit{Red}_{I}^{\mathcal{G}}(N) \subseteq \mathit{Red}_{I}^{\mathcal{G}}(N \setminus N')$.*

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 $\mathcal{G}(N) \setminus \mathit{Red}_{\mathrm{F}}(\mathcal{G}(N)) \subseteq \mathcal{G}(N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)) \subseteq \mathcal{G}(N \setminus N')$ by Lemma 36, this implies $\iota' \in \mathit{Red}_{\mathrm{I}}(\mathcal{G}(N \setminus N'))$ by (R2). 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 36, $D \in \mathcal{G}(N \setminus \mathit{Red}_{\mathrm{F}}^{\mathcal{G},\sqsupset}(N)) \subseteq \mathcal{G}(N \setminus N')$. 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 $\iota \in \mathit{Red}_{\mathrm{I}}^{\mathcal{G}}(N \setminus N')$. □

Lemma 41 *If $\iota \in FInf$ and $concl(\iota) \in N$, then $\iota \in Red_I^{\mathcal{G}}(N)$.*

Proof Let $\iota \in FInf$ such that $concl(\iota) \in N$. If $\mathcal{G}(\iota) \neq undef$, then by property (G3) of grounding functions, $\mathcal{G}(\iota)$ is a subset of $Red_{\mathrm{I}}(\mathcal{G}(concl(\iota)))$, which in turn is a subset of $Red_{\mathrm{I}}(\mathcal{G}(N))$. So $\iota \in Red_{\mathrm{I}}^{\mathcal{G}}(N)$.

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

By combining Lemmas 37– 41, we obtain our first main result, generalizing Theorem 30:

Theorem 42 *Let Red be a redundancy criterion for $\models$ and GInf, 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 $Red^{\mathcal{G},\sqsupset}$ of Red is a redundancy criterion for $\models_{\mathcal{G}}$ and FInf.*

Observe that $\sqsupset$ appears only in the second component of $Red^{\mathcal{G},\sqsupset} = (Red_{\mathrm{I}}^{\mathcal{G}}, Red_{\mathrm{F}}^{\mathcal{G},\sqsupset})$ 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:

Lemma 43 *A set $N \subseteq \mathbf{F}$ is saturated w.r.t. FInf and $Red^{\mathcal{G},\sqsupset}$ if and only if it is saturated w.r.t. FInf and $Red^{\mathcal{G},\emptyset}$.*

Lemma 44 *$(FInf, Red^{\mathcal{G},\sqsupset})$ 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 10 and 44, we obtain our second main result:

Theorem 45 *Let Red be a redundancy criterion for $\models$ and GInf, 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 46 For resolution or superposition in standard first-order logic, we can define the *instantiation* quasi-ordering $\gtrdot\!\!\!\!\raisebox{-0.5ex}{\_}$ on clauses by $C \succeq\!\!\!\cdot\; C'$ if and only if $C = C'\sigma$ for some substitution σ. In particular, if C and C' are α-renamings of each other, then $C \succeq\!\!\!\cdot\; C'$ and $C \preceq\!\!\!\cdot\; C'$.

The instantiation ordering $\gtrdot := \succeq\!\!\!\cdot \setminus \preceq\!\!\!\cdot$ is well founded. By choosing $\sqsupset := \gtrdot$, we obtain a criterion $Red^{\mathcal{G},\sqsupset}$ that includes standard redundancy (Example 3) and also supports subsumption deletion. (It is customary to define subsumption so that C is subsumed by C' if $C = C'\sigma \vee D$ for some substitution σ and some possibly empty clause D, but since the case where D is nonempty is already supported by the standard redundancy criterion, the instantiation ordering $\gtrdot$ is sufficient.)

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

Example 47 For higher-order calculi such as higher-order resolution [25] and λ-superposition [14], the instantiation ordering is not well founded, as witnessed by the chain

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

Example 48 In constraint superposition with ordering constraints [29], a ground instance of a constrained clause $C\,[\![K]\!]$ is defined as a ground clause $C\theta$ for which the constraint $K\theta$ evaluates to true. One can then define the quasi-ordering $\gtrdot\!\!\!=$ by stating that $C\,[\![K]\!] \mathrel{\underline{\gtrdot}} C'\,[\![K']\!]$ if and only if every ground instance of $C\,[\![K]\!]$ is a ground instance of $C'\,[\![K']\!]$. Again, the ordering $\gtrdot := \mathrel{\underline{\gtrdot}} \setminus \mathrel{\underline{\lessdot}}$. is not well founded, since

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

is an infinite chain if $\succ$ is a simplification ordering.

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

$$\begin{aligned}
&C \sqsupset C' \text{ if and only if}\\
&\quad C \mathrel{\underline{\gtrdot}} C'\\
&\quad \text{and } \big(\text{size}(C) > \text{size}(C')\\
&\qquad\quad \text{or } \big(\text{size}(C) = \text{size}(C')\\
&\qquad\qquad \text{and } C \text{ contains fewer distinct variables than } C'\big)\big).
\end{aligned}$$

Conversely, the $\sqsupset$ relation can be more general than subsumption. In Sect. 4, we will use it to justify the movement of formulas between sets in the given clause procedure.

Example 49 There are a few applications, notably for superposition-based decision procedures [7], where one would like to define $Red_{\mathrm{F}}^{\mathcal{G},\sqsupset}$ using the reverse instantiation ordering $\lessdot$.

In this way, a clause $\mathsf{P}(x)$ would for example become redundant in the presence of the clauses $\mathsf{P}(\mathsf{b})$ and $\mathsf{P}(\mathsf{c})$, provided that b and c are the only two ground terms. The reverse instantiation ordering is not well founded on the set of all first-order clauses: $\mathsf{P}(x) \lessdot \mathsf{P}(\mathsf{f}(x)) \lessdot \mathsf{P}(\mathsf{f}(\mathsf{f}(x))) \lessdot \cdots$. However, it is well founded if we restrict it to the set of generalizations $\mathit{gen}(D) := \{C \mid D = C\theta \text{ for some } \theta\}$ of a fixed ground clause D, so that we may in fact define $\sqsupset := (\sqsupset_D)_D$ where $\sqsupset_D := \lessdot \cap (\mathit{gen}(D) \times \mathit{gen}(D))$. For this application, the possibility of taking $\sqsupset$ to be a $\mathbf{G}$-indexed family of well-founded strict partial orderings, as opposed to a single such ordering, is vital.

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 nonempty 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 be a redundancy criterion for $\models^q$ and GInf^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 $\mathbf{F}$.[3]

For each $q \in Q$, we know by Theorem 42 that the $(\mathcal{G}^q, \emptyset)$-lifting $\mathit{Red}^{q,\mathcal{G}^q,\emptyset} = (\mathit{Red}_{\mathrm{I}}^{q,\mathcal{G}^q}, \mathit{Red}_{\mathrm{F}}^{q,\mathcal{G}^q,\emptyset})$ and the $(\mathcal{G}^q, \sqsupset)$-lifting $\mathit{Red}^{q,\mathcal{G}^q,\sqsupset} = (\mathit{Red}_{\mathrm{I}}^{q,\mathcal{G}^q}, \mathit{Red}_{\mathrm{F}}^{q,\mathcal{G}^q,\sqsupset})$ of Red^q

[3] 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.

are redundancy criteria for $\models^{q}_{\mathcal{G}^q}$ and *FInf*. Consequently, by Lemma 24 the intersections

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

and

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

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

We get the following analogue of Theorem 32.

Theorem 50 *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 $Red^{\cap\mathcal{G}}$ there exists a q such that $GInf^q(\mathcal{G}^q(N)) \subseteq \mathcal{G}^q(FInf(N)) \cup Red_I^q(\mathcal{G}^q(N))$, then $(FInf, Red^{\cap\mathcal{G}})$ is statically refutationally complete w.r.t. $\models^{\cap}_{\mathcal{G}}$.*

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 $Red^{\cap\mathcal{G}}$ 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 $Red^{\cap\mathcal{G}}$ and assume that $N \models^{\cap}_{\mathcal{G}} \{\bot\}$ for some $\bot \in \mathbf{F}_\bot$. We must show that $\bot' \in N$ for some $\bot' \in \mathbf{F}_\bot$. First, we know that there exists a q such that $GInf^q(\mathcal{G}^q(N)) \subseteq \mathcal{G}^q(FInf(N)) \cup Red_{\mathrm{I}}^q(\mathcal{G}^q(N))$. Since $Red^{\cap\mathcal{G}} = \bigcap_{q\in Q} Red^{q,\mathcal{G}^q,\emptyset}$, we know by Lemma 25 that N is saturated w.r.t. *FInf* and the $(\mathcal{G}^q, \emptyset)$-lifting $Red^{q,\mathcal{G}^q,\emptyset}$ of Red^q. Therefore, by Lemma 31, $\mathcal{G}^q(N)$ is saturated w.r.t. *GInf* and Red^q.

Furthermore, $N \models^{\cap}_{\mathcal{G}} \{\bot\}$ 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 $\mathcal{G}^q(N) \models^q \mathcal{G}^q(\bot)$. By property (G1) of grounding functions, $\mathcal{G}^q(\bot)$ 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 $C \in \mathbf{F}_\bot$ by property (G2) of grounding functions. Now define $\bot' := C$. □

Since the first components of $Red^{\cap\mathcal{G}}$ and $Red^{\cap\mathcal{G},\sqsupset}$ agree, we obtain the analogues of Lemmas 43 and 44 and Theorem 45:

Lemma 51 *A set $N \subseteq \mathbf{F}$ is saturated w.r.t. FInf and $Red^{\cap\mathcal{G},\sqsupset}$ if and only if it is saturated w.r.t. FInf and $Red^{\cap\mathcal{G}}$.*

Lemma 52 *$(FInf, Red^{\cap\mathcal{G},\sqsupset})$ is statically refutationally complete w.r.t. $\models^{\cap}_{\mathcal{G}}$ if and only if $(FInf, Red^{\cap\mathcal{G}})$ is statically refutationally complete w.r.t. $\models^{\cap}_{\mathcal{G}}$.*

Theorem 53 *If $(FInf, Red^{\cap\mathcal{G}})$ is statically refutationally complete w.r.t. $\models^{\cap}_{\mathcal{G}}$, then $(FInf, Red^{\cap\mathcal{G},\sqsupset})$ is dynamically refutationally complete w.r.t. $\models^{\cap}_{\mathcal{G}}$.*

Example 54 Intersections of liftings are needed to support selection functions in ordered resolution [6] and superposition [5]. 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 example, if *fsel* selects the first literal in $C_1 = \neg\mathsf{P}(x) \vee \neg\mathsf{Q}(\mathsf{c})$ and the second literal in $C_2 = \neg\mathsf{P}(\mathsf{b}) \vee \neg\mathsf{Q}(y)$, then *gsel* could select the first literal in $D = \neg\mathsf{P}(\mathsf{b}) \vee \neg\mathsf{Q}(\mathsf{c})$ (as in C_1) or the second literal (as in C_2). For every such *gsel*, $\models^{gsel}$ is first-order entailment, GInf^{gsel} is the set of ground inferences satisfying *gsel*, and Red^{gsel} is the redundancy criterion for GInf^{gsel}. The grounding function $\mathcal{G}^{gsel}$ maps $C \in \mathbf{F}$ to $\{D \in \mathbf{G} \mid D = C\theta \text{ for some } \theta\}$ and $\iota \in \mathit{FInf}$ to the set of ground instances of ι in GInf^{gsel} with corresponding literals selected in the premises.

If ι is the *FInf*-inference

$$\frac{\mathsf{P}(\mathsf{b}) \qquad \neg\mathsf{P}(x) \vee \neg\mathsf{Q}(\mathsf{c})}{\neg\mathsf{Q}(\mathsf{c})}$$

where the first literal in the second premise C_1 is selected by *fsel*, then

$$\frac{\mathsf{P}(\mathsf{b}) \qquad \neg\mathsf{P}(\mathsf{b}) \vee \neg\mathsf{Q}(\mathsf{c})}{\neg\mathsf{Q}(\mathsf{c})}$$

is contained in $\mathcal{G}^{gsel}(\iota) \subseteq \mathit{GInf}^{gsel}$ if *gsel* selects the first literal in the right premise (as in C_1) but it is not contained in GInf^{gsel} (and hence not in $\mathcal{G}^{gsel}(\iota)$) if *gsel* selects the second literal in the right premise (as in C_2).

In the static refutational completeness proof, only one *gsel* is needed, but this *gsel* depends on the limit of a derivation and is not known *during* the derivation. Therefore, fairness must be guaranteed w.r.t. $\mathit{Red}_{\mathrm{I}}^{gsel,\mathcal{G}^{gsel}}$ for every possible extension *gsel* of *fsel*. Checking $\mathit{Red}_{\mathrm{I}}^{\cap\mathcal{G}}$ amounts to a worst-case analysis, where we must assume that every ground instance $C\theta \in \mathbf{G}$ of a premise $C \in \mathbf{F}$ inherits the selection of C.

Example 55 Intersections of liftings are also necessary for constraint superposition calculi (Nieuwenhuis and Rubio [29]). Here 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 Red^R is redundancy up to R. The grounding function $\mathcal{G}^R$ maps $C\,[\![K]\!] \in \mathbf{F}$ to $\{D \in \mathbf{G} \mid D = C\theta, K\theta = \text{true}, x\theta \text{ isR-irreducible for all } x\}$ (except in degenerate cases where x occurs only in positive literals $x \approx t$) 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. $\mathit{Red}_{\mathrm{I}}^{R,\mathcal{G}^R}$ for every convergent rewrite system R.

To obtain a practically useful criterion, the intersection $\mathit{Red}^{\cap\mathcal{G}}$ must be approximated appropriately; compare Nieuwenhuis and Rubio's Definition 6.5 (which corresponds to our $\mathit{Red}^{\cap\mathcal{G}}$) to their Lemma 6.18.

Example 56 Some calculi have inference rules that introduce Skolem function symbols. An example is the δ-elimination rule of Ganzinger and Stuber [23]. At the nonground level, the difficulty is that whenever we generate a conclusion with a fresh symbol sk_i, we need to mark all other instances of the rule with sk_j $(j \neq i)$ as redundant; otherwise, we would end up generating lots of needless conclusions. This determinism can be avoided by encoding enough information to identify the rule instance in the subscript i.

At the ground level, a second difficulty arises. The ground inference cannot simply introduce a nullary Skolem symbol—in general, this would not match the behavior of the corresponding nonground inference. Instead, it must guess both the Skolem symbol and its argument list. This guessing can be achieved using a selection function that takes the rule instance as argument and returns the Skolem term. We can then lift the ground inference by taking the intersection of all possible selection functions.

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 $Red^{\cap\mathcal{G}}$ of such criteria. As Theorem 53 demonstrates, every static refutational completeness result for *FInf* and $Red^{\cap\mathcal{G}}$—which does not generally support the deletion of subsumed formulas during a run—immediately yields a dynamic refutational completeness result for *FInf* and $Red^{\cap\mathcal{G},\sqsupset}$—which permits the deletion of subsumed formulas during a run, provided that they are larger according to $\sqsupset$.

3.4 Adding Labels

In practice, the orderings $\sqsupset_D$ used in $(\mathcal{G},\sqsupset)$-liftings 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* be a redundancy criterion for $\models$ and *GInf*, 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 $\mathbf{FL}_\bot := \mathbf{F}_\bot \times \mathbf{L}$. Notice that there are at least as many false values in $\mathbf{FL}$ as there are labels in $\mathbf{L}$. We use $\mathcal{M}$, $\mathcal{N}$ to denote labeled formula sets. Given a set $\mathcal{N} \subseteq \mathbf{FL}$, let $\lfloor\mathcal{N}\rfloor := \{C \mid (C,l) \in \mathcal{N}\}$ denote the set of formulas without their labels.

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

(L1) for every inference $(C_n,\ldots,C_0) \in$ *FInf* and every tuple $(l_1,\ldots,l_n) \in \mathbf{L}^n$, there exists an $l_0 \in \mathbf{L}$ and an inference $((C_n,l_n),\ldots,(C_0,l_0)) \in$ *FLInf*;

(L2) 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*. Define $\mathcal{G}_\mathbf{L}$ 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 $\iota \in$ *FLInf*. The following lemmas are then obvious:

Lemma 57 *$\mathcal{G}_\mathbf{L}$ 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 $Red^{\mathcal{G}_\mathbf{L},\emptyset}$ be the $(\mathcal{G}_\mathbf{L},\emptyset)$-lifting of *Red*.

Lemma 58 *$\mathcal{N} \models_{\mathcal{G}_\mathbf{L}} \mathcal{N}'$ if and only if $\lfloor\mathcal{N}\rfloor \models_\mathcal{G} \lfloor\mathcal{N}'\rfloor$.*

Lemma 59 *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 $Red^{\mathcal{G},\emptyset}$.*

Lemma 60 *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 nonempty 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 be a redundancy criterion for $\models^q$ and $GInf^q$, 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 $Red^{q,\mathcal{G}^q,\emptyset}$ of Red^q is a redundancy criterion for the $\mathcal{G}^q$-lifting $\models^q_{\mathcal{G}^q}$ of $\models^q$ and *FInf*, and so $Red^{\cap\mathcal{G}}$ is a redundancy criterion for $\models^\cap_{\mathcal{G}}$ and *FInf*.

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 57, 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 the $\mathcal{G}^q_{\mathbf{L}}$-lifting $\models^q_{\mathcal{G}^q_{\mathbf{L}}}$ of $\models^q$ and *FLInf*, and so

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

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

Analogously to Lemmas 58–60, we obtain the following results:

Lemma 61 *$\mathcal{N} \models^\cap_{\mathcal{G}_{\mathbf{L}}} \mathcal{N}'$ if and only if $\lfloor\mathcal{N}\rfloor \models^\cap_{\mathcal{G}} \lfloor\mathcal{N}'\rfloor$.*

Lemma 62 *If a set $\mathcal{N} \subseteq \mathbf{FL}$ is saturated w.r.t. FLInf and $Red^{\cap\mathcal{G}_{\mathbf{L}}}$, then $\lfloor\mathcal{N}\rfloor \subseteq \mathbf{F}$ is saturated w.r.t. FInf and $Red^{\cap\mathcal{G}}$.*

Theorem 63 *If $(FInf, Red^{\cap\mathcal{G}})$ is statically refutationally complete w.r.t. $\models^\cap_{\mathcal{G}}$, then $(FLInf, Red^{\cap\mathcal{G}_{\mathbf{L}}})$ is statically refutationally complete w.r.t. $\models^\cap_{\mathcal{G}_{\mathbf{L}}}$.*

4 Prover Architectures

We now use the above results to prove the refutational completeness of a popular prover architecture: the given clause procedure invented by McCune and Wos [28]. 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 fix 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 nonempty 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 be a redundancy criterion for $\models^q$ and $GInf^q$, and let $\mathcal{G}^q$ be a grounding function from $\mathbf{F}$ and *FInf* to $\mathbf{G}$ and $GInf^q$. Assume $(FInf, Red^{\cap\mathcal{G}})$ is statically refutationally complete w.r.t. $\models^\cap_{\mathcal{G}}$.

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 63, $(FLInf, Red^{\cap\mathcal{G}_{\mathbf{L}}})$ is statically refutationally complete w.r.t. $\models^\cap_{\mathcal{G}_{\mathbf{L}}}$.

Let $\doteq$ be an equivalence relation on $\mathbf{F}$, let $\mathrel{\cdot\!\!\succ}$ be a well-founded strict partial ordering on $\mathbf{F}$ such that $\mathrel{\cdot\!\!\succ}$ is compatible with $\doteq$ (i.e., $C \mathrel{\cdot\!\!\succ} D$, $C \doteq C'$, $D \doteq D'$ implies $C' \mathrel{\cdot\!\!\succ} D'$), such that $C \doteq D$ implies $\mathcal{G}^q(C) = \mathcal{G}^q(D)$ for all $q \in Q$, and such that $C \mathrel{\cdot\!\!>} D$ implies $\mathcal{G}^q(C) \subseteq \mathcal{G}^q(D)$ for all $q \in Q$. We define $\mathrel{\cdot\!\!\succeq} := \mathrel{\cdot\!\!>} \cup \doteq$. In practice, $\doteq$ is typically α-renaming (or equality if formulas are considered up to α-equivalence), $\mathrel{\cdot\!\!>}$ is either the instantiation ordering $\gtrdot$ (Example 46), provided it is well founded, or some well-founded ordering included in $\gtrdot$, and for every $q \in Q$, $\mathcal{G}^q$ maps every formula $C \in \mathbf{F}$ to the set of ground instances of C, possibly modulo some theory.

Let $\sqsupset\!\!\sqsupset$ 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 $C \mathrel{\cdot\!\!>} C'$ or else $C \doteq C'$ and $l \sqsupset\!\!\sqsupset l'$. By Lemma 52, the static refutational completeness of $(FLInf, Red^{\cap\mathcal{G}_\mathbf{L}})$ w.r.t. $\models^{\cap}_{\mathcal{G}_\mathbf{L}}$ implies the static refutational completeness of $(FLInf, Red^{\cap\mathcal{G}_\mathbf{L},\sqsupset})$, which by Lemma 10 implies the dynamic refutational completeness of $(FLInf, Red^{\cap\mathcal{G}_\mathbf{L},\sqsupset})$.

This result may look intimidating, so let us unroll it. 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 the first components of $Red^{\cap\mathcal{G}_\mathbf{L},\sqsupset}$ and $Red^{\cap\mathcal{G}_\mathbf{L}}$ agree, we get this result:

Lemma 64 *An $FLInf$-inference ι is redundant w.r.t. $Red^{\cap\mathcal{G}_\mathbf{L},\sqsupset}$ and $\mathcal{N}$ if and only if the underlying $FInf$-inference $\lfloor\iota\rfloor$ is redundant w.r.t. $Red^{\cap\mathcal{G}}$ and $\lfloor\mathcal{N}\rfloor$.*

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

Lemma 65 *Let $\mathcal{N} \subseteq \mathbf{FL}$, and let (C, l) be a labeled formula. Then $(C, l) \in Red_F^{\cap\mathcal{G}_\mathbf{L},\sqsupset}(\mathcal{N})$ if one of the following conditions hold:*

(i) *$C \in Red_F^{\cap\mathcal{G}}(\lfloor\mathcal{N}\rfloor)$;*
(ii) *$C \mathrel{\cdot\!\!>} C'$ for some $C' \in \lfloor\mathcal{N}\rfloor$;*
(iii) *$C \mathrel{\cdot\!\!\succeq} C'$ for some $(C', l') \in \mathcal{N}$ with $l \sqsupset\!\!\sqsupset l'$.*

Proof (i) Let $C \in Red_\mathrm{F}^{\cap\mathcal{G}}(\lfloor\mathcal{N}\rfloor)$. 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 $\mathcal{G}^q_\mathbf{L}(C, l) = \mathcal{G}^q(C)$ and $\mathcal{G}^q(\lfloor\mathcal{N}\rfloor) = \mathcal{G}^q_\mathbf{L}(\mathcal{N})$; hence $\mathcal{G}^q_\mathbf{L}(C, l) \subseteq Red_\mathrm{F}^q(\mathcal{G}^q_\mathbf{L}(\mathcal{N}))$, which implies $(C, l) \in Red_\mathrm{F}^{q,\mathcal{G}^q_\mathbf{L},\sqsupset}(\mathcal{N})$ for every $q \in Q$ and thus $(C, l) \in Red_\mathrm{F}^{\cap\mathcal{G}_\mathbf{L},\sqsupset}(\mathcal{N})$.
(ii) Assume that $C \mathrel{\cdot\!\!>} C'$ 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 $\mathcal{G}^q_\mathbf{L}(C, l) = \mathcal{G}^q(C) \subseteq \mathcal{G}^q(C') = \mathcal{G}^q_\mathbf{L}(C', l')$, which implies $(C, l) \in Red_\mathrm{F}^{q,\mathcal{G}^q_\mathbf{L},\sqsupset}(\mathcal{N})$ for every $q \in Q$ and thus $(C, l) \in Red_\mathrm{F}^{\cap\mathcal{G}_\mathbf{L},\sqsupset}(\mathcal{N})$.
(iii) If $C \mathrel{\cdot\!\!>} C'$, the result follows from (ii). Otherwise $C \doteq C'$ for some $(C', l') \in \mathcal{N}$ with $l \sqsupset\!\!\sqsupset l'$. Then $(C, l) \sqsupset (C', l')$ and $\mathcal{G}^q(C) = \mathcal{G}^q(C')$, so $\mathcal{G}^q_\mathbf{L}(C, l) = \mathcal{G}^q(C) = \mathcal{G}^q(C') = \mathcal{G}^q_\mathbf{L}(C', l')$. This implies $(C, l) \in Red_\mathrm{F}^{q,\mathcal{G}^q_\mathbf{L},\sqsupset}(\mathcal{N})$ for every $q \in Q$; therefore, $(C, l) \in Red_\mathrm{F}^{\cap\mathcal{G}_\mathbf{L},\sqsupset}(\mathcal{N})$. □

The given clause procedure that lies at the heart of saturation provers can be presented and studied abstractly.[4] We assume that the set of labels $\mathbf{L}$ contains at least two values, one

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

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. The label identifies to which formula set each formula belongs. The active label identifies the active formula set from the given clause procedure. The other, unspecified formula sets are considered passive. 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.

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

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

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

The initial state consists of the input formulas, paired with arbitrary labels different from active. A key invariant of the given clause procedure is that all inferences from active formulas are redundant w.r.t. the current set of formulas.

The PROCESS rule covers most operations performed in a theorem prover. By Lemma 65, this includes

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

Like for $\rhd_{\mathit{Red}}$, in practice the added formulas would normally be entailed by N, but we impose no soundness restrictions.

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 64, $\mathit{FLInf}(\mathcal{N}{\downarrow_{\mathsf{active}}}, \{(C, \mathsf{active})\}) \subseteq \mathit{Red}_{\mathrm{I}}^{\cap\mathcal{G}_{\mathbf{L}}}(\mathcal{N} \cup \{(C, \mathsf{active})\} \cup \mathcal{M})$ if and only if $\mathit{FInf}(\lfloor \mathcal{N}{\downarrow_{\mathsf{active}}} \rfloor, \{C\}) \subseteq \mathit{Red}_{\mathrm{I}}^{\cap\mathcal{G}}(\lfloor \mathcal{N} \rfloor \cup \{C\} \cup \lfloor \mathcal{M} \rfloor)$. 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, however, some techniques commonly implemented in theorem provers for which we need INFER's side condition in full generality.

Lemma 66 *Every* $\Longrightarrow_{\mathsf{GC}}$*-derivation is a* $\rhd_{\mathit{Red}^{\cap\mathcal{G}_{\mathbf{L}},\sqsupset}}$*-derivation.*

Proof We need to show that every labeled formula that is deleted in a $\Longrightarrow_{\mathsf{GC}}$-step is $\mathit{Red}^{\cap\mathcal{G}_{\mathbf{L}},\sqsupset}$-redundant w.r.t. the remaining labeled formulas. For PROCESS, this is trivial. For INFER, the only deleted formula is (C, l), which is $\mathit{Red}^{\cap\mathcal{G}_{\mathbf{L}},\sqsupset}$-redundant w.r.t. (C, active) by part (iii) of Lemma 65, since $l \sqsupset \mathsf{active}$. □

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

Given $k \in \mathbb{N} \cup \{\infty\}$, let $\mathit{Inv}_{\mathcal{N}}(k)$ denote the condition

$$\mathit{FLInf}(\mathcal{N}_k{\downarrow_{\mathsf{active}}}) \subseteq \textstyle\bigcup_{i=0}^{k} \mathit{Red}_{\mathrm{I}}^{\cap\mathcal{G}_{\mathbf{L}}}(\mathcal{N}_i).$$

If $(\mathcal{N}_i)_i$ is a $\rhd_{\mathit{Red}^{\cap\mathcal{G}_\mathbf{L}},\sqsupset}$-derivation and $k \in \mathbb{N}$, by (R2) and (R3), the right-hand side is equal to $\mathit{Red}_\mathrm{I}^{\cap\mathcal{G}_\mathbf{L}}(\mathcal{N}_k)$. We will show that $\mathit{Inv}_\mathcal{N}(i)$ is an invariant of GC and that it extends to the limit, enabling us to establish fairness: $\mathit{FLInf}(\mathcal{N}_\infty) \subseteq \bigcup_i \mathit{Red}_\mathrm{I}^{\cap\mathcal{G}_\mathbf{L}}(\mathcal{N}_i)$.

Lemma 67 *Let $(\mathcal{N}_i)_i$ be a $\Longrightarrow_{\mathsf{GC}}$-derivation. If $\mathcal{N}_0{\downarrow}_{\mathsf{active}} = \emptyset$, then $\mathit{Inv}_\mathcal{N}(k)$ holds for all indices k.*

Proof Base case: The hypothesis $\mathcal{N}_0{\downarrow}_{\mathsf{active}} = \emptyset$ and the exclusion of premise-free inferences ensure that $\mathit{FLInf}({\downarrow}_{\mathsf{active}}) = \emptyset$ and hence $\mathit{Inv}_\mathcal{N}(0)$ holds.

Case Process: Consider the step $\mathcal{N}_k = \mathcal{N} \cup \mathcal{M} \Longrightarrow_{\mathsf{GC}} \mathcal{N} \cup \mathcal{M}' = \mathcal{N}_{k+1}$. We have the inclusion chain $\mathit{FLInf}(\mathcal{N}_{k+1}{\downarrow}_{\mathsf{active}}) \subseteq \mathit{FLInf}(\mathcal{N}_k{\downarrow}_{\mathsf{active}}) \subseteq \bigcup_{i=0}^{k} \mathit{Red}_\mathrm{I}^{\cap\mathcal{G}_\mathbf{L}}(\mathcal{N}_i) \subseteq \bigcup_{i=0}^{k+1} \mathit{Red}_\mathrm{I}^{\cap\mathcal{G}_\mathbf{L}}(\mathcal{N}_i)$. The first inclusion relies on Process's side condition that $\mathcal{M}'{\downarrow}_{\mathsf{active}} = \emptyset$. The second inclusion corresponds to the induction hypothesis.

Case Infer: Consider the step $\mathcal{N}_k = \mathcal{N} \cup \{(C, l)\} \Longrightarrow_{\mathsf{GC}} \mathcal{N} \cup \{(C, \mathsf{active})\} \cup \mathcal{M} = \mathcal{N}_{k+1}$. We assume $\iota \in \mathit{FLInf}(\mathcal{N}_{k+1}{\downarrow}_{\mathsf{active}})$ for some ι and show $\iota \in \bigcup_{i=0}^{k+1} \mathit{Red}_\mathrm{I}^{\cap\mathcal{G}_\mathbf{L}}(\mathcal{N}_i)$. If $\iota \in \mathit{FLInf}(\mathcal{N}{\downarrow}_{\mathsf{active}}, (C, \mathsf{active}))$, then $\iota \in \mathit{Red}_\mathrm{I}^{\cap\mathcal{G}_\mathbf{L}}(\mathcal{N}_{k+1})$ by Infer's last side condition. Otherwise, $\iota \in \mathit{FLInf}(\mathcal{N}{\downarrow}_{\mathsf{active}})$ by Infer's side condition that $\mathcal{M}{\downarrow}_{\mathsf{active}} = \emptyset$. By the induction hypothesis, $\iota \in \bigcup_{i=0}^{k} \mathit{Red}_\mathrm{I}^{\cap\mathcal{G}_\mathbf{L}}(\mathcal{N}_i)$. In both cases, $\iota \in \bigcup_{i=0}^{k+1} \mathit{Red}_\mathrm{I}^{\cap\mathcal{G}_\mathbf{L}}(\mathcal{N}_i)$. □

Lemma 68 *Let $(\mathcal{N}_i)_i$ be a nonempty $\mathcal{P}(\mathbf{FL})$-sequence. If $\mathit{Inv}_\mathcal{N}(i)$ holds for all indices i, then $\mathit{Inv}_\mathcal{N}(\infty)$ holds.*

Proof We assume $\iota \in \mathit{FLInf}(\mathcal{N}_\infty{\downarrow}_{\mathsf{active}})$ for some ι and show $\iota \in \bigcup_i \mathit{Red}_\mathrm{I}^{\cap\mathcal{G}_\mathbf{L}}(\mathcal{N}_i)$. For ι to be in $\mathit{FLInf}(\mathcal{N}_\infty{\downarrow}_{\mathsf{active}})$, all of its finitely many premises must be in $\mathcal{N}_\infty{\downarrow}_{\mathsf{active}}$. Therefore, there must exist an index k such that $\mathcal{N}_k{\downarrow}_{\mathsf{active}}$ contains all of them, and therefore $\iota \in \mathit{FLInf}(\mathcal{N}_k{\downarrow}_{\mathsf{active}})$. Since $\mathit{Inv}_\mathcal{N}(k)$ holds, $\iota \in \bigcup_{i=0}^{k} \mathit{Red}_\mathrm{I}^{\cap\mathcal{G}_\mathbf{L}}(\mathcal{N}_i) \subseteq \bigcup_i \mathit{Red}_\mathrm{I}^{\cap\mathcal{G}_\mathbf{L}}(\mathcal{N}_i)$. □

Lemma 69 *Let $(\mathcal{N}_i)_i$ be a $\Longrightarrow_{\mathsf{GC}}$-derivation. If $\mathcal{N}_0{\downarrow}_{\mathsf{active}} = \emptyset$ and $\mathcal{N}_\infty{\downarrow}_l = \emptyset$ for all $l \neq \mathsf{active}$, then $(\mathcal{N}_i)_i$ is fair.*

Proof By Lemmas 67 and 68, $\mathit{FLInf}(\mathcal{N}_\infty{\downarrow}_{\mathsf{active}}) \subseteq \bigcup_i \mathit{Red}_\mathrm{I}^{\cap\mathcal{G}_\mathbf{L}}(\mathcal{N}_i)$. By the second hypothesis, this inclusion simplifies to $\mathit{FLInf}(\mathcal{N}_\infty) \subseteq \bigcup_i \mathit{Red}_\mathrm{I}^{\cap\mathcal{G}_\mathbf{L}}(\mathcal{N}_i)$. □

Theorem 70 *Let $(\mathcal{N}_i)_i$ be a $\Longrightarrow_{\mathsf{GC}}$-derivation, where $\mathcal{N}_0{\downarrow}_{\mathsf{active}} = \emptyset$ and $\mathcal{N}_\infty{\downarrow}_l = \emptyset$ for all $l \neq \mathsf{active}$. If $\lfloor\mathcal{N}_0\rfloor \models_\mathcal{G}^\cap \{\bot\}$ 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 61, $\lfloor\mathcal{N}_0\rfloor \models_\mathcal{G}^\cap \{\bot\}$ is equivalent to $\mathcal{N}_0 \models_{\mathcal{G}_\mathbf{L}}^\cap \{(\bot, \mathsf{active})\}$. By Lemmas 66 and 69, we know that $(\mathcal{N}_i)_i$ is a fair $\rhd_{\mathit{Red}^{\cap\mathcal{G}_\mathbf{L}},\sqsupset}$-derivation. Since $(\mathit{FLInf}, \mathit{Red}^{\cap\mathcal{G}_\mathbf{L},\sqsupset})$ 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 71 The following Otter loop [28, 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 [46, 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 $N \mid \emptyset \mid \emptyset \mid \emptyset \mid \emptyset$.

$$
\begin{array}{rl}
\textsc{ChooseN} & N \uplus \{C\} \mid \emptyset \mid P \mid \emptyset \mid A \Longrightarrow_{\mathsf{OL}} N \mid \{C\} \mid P \mid \emptyset \mid A \\
\textsc{DeleteFwd} & N \mid \{C\} \mid P \mid \emptyset \mid A \Longrightarrow_{\mathsf{OL}} N \mid \emptyset \mid P \mid \emptyset \mid A \\
& \text{if } C \in \mathit{Red}_{\mathrm{F}}^{\cap\mathcal{G}}(P \cup A) \text{ or } C \mathrel{\cdot\!\succeq} C' \text{ for some } C' \in P \cup A \\
\textsc{SimplifyFwd} & N \mid \{C\} \mid P \mid \emptyset \mid A \Longrightarrow_{\mathsf{OL}} N \mid \{C'\} \mid P \mid \emptyset \mid A \\
& \text{if } C \in \mathit{Red}_{\mathrm{F}}^{\cap\mathcal{G}}(P \cup A \cup \{C'\}) \\
\textsc{DeleteBwdP} & N \mid \{C\} \mid P \uplus \{C'\} \mid \emptyset \mid A \Longrightarrow_{\mathsf{OL}} N \mid \{C\} \mid P \mid \emptyset \mid A \\
& \text{if } C' \in \mathit{Red}_{\mathrm{F}}^{\cap\mathcal{G}}(\{C\}) \text{ or } C' \mathrel{\cdot\!\succ} C \\
\textsc{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 \\
& \text{if } C' \in \mathit{Red}_{\mathrm{F}}^{\cap\mathcal{G}}(\{C, C''\}) \\
\textsc{DeleteBwdA} & N \mid \{C\} \mid P \mid \emptyset \mid A \uplus \{C'\} \Longrightarrow_{\mathsf{OL}} N \mid \{C\} \mid P \mid \emptyset \mid A \\
& \text{if } C' \in \mathit{Red}_{\mathrm{F}}^{\cap\mathcal{G}}(\{C\}) \text{ or } C' \mathrel{\cdot\!\succ} C \\
\textsc{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 \\
& \text{if } C' \in \mathit{Red}_{\mathrm{F}}^{\cap\mathcal{G}}(\{C, C''\}) \\
\textsc{Transfer} & N \mid \{C\} \mid P \mid \emptyset \mid A \Longrightarrow_{\mathsf{OL}} N \mid \emptyset \mid P \cup \{C\} \mid \emptyset \mid A \\
\textsc{ChooseP} & \emptyset \mid \emptyset \mid P \uplus \{C\} \mid \emptyset \mid A \Longrightarrow_{\mathsf{OL}} \emptyset \mid \emptyset \mid P \mid \{C\} \mid A \\
\textsc{Infer} & \emptyset \mid \emptyset \mid P \mid \{C\} \mid A \Longrightarrow_{\mathsf{OL}} M \mid \emptyset \mid P \mid \emptyset \mid A \cup \{C\} \\
& \text{if } \mathit{FInf}(A, \{C\}) \subseteq \mathit{Red}_{\mathrm{I}}^{\cap\mathcal{G}}(A \cup \{C\} \cup M)
\end{array}
$$

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.

If we are interested in soundness, we can require that the formulas added by simplification and INFER are $\mathrel{|\!\approx}$-entailed by the formulas in the state before the transition. This can be relaxed to consistency preservation—e.g., for calculi that perform skolemization.

A reasonable strategy for applying the OL rules is presented below. It relies on a well-founded ordering $\succ$ on formulas to ensure that the simplification rules actually "simplify" their target, preventing nontermination of the inner loop. It also assumes 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 $N_\infty = X_\infty = P_\infty = Y_\infty = \emptyset$. Therefore, by Theorem 70, OL is dynamically refutationally complete.

In most saturation calculi, *Red* is defined in terms of some total 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 implemented in resolution or superposition provers.

To ensure fairness, the heuristic used to apply CHOOSEP must guarantee that no formula remains indefinitely in P. Fair choice strategies typically rely on formula age, which can be represented through labels. Consider labeled formulas (C, t), where t is the timestamp, and a labeled version of OL where formulas introduced by simplification or INFER are labeled with strictly increasing timestamps.

Example 72 One fair formula choice strategy is to alternate between heuristically choosing n formulas and taking the formula with the smallest timestamp [28, Sect. 2.3.1].

Proof By contradiction. Assume $P_\infty \neq \emptyset$. Consider the formula $(C, t) \in P_\infty$ with the smallest timestamp t. There exists an index i such that C is the formula with the smallest timestamp in P_i. After at most $n + 1$ applications of CHOOSEP, C will be chosen. □

Example 73 Another fair option is to use an $\mathbb{N}$-valued weight function w that is strictly monotonic in the timestamp—i.e., for any unlabeled formula C, if $t < t'$, then $w(C, t) < w(C, t')$—and take a formula with the smallest weight [37, Sect. 4].

Proof Consider the labeled formula (C, t) with the smallest weight in P_∞. The weight function satisfies the inequation $n \leq w(D, n)$ for every n and every unlabeled D. Therefore, after $w(C, t)$ applications of INFER, new formulas introduced by simplification or INFER all have a weight larger than (C, t), and thus CHOOSEP will eventually have to choose (C, t). □

Example 74 In its superposition module [20], iProver implements a rule that eliminates the chosen passive clause, or *given clause*, if it is redundant w.r.t. a subset of its child clauses together with the active set. The following iProver loop prover IL captures this. It is based on GC and consists of all the OL transition rules and of the following rule:

$$\text{REPLACE} \quad \emptyset \mid \emptyset \mid P \mid \{C\} \mid A \Longrightarrow_{\mathsf{IL}} M \mid \emptyset \mid P \mid \emptyset \mid A$$
$$\text{if either } C \in Red_{\mathrm{F}}^{\cap\mathcal{G}}(A \cup M) \text{ or else } M = \{C'\} \text{ and } C \mathrel{\cdot\!\!\succ} C'$$

As M, iProver would use a set of possibly simplified clauses from $concl(FInf(A, \{C\}))$.

Example 75 Bachmair and Ganzinger's resolution prover RP [6, Sect. 4.3] is another instance of GC. It embodies both a concrete prover architecture and a concrete inference system: ordered resolution with selection ($\mathsf{O}_S^{\succ}$). States are triples $N \mid P \mid O$ of finite clause sets consisting of new, processed, and old (active) clauses, respectively. The instantiation relies on three labels $l_3 \sqsupset l_2 \sqsupset l_1 = \mathsf{active}$. Subsumption can be supported as described in Example 46.

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 $N = \mathit{concl}(\mathsf{O}^{\succ}_{S}(O, C))$

Let $(N_i \mid P_i \mid O_i)_i$ be a $\Longrightarrow_{\mathsf{RP}}$-derivation, where $P_0 = O_0 = \emptyset$. Since the transition system excluding Infer terminates [37, Sect. 4] and we can always apply Choose to empty N, we have $N_\infty = \emptyset$. The key restriction that is needed to ensure fairness is that the choice of C in Infer must be fair. This ensures $P_\infty = \emptyset$. Thus, by Theorem 70, RP is dynamically refutationally complete. Incidentally, our version of RP repairs a small mistake in Bachmair and Ganzinger's definition of the notation $\mathit{Inf}(N, \{C\})$, used in the Infer rule [36, Sect. 7.1].

4.2 Delayed Inferences

The given clause prover GC presented in the previous subsection is sufficient to describe a prover based on an Otter loop as well as a basic DISCOUNT loop prover—which differs from the Otter loop prover OL in that the passive formulas are neither simplified or deleted using SimplifyBwdP or DeleteBwdP, nor are they used to simplify or delete other formulas in SimplifyFwd or DeleteFwd. To describe a DISCOUNT loop prover with orphan formula deletion, however, we need to extend GC.

An *orphan* formula is a passive formula generated by an inference for which at least one premise is no longer active.

To model orphan formula 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 more compactly than a new formula, as a tuple of premises together with instructions on how to compute the conclusion.

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

$$\begin{array}{ll}
\textsc{Process} & (T,\, \mathcal{N} \cup \mathcal{M}) \Longrightarrow_{\mathsf{LGC}} (T,\, \mathcal{N} \cup \mathcal{M}') \\
& \text{where } \mathcal{M} \subseteq \mathit{Red}_{\mathrm{F}}^{\cap \mathcal{G}_{\mathrm{L}}, \sqsupset}(\mathcal{N} \cup \mathcal{M}') \text{ and } \mathcal{M}'{\downarrow}_{\mathsf{active}} = \emptyset \\
\textsc{ScheduleInfer} & (T,\, \mathcal{N} \cup \{(C, l)\}) \Longrightarrow_{\mathsf{LGC}} (T \cup T',\, \mathcal{N} \cup \{(C, \mathsf{active})\}) \\
& \text{where } l \neq \mathsf{active} \text{ and } T' = \mathit{FInf}(\lfloor \mathcal{N}{\downarrow}_{\mathsf{active}} \rfloor, \{C\}) \\
\textsc{ComputeInfer} & (T \cup \{\iota\},\, \mathcal{N}) \Longrightarrow_{\mathsf{LGC}} (T,\, \mathcal{N} \cup \mathcal{M}) \\
& \text{where } \mathcal{M}{\downarrow}_{\mathsf{active}} = \emptyset \text{ and } \iota \in \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}}(\lfloor \mathcal{N} \cup \mathcal{M} \rfloor) \\
\textsc{DeleteOrphans} & (T \cup T',\, \mathcal{N}) \Longrightarrow_{\mathsf{LGC}} (T,\, \mathcal{N}) \\
& \text{where } T' \cap \mathit{FInf}(\lfloor \mathcal{N}{\downarrow}_{\mathsf{active}} \rfloor) = \emptyset
\end{array}$$

Initial states are states $(T, \mathcal{N})$ such that T consists of all premise-free inferences of *FInf* and $\mathcal{N}$ contains the input formulas paired with arbitrary labels different from active. A key invariant of LGC is that all inferences from active formulas are either scheduled in T or redundant w.r.t. $\mathcal{N}$.

Process has the same behavior as the corresponding GC rule, except for the additional T component, which it ignores.

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 the rule cannot delete premise-free inferences, since the side condition is then trivially false.

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 [24], or as a set of explicit formulas C_0 with information about their parents $(C_n, \ldots, C_1)$ as in E [39]. 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 $(\iota, \mathit{simp}(C_0))$.

Lemma 76 *If $(T_i, \mathcal{N}_i)_i$ is an $\Longrightarrow_{\mathsf{LGC}}$-derivation, then $(\mathcal{N}_i)_i$ is a $\rhd_{\mathit{Red}^{\cap \mathcal{G}_{\mathrm{L}}, \sqsupset}}$-derivation.*

Proof We must show that every labeled formula that is deleted in an $\Longrightarrow_{\mathsf{LGC}}$-step from the $\mathcal{N}$ component is $\mathit{Red}^{\cap \mathcal{G}_{\mathrm{L}}, \sqsupset}$-redundant w.r.t. the remaining labeled formulas. For Process this is trivial. For ScheduleInfer, the only deleted formula is (C, l), which is $\mathit{Red}^{\cap \mathcal{G}_{\mathrm{L}}, \sqsupset}$-redundant w.r.t. (C, active) by part (iii) of Lemma 65, since $l \sqsupset \mathsf{active}$. Finally, the rules ComputeInfer and DeleteOrphans do not delete any formulas. □

Given $k \in \mathbb{N} \cup \{\infty\}$, let $\mathit{Inv}_{T,\mathcal{N}}(k)$ denote the condition

$$\mathit{FLInf}(\mathcal{N}_k{\downarrow}_{\mathsf{active}}) \subseteq \lceil T_k \rceil \cup \bigcup_{i=0}^{k} \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}_{\mathrm{L}}}(\mathcal{N}_i)$$

where $\iota \in \lceil T \rceil$ if and only if $\lfloor \iota \rfloor \in T$ for every ι and every T. We will show that $\mathit{Inv}_{T,\mathcal{N}}(i)$ is an invariant of LGC and that it extends to the limit, enabling us to establish fairness.

Lemma 77 *Let $(T_i, \mathcal{N}_i)_i$ be an $\Longrightarrow_{\mathsf{LGC}}$-derivation. If $\mathcal{N}_0{\downarrow}_{\mathsf{active}} = \emptyset$ and $T_0 \supseteq \mathit{FInf}(\emptyset)$, then $\mathit{Inv}_{T,\mathcal{N}}(k)$ holds for all indices k.*

Proof Base case: The hypotheses ensure that $\mathit{FLInf}(\mathcal{N}_0{\downarrow}_{\mathsf{active}}) = \mathit{FLInf}(\emptyset) \subseteq \lceil T_0 \rceil$.
Case Process: This case is essentially as in the proof of Lemma 67.

CASE SCHEDULEINFER: Consider the step $(T_k, \mathcal{N}_k) = (T, \mathcal{N} \cup \{(C, l)\}) \Longrightarrow_{\mathsf{LGC}} (T \cup T', \mathcal{N} \cup \{(C, \mathsf{active})\}) = (T_{k+1}, \mathcal{N}_{k+1})$. We assume $\iota \in \mathit{FLInf}(\mathcal{N}_{k+1}{\downarrow}_{\mathsf{active}})$ for some ι and show $\iota \in \lceil T \cup T' \rceil \cup \bigcup_{i=0}^{k+1} \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N}_i)$. If $\iota \in \mathit{FLInf}(\mathcal{N}{\downarrow}_{\mathsf{active}}, (C, \mathsf{active}))$, then $\lfloor \iota \rfloor \in T'$ by definition of T' and thus $\iota \in \lceil T \cup T' \rceil$. Otherwise, $\iota \in \mathit{FLInf}(\mathcal{N}{\downarrow}_{\mathsf{active}})$. By the induction hypothesis, $\iota \in \lceil T \rceil \cup \bigcup_{i=0}^{k} \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N}_i)$. In both cases, $\iota \in \lceil T \cup T' \rceil \cup \bigcup_{i=0}^{k+1} \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N}_i)$.
CASE COMPUTEINFER: Consider the step $(T_k, \mathcal{N}_k) = (T \cup \{\iota'\}, \mathcal{N}) \Longrightarrow_{\mathsf{LGC}} (T, \mathcal{N} \cup \mathcal{M}) = (T_{k+1}, \mathcal{N}_{k+1})$. We assume $\iota \in \mathit{FLInf}(\mathcal{N}_{k+1}{\downarrow}_{\mathsf{active}})$ for some ι and show $\iota \in \lceil T \rceil \cup \bigcup_{i=0}^{k+1} \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N}_i)$. By COMPUTEINFER's side condition that $\mathcal{M}{\downarrow}_{\mathsf{active}} = \emptyset$, we have that $\iota \in \mathit{FLInf}(\mathcal{N}{\downarrow}_{\mathsf{active}})$. By the induction hypothesis, $\iota \in \lceil T \cup \{\iota'\} \rceil \cup \bigcup_{i=0}^{k} \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N}_i)$. If $\iota \in \lceil \{\iota'\} \rceil$, then $\iota \in \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N} \cup \mathcal{M})$ by COMPUTEINFER's last side condition. Otherwise, $\iota \in \lceil T \rceil \cup \bigcup_{i=0}^{k} \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N}_i)$. In both cases, $\iota \in \lceil T \rceil \cup \bigcup_{i=0}^{k+1} \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N}_i)$.
CASE DELETEORPHANS: Consider the transition $(T_k, \mathcal{N}_k) = (T \cup T', \mathcal{N}) \Longrightarrow_{\mathsf{LGC}} (T, \mathcal{N}) = (T_{k+1}, \mathcal{N}_{k+1})$. We assume $\iota \in \mathit{FLInf}(\mathcal{N}{\downarrow}_{\mathsf{active}})$ for some ι. By the induction hypothesis, $\iota \in \lceil T \cup T' \rceil \cup \bigcup_{i=0}^{k+1} \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N}_i)$. Since $\iota \notin \lceil T' \rceil$ by DELETEORPHANS's side condition, we have $\iota \in \lceil T \rceil \cup \bigcup_{i=0}^{k+1} \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N}_i)$. □

Lemma 78 *Let $(T_i, \mathcal{N}_i)_i$ be a nonempty $(\mathcal{P}(\mathit{FInf}) \times \mathcal{P}(\mathbf{FL}))$-sequence. If $\mathit{Inv}_{T,\mathcal{N}}(i)$ holds for all indices i, then $\mathit{Inv}_{T,\mathcal{N}}(\infty)$ holds.*

Proof We assume $\iota \in \mathit{FLInf}(\mathcal{N}_\infty{\downarrow}_{\mathsf{active}})$ for some ι and show $\iota \in \lceil T_\infty \rceil \cup \bigcup_i \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N}_i)$. Assume $\iota \notin \lceil T_\infty \rceil$. Clearly, there must exist an index k such that $\mathcal{N}_k{\downarrow}_{\mathsf{active}}$ contains all of ι's premises and $\iota \notin \lceil T_k \rceil$. Therefore $\iota \in \mathit{FLInf}(\mathcal{N}_k{\downarrow}_{\mathsf{active}})$. Since $\mathit{Inv}_{T,\mathcal{N}}(k)$ holds, $\iota \in \bigcup_{i=0}^{k} \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N}_i) \subseteq \bigcup_i \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N}_i)$. □

Lemma 79 *Let $(T_i, \mathcal{N}_i)_i$ be an $\Longrightarrow_{\mathsf{LGC}}$-derivation. If $\mathcal{N}_0{\downarrow}_{\mathsf{active}} = \emptyset$, $\mathcal{N}_\infty{\downarrow}_l = \emptyset$ for all $l \neq \mathsf{active}$, $T_0 \supseteq \mathit{FInf}(\emptyset)$, and $T_\infty = \emptyset$, then $(\mathcal{N}_i)_i$ is fair.*

Proof By Lemmas 67 and 68, $\mathit{FLInf}(\mathcal{N}_\infty{\downarrow}_{\mathsf{active}}) \subseteq \lceil T_\infty \rceil \cup \bigcup_i \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N}_i)$. By the second and fourth hypotheses, this inclusion simplifies to $\mathit{FLInf}(\mathcal{N}_\infty) \subseteq \bigcup_i \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}\mathbf{L}}(\mathcal{N}_i)$. □

Theorem 80 *Let $(T_i, \mathcal{N}_i)_i$ be an $\Longrightarrow_{\mathsf{LGC}}$-derivation, where $\mathcal{N}_0{\downarrow}_{\mathsf{active}} = \emptyset$, $\mathcal{N}_\infty{\downarrow}_l = \emptyset$ for all $l \neq \mathsf{active}$, $T_0 \supseteq \mathit{FInf}(\emptyset)$, and $T_\infty = \emptyset$. If $\lfloor \mathcal{N}_0 \rfloor \models_{\mathcal{G}}^{\cap} \{\bot\}$ 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 61, $\lfloor \mathcal{N}_0 \rfloor \models_{\mathcal{G}}^{\cap} \{\bot\}$ is equivalent to $\mathcal{N}_0 \models_{\mathcal{G}\mathbf{L}}^{\cap} \{(\bot, \mathsf{active})\}$. By Lemmas 76 and 79, we know that $(\mathcal{N}_i)_i$ is a fair $\rhd_{\mathit{Red}^{\cap \mathcal{G}\mathbf{L}}, \sqsupset}$-derivation. Since $(\mathit{FLInf}, \mathit{Red}^{\cap \mathcal{G}\mathbf{L}, \sqsupset})$ 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 81 The following DISCOUNT loop [1] prover DL is an instance of the lazy given clause prover LGC. This loop design is inspired by Schulz's description of E [39] 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, the passive formulas, and the 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 is the set of all premise-free inferences of FInf.

COMPUTEINFER $\quad T \uplus \{\iota\} \mid P \mid \emptyset \mid A \Longrightarrow_{\mathsf{DL}} T \mid P \mid \{C\} \mid A$
$\quad$ if $\iota \in \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}}(A \cup \{C\})$

$$
\begin{array}{rl}
\textsc{ChooseP} & T \mid P \uplus \{C\} \mid \emptyset \mid A \Longrightarrow_{\mathsf{DL}} T \mid P \mid \{C\} \mid A \\
\textsc{DeleteFwd} & T \mid P \mid \{C\} \mid A \Longrightarrow_{\mathsf{DL}} T \mid P \mid \emptyset \mid A \\
 & \text{if } C \in \mathit{Red}_{\mathrm{F}}^{\cap\mathcal{G}}(A) \text{ or } C \mathrel{\cdot\!\succeq} C' \text{ for some } C' \in A \\
\textsc{SimplifyFwd} & T \mid P \mid \{C\} \mid A \Longrightarrow_{\mathsf{DL}} T \mid P \mid \{C'\} \mid A \\
 & \text{if } C \in \mathit{Red}_{\mathrm{F}}^{\cap\mathcal{G}}(A \cup \{C'\}) \\
\textsc{DeleteBwd} & T \mid P \mid \{C\} \mid A \uplus \{C'\} \Longrightarrow_{\mathsf{DL}} T \mid P \mid \{C\} \mid A \\
 & \text{if } C' \in \mathit{Red}_{\mathrm{F}}^{\cap\mathcal{G}}(\{C\}) \text{ or } C' \mathrel{\cdot\!\succ} C \\
\textsc{SimplifyBwd} & T \mid P \mid \{C\} \mid A \uplus \{C'\} \Longrightarrow_{\mathsf{DL}} T \mid P \cup \{C''\} \mid \{C\} \mid A \\
 & \text{if } C' \in \mathit{Red}_{\mathrm{F}}^{\cap\mathcal{G}}(\{C, C''\}) \\
\textsc{ScheduleInfer} & T \mid P \mid \{C\} \mid A \Longrightarrow_{\mathsf{DL}} T \cup T' \mid P \mid \emptyset \mid A \cup \{C\} \\
 & \text{if } T' = \mathit{FInf}(A, \{C\}) \\
\textsc{DeleteOrphans} & T \uplus T' \mid P \mid Y \mid A \Longrightarrow_{\mathsf{DL}} T \mid P \mid Y \mid A \\
 & \text{if } T' \cap \mathit{FInf}(A) = \emptyset
\end{array}
$$

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 $\mathit{FInf}(N, \{C\})$ 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 priority queue, to ensure fairness.
   - 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 $l_3 \sqsupset l_2 \sqsupset l_1 = \mathsf{active}$ corresponding to the sets P, Y, A, respectively.

Example 82 Higher-order unification can give rise to infinitely many incomparable unifiers. As a result, in λ-superposition [14], performing all inferences between two clauses can lead to infinitely many conclusions, which need to be enumerated fairly. The Zipperposition prover [14], which implements the calculus, performs this enumeration in an extended DISCOUNT loop.

Infinitary inference rules are also useful to reason about the theory of datatypes and codatatypes. Superposition with (co)datatypes [19] includes n-ary Acycl and Uniq rules, which had to be restricted and complemented with axioms so that they could be implemented in Vampire [27]. In Zipperposition, it would be 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 81). The ChooseP, DeleteFwd, SimplifyFwd, DeleteBwd, and SimplifyBwd rules are essentially 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 $\iota_0 \in \mathit{Red}_{\mathrm{I}}^{\cap \mathcal{G}}(A \cup \{C\})$

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 $(\iota_i^j)_i^j$ 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. The same rule can also be used to remove empty sequences from T, since the side condition is then vacuously true, thereby providing a form of garbage collection.

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.

Remark 83 The above approach to orphan formula deletion works because formulas recognized as orphans, belonging to the T state component, cannot have been used to make other formulas or inferences redundant—only passive and active formulas are considered by the redundancy criterion. If formulas from the P or A set could be detected as orphans, we could lose refutational completeness. To see this, consider the abstract scenario in which a formula C that is crucial for a refutation is subsumed by D, which is in turn deleted for being an orphan formula. Then C is lost forever even if is not an orphan formula.

Nevertheless, the idea of detecting orphan formulas outside the scheduled inferences in T can be salvaged as follows: Annotate each formula D with its parentage, and whenever D is used to simplify other clauses or to make other inferences redundant, remember this fact. Only consider D an orphan formula if it has lost a parent and if it has never beeen used to delete other formulas or to make other inferences redundant.

4.3 Integrating Saturation Calculi

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 others, some reformulation of the redundancy criterion may be necessary.

Example 84 As explained in Examples 54 and 55, redundancy criteria for calculi with selection functions [5, 6] or constraints [29, 30] can be defined as intersections $Red^{\cap\mathcal{G}}$ of lifted redundancy criteria.

Example 85 In Bachmair and Ganzinger's associative–commutative (AC) superposition calculus [4], the redundancy of general clauses and inferences is defined using a grounding function $\mathcal{G}$ that maps every clause C to the set of its ground instances $C\theta$ and every inference ι to the set of its ground instances $\iota\theta$. ("Instance" means "syntactic instance" here, that is, not "instance modulo AC.") In principle, one could now apply $(\mathcal{G}, \sqsupset)$-lifting, where we choose $\sqsupset$ as the instantiation ordering modulo AC. This would be pointless, though, since in the definition of $Red_{\mathrm{F}}^{\mathcal{G},\sqsupset}$ the ordering $\sqsupset$ is used only if D is a common syntactic instance of C and C'. Note that, for example, $C = \mathsf{f}(\mathsf{c} + (\mathsf{c} + z)) \approx \mathsf{b}$ is an AC-instance of $C' = \mathsf{f}((x + x) + y) \approx \mathsf{b}$, but since C and C' have no common syntactic ground instances, this fact is never exploited in $Red_{\mathrm{F}}^{\mathcal{G},\sqsupset}$. We can repair this by redefining $\mathcal{G}$ so that it maps every ι to the set of its syntactic ground instances $\iota\theta$, as before, but C to the set of all D that are AC-equal to some ground instance $C\theta$. This qualifies as a grounding function as well, and since Bachmair and Ganzinger's definition of redundancy for ground clauses is invariant under AC, the new definition of redundancy for general clauses is equivalent to the old one.

Example 86 Waldmann [44] 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 such 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 87 The definition of redundancy for Bachmair, Ganzinger, and Waldmann's hierarchic superposition calculus [8] 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 is inconsistent w.r.t. some base (background) theory. For these inferences, $\mathcal{G}(\iota) = \mathit{undef}$.

Baumgartner and Waldmann's variant of hierarchic superposition [10] uses a slightly different definition of redundancy: A clause C is redundant if $\mathcal{G}(C) \subseteq Red_{\mathrm{F}}(\mathcal{G}(N) \cup \mathit{Th}) \cup \mathit{Th}$; a non-CLOSE inference ι is redundant if $\mathcal{G}(\iota) \subseteq Red_{\mathrm{I}}(\mathcal{G}(N) \cup \mathit{Th})$, where Th 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 $Red_{\mathrm{F}}^{\mathit{Th}}(M) := Red_{\mathrm{F}}(M \cup \mathit{Th}) \cup \mathit{Th}$, and $Red_{\mathrm{I}}^{\mathit{Th}}(M) := Red_{\mathrm{I}}(M \cup \mathit{Th})$. It is easy to check that $Red^{\mathit{Th}} := (Red_{\mathrm{I}}^{\mathit{Th}}, Red_{\mathrm{F}}^{\mathit{Th}})$ is also a redundancy criterion and that the properties above are equivalent to $\mathcal{G}(C) \subseteq Red_{\mathrm{F}}^{\mathit{Th}}(\mathcal{G}(N))$ and $\mathcal{G}(\iota) \subseteq Red_{\mathrm{I}}^{\mathit{Th}}(\mathcal{G}(N))$. For CLOSE inferences, we have again $\mathcal{G}(\iota) = \mathit{undef}$.

Example 88 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. The formulas are the rewrite rules and equations. The only inferences are orientation and critical pair computation; the other inferences of the unfailing completion calculus (e.g., simplifications of equations or rules) must be considered as simplifications in our framework, rather than as inferences. With these definitions, formulas and inferences are redundant if for every rewrite proof using that rewrite rule, equation, or critical peak, there exists a smaller rewrite proof.[5]

The requirement that the redundancy criterion must be obtained by lifting (which is necessary to introduce the labeling) can then be trivially fulfilled by "self-lifting"—i.e., by defining $\mathbf{G} := \mathbf{F}$ and $\cdot\!\succ := \emptyset$ 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 example, with a term ordering satisfying $\mathsf{f(c)} \succ \mathsf{f(b)} \succ \mathsf{f(a)} \succ \mathsf{c} \succ \mathsf{b} \succ \mathsf{a}$, the equations $\mathsf{c} \approx \mathsf{b}$ and $\mathsf{c} \approx \mathsf{a}$ make $\mathsf{f(b)} \approx \mathsf{f(a)}$ redundant in the superposition calculus (since they are smaller in the induced clause ordering), but they do not make $\mathsf{f(b)} \approx \mathsf{f(a)}$ redundant in unfailing completion (since the rewrite proof $\mathsf{f(b)} \leftrightarrow \mathsf{f(c)} \leftrightarrow \mathsf{f(a)}$ using $\mathsf{c} \approx \mathsf{b}$ and $\mathsf{c} \approx \mathsf{a}$ is larger than the rewrite proof $\mathsf{f(b)} \leftrightarrow \mathsf{f(a)}$ using $\mathsf{f(b)} \approx \mathsf{f(a)}$).

5 Isabelle Development

The framework described in the previous sections has been formalized in Isabelle/HOL [31, 32], including all the theorems and lemmas, the prover architectures GC and LGC, and the example prover RP. The Isabelle theory files are available in the *Archive of Formal Proofs* [16, 40]. The development is also part of the IsaFoL (Isabelle Formalization of Logic) [17] effort, which aims at developing a reusable computer-checked library of results about automated reasoning.

The main theory files of the development are listed below:

– `Calculus.thy` collects basic definitions and lemmas about consequence relations, inference systems, and redundancy criteria, including the equivalence of static and dynamic refutational completeness.
– `Calculus_Variations.thy` contains alternative notions of inferences, redundancy, saturation, and completeness found in the literature.
– `Intersection_Calculus.thy` introduces calculi equipped with a family of redundancy criteria, whose intersection is taken.
– `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.
– `Given_Clause_Architectures.thy` and `Given_Clause_Architectures_Revisited.thy` include results about the

[5] Proofs are compared using a proof ordering—that is, the multiset extension of an ordering that compares individual proof steps $s \rightarrow_R t$, $s \leftarrow_R t$, or $s \leftrightarrow_E t$.

given clause prover GC and its extension LGC with delayed inferences. The invariant-based proofs presented in this paper are found in the latter theory file.
- `FO_Ordered_Resolution_Prover_Revisited.thy` re-proves the ordered resolution prover RP refutationally complete using our framework, providing a modular alternative to the Isabelle formalization by Schlichtkrull et al. [36, 38].

The development relies heavily on Isabelle's locales [9]. 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 64, 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 $\mathbf{F}$ and $\mathbf{L}$ by types. This lightened the development in two ways. First, it relieved us from having to thread through nonemptiness conditions. Second, objects are automatically 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 describe the dynamic behavior of a calculus, are represented by the same lazy list codatatype [18] and auxiliary definitions that were used by Schlichtkrull et al.

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 (Sect. 3.3); our first attempt at verifying RP in Isabelle using the framework made it clear that the theory was not quite general enough yet to support selection functions (Example 54).

6 Conclusion

We presented a formal framework for saturation theorem proving inspired by Bachmair and Ganzinger's *Handbook* chapter [6]. 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 and can be used to verify actual provers.

To employ the framework, users must provide a statically complete saturation calculus expressible as the lifting $(\mathit{FInf}, \mathit{Red}^{\mathcal{G}})$ or $(\mathit{FInf}, \mathit{Red}^{\cap\mathcal{G}})$ of a ground calculus $(\mathit{GInf}, \mathit{Red})$, where *Red* qualifies as a redundancy criterion and $\mathcal{G}$ qualifies as a grounding function or grounding function family. The framework can be used to derive two main results:

1. After defining a well-founded ordering $\sqsupset$ or a family of well-founded orderings that capture instantiation, invoke Theorem 53 to show $(\mathit{FInf}, \mathit{Red}^{\cap\mathcal{G},\sqsupset})$ dynamically complete.
2. Based on the previous step, invoke Theorems 70 or 80 to derive the dynamic completeness of a prover architecture building on the given clause procedure, such as the Otter, iProver, DISCOUNT, or Zipperposition loop (Examples 71, 74, 81, or 82).

The framework can also help establish the static completeness of the nonground calculus. For many calculi (with the notable exceptions of constraint and hierarchic superposition), Theorems 32 or 50 can be used to lift the static completeness of $(\mathit{GInf}, \mathit{Red})$ to $(\mathit{FInf}, \mathit{Red}^{\mathcal{G}})$ or $(\mathit{FInf}, \mathit{Red}^{\cap\mathcal{G}})$.

The main missing piece of the framework is a generic treatment of clause splitting. Until recently, the only formal treatment of splitting, by Fietzke and Weidenbach [22], hard-codes both the underlying calculus and the splitting strategy. Voronkov's AVATAR architecture [42] is more flexible and yields truly impressive empirical results, but he and his collaborators left the question of AVATAR's refutational completeness open. Ebner et al. [21] recently provided an answer by introducing and instantiating a generic splitting framework based on our saturation framework.

Acknowledgements We thank Alexander Bentkamp for discussions about prover architectures for higher-order logic and for feedback from instantiating the framework, especially concerning Theorem 50. We thank Mathias Fleury and Christian Sternagel for their help with the Isabelle development. Finally, we thank Gabriel Ebner, Robert Lewis, Visa Nummelin, Dmitriy Traytel, and the anonymous reviewers 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).

Funding Open Access funding enabled and organized by Projekt DEAL.

Index

References

1. Avenhaus, J., Denzinger, J., Fuchs, M.: DISCOUNT: a system for distributed equational deduction. In: Hsiang, J. (ed.) RTA-95. LNCS, pp. 397–402. Springer, Heidelberg (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, Boston (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, Heidelberg (1990)
4. Bachmair, L., Ganzinger, H.: Associative-commutative superposition. In: Dershowitz, N., Lindenstrauss, N. (eds.) CTRS-94. LNCS, pp. 1–14. Springer, Heidelberg (1994)
5. Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. J. Log. Comput. **4**(3), 217–247 (1994)
6. 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, Cambridge (2001)
7. 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, Heidelberg (1993)
8. Bachmair, L., Ganzinger, H., Waldmann, U.: Refutational theorem proving for hierarchic first-order theories. Appl. Algebra Eng. Commun. Comput. **5**, 193–212 (1994)
9. Ballarin, C.: Locales: A module system for mathematical theories. J. Autom. Reason. **52**(2), 123–153 (2014)
10. 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. LNCS, vol. 11560, pp. 15–56. Springer, Heidelberg (2019)
11. Benanav, D.: Simultaneous paramodulation. In: Stickel, M.E. (ed.) CADE-10. LNCS, vol. 449, pp. 442–455. Springer, Heidelberg (1990)
12. Bentkamp, A., Blanchette, J., Cruanes, S., Waldmann, U.: Superposition for lambda-free higher-order logic. Log. Meth. Comput. Sci. 17(2) (2021)
13. Bentkamp, A., Blanchette, J., Tourret, S., Vukmirović, P.: Superposition for full higher-order logic. In: Platzer, A., Sutcliffe, G. (eds.) CADE-28. LNCS, pp. 396–412. Springer, Heidelberg (2021)
14. Bentkamp, A., Blanchette, J., Tourret, S., Vukmirović, P., Waldmann, U.: Superposition with lambdas. J. Autom. Reason. **65**(7), 893–940 (2021)
15. Bhayat, A.: Automated theorem proving in higher-order logic. Ph.D. thesis, University of Manchester (2021)
16. Blanchette, J., Tourret, S.: Extensions to the comprehensive framework for saturation theorem proving. Archive of Formal Proofs **2021** (2021). https://www.isa-afp.org/entries/Saturation_Framework_Extensions.html
17. 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, New York (2019)
18. Blanchette, J.C., Hölzl, J., Lochbihler, A., Panny, L., Popescu, A., Traytel, D.: Truly modular (co)datatypes for Isabelle/HOL. In: Klein, G., Gamboa, R. (eds.) ITP 2014. LNCS, vol. 8558, pp. 93–110. Springer, Heidelberg (2014)
19. 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, Heidelberg (2018)
20. Duarte, A., Korovin, K.: Implementing superposition in iProver (system description). In: IJCAR (2). LNCS, vol. 12167, pp. 388–397. Springer, Heidelberg (2020)
21. Ebner, G., Blanchette, J., Tourret, S.: A unifying splitting framework. In: Platzer, A., Sutcliffe, G. (eds.) CADE-28. LNCS, vol. 12699, pp. 344–360. Springer, Heidelberg (2021)
22. Fietzke, A., Weidenbach, C.: Labelled splitting. Ann. Math. Artif. Intell. **55**(1–2), 3–34 (2009)
23. Ganzinger, H., Stuber, J.: Superposition with equivalence reasoning and delayed clause normal form transformation. Information and Computation **199**(1–2), 3–23 (2005)
24. Hillenbrand, T., Löchner, B.: The next Waldmeister loop. In: Voronkov, A. (ed.) CADE-18. LNCS, vol. 2392, pp. 486–500. Springer, Heidelberg (2002)
25. Huet, G.P.: A mechanization of type theory. In: Nilsson, N.J. (ed.) IJCAI-73, pp. 139–146. William Kaufmann, San Mateo (1973)
26. Järvisalo, M., Biere, A., Heule, M.: Blocked clause elimination. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 129–144. Springer, Heidelberg (2010)
27. Kovács, L., Voronkov, A.: First-order theorem proving and Vampire. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 1–35. Springer, Heidelberg (2013)
28. McCune, W., Wos, L.: Otter—the CADE-13 competition incarnations. J. Autom. Reason. **18**(2), 211–220 (1997)

29. Nieuwenhuis, R., Rubio, A.: Theorem proving with ordering and equality constrained clauses. J. Symb. Comput. **19**(4), 321–351 (1995)
30. 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)
31. Nipkow, T., Klein, G.: Concrete Semantics: With Isabelle/HOL. Springer, Heidelberg (2014)
32. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer, Heidelberg (2002)
33. Nummelin, V., Bentkamp, A., Tourret, S., Vukmirović, P.: Superposition with first-class Booleans and inprocessing clausification. In: Platzer, A., Sutcliffe, G. (eds.) CADE-28. LNCS, vol. 12699, pp. 378–395. Springer, Heidelberg (2021)
34. Peltier, N.: A variant of the superposition calculus. Archive of Formal Proofs **2016**, (2016). https://www.isa-afp.org/entries/SuperCalc.shtml
35. Robinson, J.A.: A machine-oriented logic based on the resolution principle. J. ACM **12**(1), 23–41 (1965)
36. Schlichtkrull, A., Blanchette, J., Traytel, D., Waldmann, U.: Formalizing Bachmair and Ganzinger's ordered resolution prover. J. Autom. Reason. **64**(7), 1169–1195 (2020)
37. 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, New York (2019)
38. Schlichtkrull, A., Blanchette, J.C., Traytel, D., Waldmann, U.: Formalization of Bachmair and Ganzinger's ordered resolution prover. Archive of Formal Proofs, vol. 2018, (2018). https://www.isa-afp.org/entries/Ordered_Resolution_Prover.html
39. Schulz, S.: E—a brainiac theorem prover. AI Commun. **15**(2–3), 111–126 (2002)
40. Tourret, S.: A comprehensive framework for saturation theorem proving. Archive of Formal Proofs vol. 2020 (2020). https://www.isa-afp.org/entries/Saturation_Framework.shtml
41. Tourret, S., Blanchette, J.: A modular Isabelle framework for verifying saturation provers. In: Hriţcu, C., Popescu, A. (eds.) CPP 2021, pp. 224–237. ACM, New York (2021)
42. 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, Heidelberg (2014)
43. Vukmirović, P., Blanchette, J., Heule, M.J.: SAT-inspired eliminations for superposition. In: FMCAD 2021, pp. 231–240. IEEE (2021)
44. Waldmann, U.: Cancellative abelian monoids and related structures in refutational theorem proving (part I). J. Symb. Comput. **33**(6), 777–829 (2002)
45. Waldmann, U., Tourret, S., Robillard, S., Blanchette, J.: A comprehensive framework for saturation theorem proving. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020, Part I. LNCS, vol. 12166, pp. 316–334. Springer, Heidelberg (2020)
46. 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, Cambridge (2001)