A Unifying Splitting Framework (Technical Report)

Gabriel Ebner[1] (✉), Jasmin Blanchette[1,2,3], and Sophie Tourret[2,3]

[1] Vrije Universiteit Amsterdam, Amsterdam, the Netherlands
{g.e.ebner,j.c.blanchette}@vu.nl
[2] Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
{jasmin.blanchette,sophie.tourret}@inria.fr
[3] Max-Planck-Institut für Informatik, Saarland Informatics Campus, Saarbrücken, Germany
{jasmin.blanchette,stourret}@mpi-inf.mpg.de

Abstract. AVATAR is an elegant and effective way to split clauses in a saturation prover using a SAT solver. But is it refutationally complete? And how does it relate to other splitting architectures? To answer these questions, we present a unifying framework that extends a saturation calculus (e.g., superposition) with splitting and embeds the result in a prover guided by a SAT solver. The framework also allows us to study locking, a subsumption-like mechanism based on the current propositional model. Various architectures are instances of the framework, including AVATAR, labeled splitting, and SMT with quantifiers.

1 Introduction

One of the great strengths of saturation calculi such as resolution [24] and superposition [1] is that they avoid case distinctions. Derived clauses hold unconditionally, and the prover can stop as soon as it derives the empty clause, without having to backtrack. The drawback is that these calculi often generate long, unwieldy clauses that slow down the prover. A remedy is to partition the search space by splitting a multiple-literal clause $C_1 \vee \cdots \vee C_n$ into subclauses $C_1, \ldots, C_n$ that share no variables. Splitting approaches include splitting with backtracking [31], splitting without backtracking [23], labeled splitting [11], and AVATAR [28].

The SAT-based AVATAR architecture is of particular interest because it is so successful. Voronkov reported that an AVATAR-enabled Vampire could solve 421 TPTP [27] problems that had never been solved before by any system [28, Sect. 9], a mind-boggling number. AVATAR works well in combination with the superposition calculus because it combines superposition's strong equality reasoning with the SAT solver's strong clausal reasoning. It is also appealing theoretically, because it gracefully generalizes traditional saturation provers and yet degenerates to a SAT solver if the problem is propositional.

Example 1. To illustrate the approach, we follow the key steps of an AVATAR-enabled resolution prover on the initial clause set containing $\neg\mathsf{p}(\mathsf{a})$, $\neg\mathsf{q}(z, z)$, and

$\mathsf{p}(x) \vee \mathsf{q}(y, \mathsf{b})$. The disjunction can be split into $\mathsf{p}(x) \leftarrow \{[\mathsf{p}(x)]\}$ and $\mathsf{q}(y, \mathsf{b}) \leftarrow \{[\mathsf{q}(y, \mathsf{b})]\}$, where $C \leftarrow \{[C]\}$ indicates that the clause C is enabled only in models in which the associated propositional variable $[C]$ is true. A SAT solver is then run to choose a model $\mathcal{J}$ of $[\mathsf{p}(x)] \vee [\mathsf{q}(y, \mathsf{b})]$. Suppose $\mathcal{J}$ makes $[\mathsf{p}(x)]$ true and $[\mathsf{q}(y, \mathsf{b})]$ false. Then resolution of $\mathsf{p}(x) \leftarrow \{[\mathsf{p}(x)]\}$ with $\neg\mathsf{p}(\mathsf{a})$ produces $\bot \leftarrow \{[\mathsf{p}(x)]\}$, which closes the branch. Next, the SAT solver makes the right disjunct true, and resolving $\mathsf{q}(y, \mathsf{b}) \leftarrow \{[\mathsf{q}(y, \mathsf{b})]\}$ with $\neg\mathsf{q}(z, z)$ yields $\bot \leftarrow \{[\mathsf{q}(y, \mathsf{b})]\}$. The SAT solver then reports "unsatisfiable," concluding the refutation.

What about refutational completeness? Far from being a purely theoretical concern, establishing completeness—or finding counterexamples—could yield insights into splitting and maybe lead to an even stronger AVATAR. Before we can answer this open question, we must mathematize splitting. Our starting point is the *saturation framework* by Waldmann, Tourret, Robillard, and Blanchette [29], based on Bachmair and Ganzinger [2]. It covers a wide array of techniques, but "the main missing piece of the framework is a generic treatment of clause splitting" [29, p. 332]. We provide that missing piece, in the form of a *splitting framework*, and use it to show the completeness of an AVATAR-like architecture. The framework is currently a pen-and-paper creature; a formalization using Isabelle/HOL [19] is underway.

Our framework has five layers, linked by refinement. The first layer consists of a *base calculus*, such as resolution or superposition. It must be presentable as an inference system and a redundancy criterion, as required by the saturation framework, and it must be refutationally complete.

From a base calculus, our framework can be used to derive the second layer, which we call the *splitting calculus* (Sect. 3). This extends the base calculus with splitting and inherits the base's completeness. It works on A-clauses or A-formulas $C \leftarrow A$, where C is a base clause or formula and A is a set of propositional literals, called assertions.

Using the saturation framework, we can prove the dynamic completeness of an abstract prover, formulated as a transition system, that implements the splitting calculus. However, this ignores a vital component of AVATAR: the SAT solver. AVATAR considers only inferences involving A-formulas whose assertions are true in the current propositional model. The role of the third layer is to reflect this behavior. A *model-guided prover* operates on states of the form $(\mathcal{J},\mathcal{N})$, where $\mathcal{J}$ is a propositional model and $\mathcal{N}$ is a set of A-formulas (Sect. 4).

The fourth layer introduces AVATAR's *locking* mechanism (Sect. 5). With locking, an A-formula $D \leftarrow B$ can be temporarily disabled by another A-formula $C \leftarrow A$ if C subsumes D, even if $A \not\subseteq B$. Here we make a first discovery: AVATAR-style locking compromises completeness and must be curtailed.

Finally, the fifth layer is an *AVATAR-based prover* (Sect. 6). This refines the locking model-guided prover of the fourth layer with the given clause procedure, which saturates an A-formula set by distinguishing between active and passive A-formulas. Here we make another discovery: Selecting A-formulas fairly is not enough to guarantee completeness. We need a stronger criterion.

There are also implications for other architectures. In a hypothetical tête-à-tête with the designers of labeled splitting, they might gently point out that by pioneering the use of a propositional model, including locking, they almost invented AVATAR themselves. Likewise, developers of SMT solvers might be tempted to claim that Voronkov merely reinvented SMT. To investigate such questions, we apply our framework to splitting without backtracking, labeled splitting, and SMT with quantifiers (Sect. 7). This gives us a solid basis for comparison as well as some new theoretical results.

2 Preliminaries

Our framework is parameterized by abstract notions of formulas, consequence relations, inferences, and redundancy. We largely follow the conventions of Waldmann et al. [29]. A-formulas generalize Voronkov's A-clauses [28].

2.1 Formulas

A set $\mathbf{F}$ of *formulas*, ranged over by $C, D \in \mathbf{F}$, is a set that contains a distinguished element $\bot$ denoting falsehood. A *consequence relation* $\models$ over $\mathbf{F}$ is a relation $\models \subseteq (\mathcal{P}(\mathbf{F}))^2$ with the following properties for all sets $M, N, P, Q \subseteq \mathbf{F}$ and formulas $C, D \in \mathbf{F}$:

(D1) $\{\bot\} \models \emptyset$;
(D2) $\{C\} \models \{C\}$;
(D3) if $M \subseteq N$ and $P \subseteq Q$, then $M \models P$ implies $N \models Q$;
(D4) if $M \models P$ and $N \models Q \cup \{C\}$ for every $C \in M$ and $N \cup \{D\} \models Q$ for every $D \in P$, then $N \models Q$.

The intended interpretation of $M \models N$ is $\bigwedge M \longrightarrow \bigvee N$. Notice the disjunction on the right-hand side. A consequence relation $\models$ is called *compact* if for any M, N such that $M \models N$ there exist finite sets $M' \subseteq M$ and $N' \subseteq N$ such that $M' \models N'$.

For their saturation framework, Waldmann et al. instead consider a conjunctive version of the consequence relation, with different axioms. Given a consequence relation $\models$, we can easily obtain a (conjunctive) consequence relation $\models'$ in the sense of the saturation framework by defining $M \models' N$ if and only if $M \models \{C\}$ for all $C \in N$. The two versions differ only when the right-hand side is not a singleton.

The $\models$ notation can be extended to allow negation on either side. Let $\mathbf{F}_\sim$ be defined as $\mathbf{F} \uplus \{\sim C \mid C \in \mathbf{F}_\sim\}$ such that $\sim\sim C = C$. Given $M, N \subseteq \mathbf{F}_\sim$, we have $M \models N$ if and only if

$$\begin{aligned} &\{C \in \mathbf{F} \mid \quad C \in M\} \cup \{C \in \mathbf{F} \mid \sim C \in N\} \\ \models\ &\{C \in \mathbf{F} \mid \sim C \in M\} \cup \{C \in \mathbf{F} \mid \quad C \in N\} \end{aligned}$$

We write $M \models\!\models N$ for the conjunction of $M \models N$ and $N \models M$.

Following the saturation framework [29, p. 318], we distinguish between the consequence relation $\models$ used for stating refutational completeness and the consequence relation $\models\!\!\!\approx$ used for stating soundness. For example, $\models$ could be entailment for first-order logic with equality, whereas $\models\!\!\!\approx$ could also draw on linear arithmetic, or interpret Skolem symbols so as to make skolemization sound. Normally $\models \subseteq \models\!\!\!\approx$, but this is not required. We require that $\models\!\!\!\approx$ is compact.

Example 2. In clausal first-order logic with equality, as implemented in superposition provers, the formulas in $\mathbf{F}$ consist of clauses over a signature Σ. Each clause C is a finite multiset of literals $L_1, \ldots, L_n$ written $C = L_1 \vee \cdots \vee L_n$. Each literal L is either an atom or its negation ($\neg$), and each atom is an unoriented equation $s \approx t$. We define the consequence relation $\models$ by letting $M \models N$ if and only if every first-order Σ-interpretation that satisfies all clauses in M also satisfies at least one clause in N.

2.2 Calculi and Derivations

A refutational calculus combines a set of inferences, which are a priori mandatory, and a redundancy criterion, which identifies inferences that need not be performed after all as well as formulas that can be deleted.

Let $\mathbf{F}$ be a set of formulas equipped with $\bot$. An $\mathbf{F}$-*inference* ι is a tuple $(C_n, \ldots, C_1, D) \in \mathbf{F}^{n+1}$. The formulas $C_n, \ldots, C_1$ are the *premises*, and D is the *conclusion*. Define $\mathit{prems}(\iota) = \{C_n, \ldots, C_1\}$ and $\mathit{concl}(\iota) = \{D\}$. An inference ι is *sound* w.r.t. $\models\!\!\!\approx$ if $\mathit{prems}(\iota) \models\!\!\!\approx \mathit{concl}(\iota)$. An *inference system* Inf is a set of $\mathbf{F}$-inferences.

Given $N \subseteq \mathbf{F}$, we let $\mathit{Inf}(N)$ denote the set of all inferences in Inf whose premises are included in N, and $\mathit{Inf}(N, M) = \mathit{Inf}(N \cup M) \setminus \mathit{Inf}(N \setminus M)$ for the set of all inferences in Inf such that one or more premises are in M and the remaining premises are in N.

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 $\mathit{Red}_\mathrm{F} : \mathcal{P}(\mathbf{F}) \to \mathcal{P}(\mathbf{F})$ enjoy the following properties for all sets $M, N \subseteq \mathbf{F}$:

(R1) if $N \models \{\bot\}$, then $N \setminus \mathit{Red}_\mathrm{F}(N) \models \{\bot\}$;
(R2) if $M \subseteq N$, then $\mathit{Red}_\mathrm{F}(M) \subseteq \mathit{Red}_\mathrm{F}(N)$ and $\mathit{Red}_\mathrm{I}(M) \subseteq \mathit{Red}_\mathrm{I}(N)$;
(R3) if $M \subseteq \mathit{Red}_\mathrm{F}(N)$, then $\mathit{Red}_\mathrm{F}(N) \subseteq \mathit{Red}_\mathrm{F}(N \setminus M)$ and $\mathit{Red}_\mathrm{I}(N) \subseteq \mathit{Red}_\mathrm{I}(N \setminus M)$;
(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 said to be *redundant* w.r.t. N. Red_I indicates which inferences must be performed, whereas Red_F justifies the deletion of formulas deemed useless. The above properties make the passage from static to dynamic completeness possible: (R1) ensures that deleting a redundant formula preserves a set's inconsistency, so as not too lose refutations; (R2) and (R3) ensure that arbitrary formulas can be added and redundant formulas can be deleted by the prover; and (R4) ensures that adding an inference's conclusion to the formula set makes the inference redundant.

A pair $(\mathit{Inf}, \mathit{Red})$ forms a *calculus*. A set $N \subseteq \mathbf{F}$ is *saturated* w.r.t. Inf and Red_I if $\mathit{Inf}(N) \subseteq \mathit{Red}_\mathrm{I}(N)$. The calculus $(\mathit{Inf}, \mathit{Red})$ is *statically* (*refutationally*) *complete* (w.r.t. $\models$) if for every set $N \subseteq \mathbf{F}$ that is saturated w.r.t. Inf and Red_I and such that $N \models \{\bot\}$, we have $\bot \in N$.

Lemma 3. *Assume that the calculus $(\mathit{Inf}, \mathit{Red})$ is statically complete. Then $\bot \notin \mathit{Red}_\mathrm{F}(N)$ for every $N \subseteq \mathbf{F}$.*

Proof. By (R2), it suffices to show $\bot \notin \mathit{Red}_\mathrm{F}(\mathbf{F})$. Clearly, by (D2) and (D3) $\mathbf{F} \models \{\bot\}$. Thus by (R1), $\mathbf{F} \setminus \mathit{Red}_\mathrm{F}(\mathbf{F}) \models \{\bot\}$. Moreover by (R3) and (R4), $\mathbf{F} \setminus \mathit{Red}_\mathrm{F}(\mathbf{F})$ is saturated. Hence, since $(\mathit{Inf}, \mathit{Red})$ is statically complete, $\bot \in \mathbf{F} \setminus \mathit{Red}_\mathrm{F}(\mathbf{F})$. Therefore, $\bot \notin \mathit{Red}_\mathrm{F}(\mathbf{F})$. □

Remark 4. Given a redundancy criterion $(\mathit{Red}_\mathrm{I}, \mathit{Red}_\mathrm{F})$, where $\bot \notin \mathit{Red}_\mathrm{F}(\mathbf{F})$, we can make it stricter as follows. Define Red'_I such that $\iota \in \mathit{Red}'_\mathrm{I}(N)$ if and only if either $\iota \in \mathit{Red}_\mathrm{I}(N)$ or $\bot \in N$. Define Red'_F such that $C \in \mathit{Red}'_\mathrm{F}(N)$ if and only if either $C \in \mathit{Red}_\mathrm{F}(N)$ or else both $\bot \in N$ and $C \neq \bot$. Obviously, $\mathit{Red}' = (\mathit{Red}'_\mathrm{I}, \mathit{Red}'_\mathrm{F})$ is a redundancy criterion. Moreover, if N is saturated w.r.t. Inf and Red_I, then N is saturated w.r.t. Inf and Red'_I, and if the calculus $(\mathit{Inf}, \mathit{Red})$ is statically complete, then $(\mathit{Inf}, \mathit{Red}')$ is also statically complete. (In the last case, the condition $\bot \notin \mathit{Red}_\mathrm{F}(\mathbf{F})$ holds by Lemma 3.)

A *sequence* $(x_i)_i$ over a set X is a function from $\mathbb{N}$ to X that maps each $i \in \mathbb{N}$ to $x_i \in X$. Let $(X_i)_i$ be a sequence of sets. Its *limit inferior* is $X_\infty = \liminf_{j\to\infty} X_j = \bigcup_i \bigcap_{j\geq i} X_j$, and its *limit superior* is $X^\infty = \limsup_{j\to\infty} X_j = \bigcap_i \bigcup_{j\geq i} X_j$. The elements of X_∞ are called *persistent*. A sequence $(N_i)_i$ of sets of $\mathbf{F}$-formulas is *weakly fair* w.r.t. Inf and Red_I if $\mathit{Inf}(N_\infty) \subseteq \bigcup_i \mathit{Red}_\mathrm{I}(N_i)$ and *strongly fair* if $\limsup_{i\to\infty} \mathit{Inf}(N_i) \subseteq \bigcup_i \mathit{Red}_\mathrm{I}(N_i)$. Weak fairness requires that all inferences possible from some index i and ever after eventually be performed or become redundant for another reason. Strong fairness requires the same from all inferences that are possible infinitely often, even if not continuously so. Both can be used to ensure that some limit is saturated.

Given a relation $\rhd \subseteq X^2$, a $\rhd$-*derivation* is an sequence of X elements such that $x_i \rhd x_{i+1}$ for every i. Finite runs can be extended to derivations by repeating the final state infinitely. We must then ensure that $\rhd$ supports such stuttering from states that would otherwise have no out-transitions. Abusing language, and departing from the saturation framework, we will say that a derivation $(x_i)_i$ *terminates* if $x_i = x_{i+1} = \cdots$ for some index i.

Let $\rhd_{\mathit{Red}_\mathrm{F}} \subseteq (\mathcal{P}(\mathbf{F}))^2$ be the relation such that $M \rhd_{\mathit{Red}_\mathrm{F}} N$ if and only if $M \setminus N \subseteq \mathit{Red}_\mathrm{F}(N)$. Note that it is reflexive and hence supports stuttering. The relation is also transitive due to (R3). We could additionally require soundness ($M \not\approx N$) or consistency preservation ($M \not\approx \{\bot\}$ implies $N \not\approx \{\bot\}$), but this is unnecessary for proving completeness. The calculus $(\mathit{Inf}, \mathit{Red})$ is *dynamically* (*refutationally*) *complete* (w.r.t. $\models$) if for every $\rhd_{\mathit{Red}_\mathrm{F}}$-derivation $(N_i)_i$ that is weakly fair w.r.t. Inf and Red_I and such that $N_0 \models \{\bot\}$, we have $\bot \in N_i$ for some i.

2.3 A-Formulas

We fix throughout a countable set $\mathbf{V}$ of *propositional variables* $\mathsf{v}_0, \mathsf{v}_1, \ldots$. For each $\mathsf{v} \in \mathbf{V}$, let $\neg\mathsf{v} \in \neg\mathbf{V}$ denote its negation, with $\neg\neg\mathsf{v} = \mathsf{v}$. We assume that a formula $\mathit{fml}(\mathsf{v}) \in \mathbf{F}$ is associated with each propositional variable $\mathsf{v} \in \mathbf{V}$. Intuitively, v approximates $\mathit{fml}(\mathsf{v})$ at the propositional level. This definition is extended so that $\mathit{fml}(\neg\mathsf{v}) = {\sim}\mathit{fml}(\mathsf{v})$.

A propositional literal, or *assertion*, $a \in \mathbf{A} = \mathbf{V} \cup \neg\mathbf{V}$ over $\mathbf{V}$ is either a propositional variable v or its negation $\neg\mathsf{v}$.

A *propositional interpretation* $\mathcal{J} \subseteq \mathbf{A}$ is a set of assertions such that for every $\mathsf{v} \in \mathbf{V}$, exactly one of $\mathsf{v} \in \mathcal{J}$ and $\neg\mathsf{v} \in \mathcal{J}$ holds.

Remark 5. Our propositional interpretations are always total. We could also consider partial interpretations, that is, $\mathcal{J} \subseteq \mathbf{A}$ such that at most of $\mathsf{v} \in \mathcal{J}$ and $\neg\mathsf{v} \in \mathcal{J}$ holds for every $\mathsf{v} \in V$. But this is not necessary, because partial interpretations can always be simulated using total ones: For any variable v in the partial interpretation we use two variables v^+ and v^- in the total interpretation, and interpret v^+ as true if v is true, as well as v^- as true if v is false. By adding the propositional clause $\bot \leftarrow \{\mathsf{v}^-, \mathsf{v}^+\}$, every total model of the translated A-formulas corresponds to a partial model of the original A-formulas.

An *A-formula* over a set $\mathbf{F}$ of *base formulas* and an assertion set $\mathbf{A}$ is a pair $\mathcal{C} = (C, A) \in \mathbf{AF} = \mathbf{F} \times \mathcal{P}_{\mathrm{fin}}(\mathbf{A})$, written $C \leftarrow A$, where C is a formula and A is a finite set of assertions $\{a_1, \ldots, a_n\}$ understood as an implication $a_1 \wedge \cdots \wedge a_n \longrightarrow C$. We identify $C \leftarrow \emptyset$ with C and define the projections $\lfloor C \leftarrow A \rfloor = C$ and $\lfloor (C_n \leftarrow A_n, \ldots, C_0 \leftarrow A_0) \rfloor = (C_n, \ldots, C_0)$. Moreover, $\mathcal{N}_\bot$ is the set consisting of all A-formulas of the form $\bot \leftarrow A \in \mathcal{N}$, where $A \in \mathcal{P}_{\mathrm{fin}}(\mathbf{A})$. Since $\bot \leftarrow \{a_1, \ldots, a_n\}$ can be read as $\neg a_1 \vee \cdots \vee \neg a_n$, we call such A-formulas *propositional clauses*. Note the use of calligraphic letters (e.g., $\mathcal{C}, \mathcal{N}$) to range over A-formulas and sets of A-formulas.

Model-guided provers only consider provers whose assertions are true in the current interpretation. Thus we say that an A-formula $C \leftarrow A \in \mathbf{AF}$ is *enabled* in a propositional interpretation $\mathcal{J}$ if $A \subseteq \mathcal{J}$. A set of A-formulas is *enabled* in $\mathcal{J}$ if all of its members are enabled in $\mathcal{J}$. Given an A-formula set $\mathcal{N} \subseteq \mathbf{AF}$, the *enabled projection* $\mathcal{N}_\mathcal{J} \subseteq \lfloor \mathcal{N} \rfloor$ consists of the projections $\lfloor \mathcal{C} \rfloor$ of all A-formulas $\mathcal{C}$ enabled in $\mathcal{J}$. Analogously, the *enabled projection* $\mathit{Inf}_\mathcal{J} \subseteq \lfloor \mathit{Inf} \rfloor$ of a set Inf of $\mathbf{AF}$-inferences consists of the projections $\lfloor \iota \rfloor$ of all inferences $\iota \in \mathit{Inf}$ whose premises are all enabled in $\mathcal{J}$.

A propositional interpretation $\mathcal{J}$ is a *propositional model* of $\mathcal{N}_\bot$, written $\mathcal{J} \models \mathcal{N}_\bot$, if $\bot \notin (\mathcal{N}_\bot)_\mathcal{J}$. Moreover, we write $\mathcal{J} \mathrel{|\!\approx} \mathcal{N}_\bot$ if $\bot \notin (\mathcal{N}_\bot)_\mathcal{J}$ or $\mathit{fml}(\mathcal{J}) \mathrel{|\!\approx} \{\bot\}$. A set $\mathcal{N}_\bot$ is *propositionally satisfiable* if there exists an interpretation $\mathcal{J}$ such that $\mathcal{J} \models \mathcal{N}_\bot$. In contrast with consequence relations, propositional modelhood $\models$ interprets the set $\mathcal{N}_\bot$ conjunctively: $\mathcal{J} \models \mathcal{N}_\bot$ is understood as $\mathcal{J} \models \bigwedge \mathcal{N}_\bot$.

Next, we lift $\models$ and $\mathrel{|\!\approx}$ from $\mathcal{P}(\mathbf{F})$ to $\mathcal{P}(\mathbf{AF})$: $\mathcal{M} \models \mathcal{N}$ if and only if $\mathcal{M}_\mathcal{J} \models \lfloor \mathcal{N} \rfloor$ for every $\mathcal{J}$ in which $\mathcal{N}$ is enabled, and $\mathcal{M} \mathrel{|\!\approx} \mathcal{N}$ if and only if $\mathit{fml}(\mathcal{J}) \cup \mathcal{M}_\mathcal{J} \mathrel{|\!\approx} \lfloor \mathcal{N} \rfloor$ for every $\mathcal{J}$ in which $\mathcal{N}$ is enabled. The consequence relation $\models$ is used for the completeness of the splitting prover and only captures what inferences such a

prover must perform. In contrast, $\mathrel{|\!\approx}$ captures a stronger semantics: for example, thanks to $\mathit{fml}(\mathcal{J})$ among the premises for $\mathrel{|\!\approx}$, the A-formula $\mathit{fml}(a) \leftarrow \{a\}$ is always a $\mathrel{|\!\approx}$-tautology. Also note that $\models \subseteq \mathrel{|\!\approx}$ on sets that contain exclusively propositional clauses (even when $\models \not\subseteq \mathrel{|\!\approx}$ on $\mathcal{P}(\mathbf{F})$).

Lemma 6. *The relations $\models$ and $\mathrel{|\!\approx}$ on $\mathcal{P}(\mathbf{AF})$ are consequence relations.*

Proof. We consider only $\mathrel{|\!\approx}$. The proof for $\models$ is analogous. For (D1), we need to show $\mathit{fml}(\mathcal{J}) \cup \{\bot\} \mathrel{|\!\approx} \emptyset$ for all $\mathcal{J}$ because $\emptyset$ is always enabled. This follows from (D1) and (D3). For (D2), we need to show $\mathit{fml}(\mathcal{J}) \cup \{C \leftarrow A\}_{\mathcal{J}} \mathrel{|\!\approx} \lfloor \{C \leftarrow A\} \rfloor$, assuming $C \leftarrow A$ is enabled in $\mathcal{J}$. Hence it suffices to show $\mathit{fml}(\mathcal{J}) \cup \{C\} \mathrel{|\!\approx} \{C\}$, which follows from (D2) and (D3). For (D3), it suffices to show $\mathit{fml}(\mathcal{J}) \cup N_{\mathcal{J}} \mathrel{|\!\approx} \lfloor Q \rfloor$ assuming that Q is enabled in $\mathcal{J}$, and $\mathit{fml}(\mathcal{J}) \cup M_{\mathcal{J}} \mathrel{|\!\approx} \lfloor P \rfloor$ for any $M \subseteq N$ and $P \subseteq Q$. This follows from (D3) and monotonicity of $\lfloor\ \rfloor$ and $(\ )_{\mathcal{J}}$. For (D4), we show $\mathit{fml}(\mathcal{J}) \cup N_{\mathcal{J}} \mathrel{|\!\approx} \lfloor Q \rfloor$ assuming Q is enabled in $\mathcal{J}$ and $M \mathrel{|\!\approx} P$ and $N \mathrel{|\!\approx} Q \cup \{C\}$ for all $C \in M$ and $N \cup \{D\} \mathrel{|\!\approx} Q$ for all $D \in P$. If P is enabled in $\mathcal{J}$, we have $\mathit{fml}(\mathcal{J}) \cup M_{\mathcal{J}} \mathrel{|\!\approx} \lfloor P \rfloor$ and the claim follows directly from (D4). Otherwise, if P is disabled in $\mathcal{J}$, then there exists a $D \in N$ that is disabled in $\mathcal{J}$. In this case, the assumption $N \cup \{D\} \mathrel{|\!\approx} Q$ implies the claim since $(N \cup \{D\})_{\mathcal{J}} = N_{\mathcal{J}}$. □

Lemma 7. *The two versions of $\models$ coincide on $\mathbf{F}$-formulas, and similarly for $\mathrel{|\!\approx}$.*

Proof. The first property is obvious. For the second property, the argument is as follows. Let $M, N \subseteq \mathbf{F}$. Then $M \mathrel{|\!\approx}_{\mathbf{F}} N$ if and only if $M \mathrel{|\!\approx}_{\mathbf{AF}} N$, where the subscripts should be self-explanatory. First assume that $M \mathrel{|\!\approx}_{\mathbf{F}} N$. Then clearly $\mathit{fml}(\mathcal{J}) \cup M \mathrel{|\!\approx}_{\mathbf{AF}} N$ for any $\mathcal{J}$ by (D3). The other direction, however, requires compactness. Assuming $M \mathrel{|\!\approx}_{\mathbf{AF}} N$, we have $\mathit{fml}(\mathcal{J}) \cup M \mathrel{|\!\approx}_{\mathbf{F}} N$ for every $\mathcal{J}$. Compactness of $\mathrel{|\!\approx}_{\mathbf{F}}$ allows us to restrict to a finite set of variables $V_n = \{\mathsf{v}_1, \ldots, \mathsf{v}_n\}$ such that $\mathit{fml}(\mathcal{J}) \cup M \mathrel{|\!\approx}_{\mathbf{F}} N$ for any $\mathcal{J} \subseteq V \cup \neg V$ which contains either v or $\neg\mathsf{v}$ for any $\mathsf{v} \in V$. The conclusion then follows by induction on n. We have $\mathit{fml}(\mathcal{J}') \cup \{\mathit{fml}(\mathsf{v}_n)\} \cup M \mathrel{|\!\approx}_{\mathbf{F}} N$ and $\mathit{fml}(\mathcal{J}') \cup M \mathrel{|\!\approx}_{\mathbf{F}} N \cup \{\mathit{fml}(\mathsf{v}_n)\}$. We then get $\mathit{fml}(\mathcal{J}') \cup M \mathrel{|\!\approx}_{\mathbf{F}} N$ by (D4) and $M \mathrel{|\!\approx}_{\mathbf{F}} N$ by the induction hypothesis. □

Given a formula $C \in \mathbf{F}_{\sim}$, let $\mathit{asn}(C)$ denote the set of assertions $a \in \mathbf{A}$ such that $\{\mathit{fml}(a)\} \mathrel{|\!\approx\!|} \{C\}$. Normally, we would make sure that $\mathit{asn}(C)$ is nonempty for every formula C. Given $a \in \mathit{asn}(C)$, observe that if $a \in \mathit{asn}(D)$, then $\{C\} \mathrel{|\!\approx\!|} \{D\}$, and if $\neg a \in \mathit{asn}(D)$, then $\{C\} \mathrel{|\!\approx\!|} \{\sim D\}$.

Example 8. In the original description of AVATAR [28], the connection between first-order clauses and assertions takes the form of a function $[\ ] : \mathbf{F} \to \mathbf{A}$. The encoding is such that $[\neg C] = \neg[C]$ for every ground (i.e., variable-free) unit clause C and $[C] = [D]$ if and only if C is syntactically equal to D up to variable renaming. This can be supported in our framework by letting $\mathit{fml}(\mathsf{v}) = C$ for some C such that $[C] = \mathsf{v}$, for every propositional variable v.

A different encoding is used to exploit the theories of an SMT solver [4]. With a notion of $\mathrel{|\!\approx}$-entailment that gives a suitable meaning to Skolem symbols, we can go further and have $[\neg C(\mathsf{sk}_{\neg C(x)})] = \neg[C(x)]$. Even if the superposition prover considers $\mathsf{sk}_{\neg C(x)}$ an uninterpreted symbol (according to $\models$), the SAT

or SMT solver can safely prune the search space by assuming that $C(x)$ and $\neg C(\mathsf{sk}_{\neg C(x)})$ are mutually exclusive (according to $\not\approx$).

3 Splitting Calculi

Let $\mathbf{F}$ be a set of base formulas equipped with $\bot$, $\models$, and $\not\approx$. The consequence relation $\not\approx$ is assumed to be nontrivial: (D5) $\emptyset \not\approx \emptyset$. Let $\mathbf{A}$ be a set of assertions over $\mathbf{V}$ and $\mathbf{AF}$ be the set of A-formulas over $\mathbf{F}$ and $\mathbf{A}$. Let $(\mathit{FInf}, \mathit{FRed})$ be a base calculus for $\mathbf{F}$-formulas, where FRed is a redundancy criterion that additionally satisfies

(R5) $\mathit{Inf}(\mathbf{F}, \mathit{Red}_{\mathrm{F}}(N)) \subseteq \mathit{Red}_{\mathrm{I}}(N)$ for every $N \subseteq \mathbf{F}$;
(R6) $\bot \notin \mathit{FRed}_{\mathrm{F}}(N)$ for every $N \subseteq \mathbf{F}$;
(R7) $C \in \mathit{FRed}_{\mathrm{F}}(\{\bot\})$ for every $C \neq \bot$.

These requirements can easily be met by a well-designed redundancy criterion. Requirement (R5) is called "reducedness" by Waldmann et al. [30, Sect. 2.3]. Requirement (R6) must hold of any complete calculus (Lemma 3), and (R7) can be made without loss of generality (Remark 4). Bachmair and Ganzinger's redundancy criterion for superposition [1, Sect. 4.3] meets (R1)–(R7).

In this section, we will define the *splitting calculus* $(\mathit{SInf}, \mathit{SRed})$ induced by the base calculus. An input problem is specified as a finite set of $\mathbf{F}$-formulas, which are regarded as A-formulas with no assertions. We will show that the splitting calculus is sound w.r.t. $\not\approx$ and that it is statically and dynamically complete w.r.t. $\models$. We will also show that it meets a pair of weaker, "local completeness" criteria that capture the switching of propositional models that characterizes most splitting architectures.

3.1 The Inference Rules

We start with the mandatory inference rules.

Definition 9. The *splitting inference system* SInf consists of all instances of the following two rules:

$$\frac{(C_i \leftarrow A_i)_{i=1}^n}{D \leftarrow A_1 \cup \dots \cup A_n}\ \textsc{Base} \qquad\qquad \frac{(\bot \leftarrow A_i)_{i=1}^n}{\bot}\ \textsc{Unsat}$$

For Base, the side condition is $(C_n, \dots, C_1, D) \in \mathit{FInf}$. For Unsat, the side condition is that $\{\bot \leftarrow A_1, \dots, \bot \leftarrow A_n\}$ is propositionally unsatisfiable.

In addition, the following optional inference rules can be used:

$$\frac{C \leftarrow A}{\bot \leftarrow \{\neg a_1, \dots, \neg a_n\} \cup A \quad (C_i \leftarrow \{a_i\})_{i=1}^n}\ \textsc{Split}$$

$$\frac{(\bot \leftarrow A_i)_{i=1}^n \quad C \leftarrow A}{(\bot \leftarrow A_i)_{i=1}^n} \textsc{Collect} \qquad \frac{(\bot \leftarrow A_i)_{i=1}^n \quad C \leftarrow A \cup B}{(\bot \leftarrow A_i)_{i=1}^n \quad C \leftarrow A} \textsc{Trim}$$

$$\frac{(\bot \leftarrow A_i)_{i=1}^n}{\bot} \textsc{StrongUnsat} \qquad \frac{C \leftarrow A}{\bot \leftarrow \{\neg a\} \cup A} \textsc{Approx} \qquad \frac{}{C \leftarrow A} \textsc{Tauto}$$

The following side conditions apply. For Split: $C \neq \bot$ is splittable into $C_1, \ldots, C_n$ and $a_i \in \mathit{asn}(C_i)$ for each i. A formula C is *splittable* into two or more formulas $C_1, \ldots, C_n$ if $\{C\} \not\approx \{C_1, \ldots, C_n\}$ and $C \in \mathit{FRed}_{\mathrm{F}}(\{C_i\})$ for each i. For Collect: $C \neq \bot$ and $\{\bot \leftarrow A_i\}_{i=1}^n \not\approx \{\bot \leftarrow A\}$. For Trim: $C \neq \bot$ and $\{\bot \leftarrow A_i\}_{i=1}^n \cup \{\bot \leftarrow A\} \not\approx \{\bot \leftarrow B\}$. For StrongUnsat: $\{\bot \leftarrow A_i\}_{i=1}^n \not\approx \{\bot\}$. For Approx: $a \in \mathit{asn}(C)$. For Tauto: $\not\approx \{C \leftarrow A\}$.

The three rules identified by double bars are simplifications; they replace their premises with their conclusions in the current A-formula set. The premises' removal is justified by $\mathit{SRed}_{\mathrm{F}}$, defined below.

The Split rule performs an n-way case split on C. Each case C_i is approximated by an assertion a_i. The first conclusion expresses that the case distinction is exhaustive. The n other conclusions assume C_i if its approximation a_i is true. In a clausal prover, typically $C = C_1 \vee \cdots \vee C_n$, where the subclauses C_i have mutually disjoint sets of variables and form a maximal split, with each ground subclause C_i consisting of a single literal.

Collect removes A-formulas whose assertions cannot be satisfied by any model of the propositional clauses—a form of garbage collection. Similarly, Trim removes assertions that are entailed by existing propositional clauses.

StrongUnsat is a variant of Unsat that uses $\not\approx$ instead of $\not\models$. A splitting prover may choose to apply StrongUnsat if desired, but only Unsat is necessary for completeness. In practice, $\not\approx$-entailment can be much more expensive to decide, or even be undecidable. Thus, a splitting prover could invoke an SMT solver [4] ($\not\approx$) with a time limit, falling back on a SAT solver ($\not\models$) if necessary.

Approx can be used to make any derived A-formula visible to $\not\approx$. It is similar to a one-way split. Tauto, which asserts a $\not\approx$-tautology, allows communication in the other direction, from the SAT solver to the calculus.

Example 10. Suppose the base calculus is first-order resolution [2] and the initial clauses are $\neg\mathsf{p}(\mathsf{a})$, $\neg\mathsf{q}(z, z)$, and $\mathsf{p}(x) \vee \mathsf{q}(y, \mathsf{b})$, as in Example 1. Split replaces the last clause by $\bot \leftarrow \{\neg\mathsf{v}_0, \neg\mathsf{v}_1\}$, $\mathsf{p}(x) \leftarrow \{\mathsf{v}_0\}$, and $\mathsf{q}(y, \mathsf{b}) \leftarrow \{\mathsf{v}_1\}$. Two Base inferences then generate $\bot \leftarrow \{\mathsf{v}_0\}$ and $\bot \leftarrow \{\mathsf{v}_1\}$. Finally, Unsat generates $\bot$.

Example 11. Consider a splitting calculus obeying the AVATAR conventions of Example 8. When splitting on $C(x) \vee D(y)$, after closing the $C(x)$ case, we can assume that $C(x)$ does not hold when considering the $D(y)$ case. This can be achieved by adding the A-clause $\neg C(\mathsf{sk}_{\neg C(x)}) \leftarrow \{\neg[C(x)]\}$ using Tauto. If we use an SMT solver that is strong enough to determine that $\neg C(\mathsf{sk}_{\neg C(x)})$ and $D(y)$ are inconsistent, we can then apply StrongUnsat immediately, skipping the $D(y)$ branch altogether. This would be the case if we took $C(x) := \mathsf{f}(x) > 0$ and $D(y) := \mathsf{f}(y) > 3$ with a solver that supports linear arithmetic and quantifiers.

Theorem 12 (Soundness). *The rules* Unsat, Split, Collect, Trim, StrongUnsat, *and* Approx *are sound w.r.t.* $\models\!\!\approx$. *Moreover, if every rule in FInf is sound w.r.t.* $\models\!\!\approx$ *(on* $\mathcal{P}(\mathbf{F})$*), then the rule* Base *is sound w.r.t.* $\models\!\!\approx$ *(on* $\mathcal{P}(\mathbf{AF})$*).*

Proof. Cases Unsat, StrongUnsat: Trivial.

Case Split: For the left conclusion, by definition of $\models\!\!\approx$, it suffices to show $\mathit{fml}(\mathcal{J}) \cup \{C\} \models\!\!\approx \{\bot\}$ for every $\mathcal{J} \supseteq A \cup \{\neg a_1, \ldots, \neg a_n\}$. By the side condition $\{C\} \models\!\!\approx \{C_1, \ldots, C_n\}$, it suffices in turn to show $\mathit{fml}(\mathcal{J}) \cup \{C_i\} \models\!\!\approx \{\bot\}$ for every i. Notice that $\sim C_i \in \mathit{fml}(\mathcal{J})$. The entailment amounts to $\big(\mathit{fml}(\mathcal{J}) \setminus \{\sim C_i\}\big) \cup \{C_i\} \models\!\!\approx \{C_i\}$, which follows from (D2) and (D3).

For the right conclusions, we must show $\mathit{fml}(\mathcal{J}) \cup \lfloor \{C \leftarrow A\}_{\mathcal{J}} \rfloor \models\!\!\approx \{C_i\}$ for every $\mathcal{J} \supseteq \{a_i\}$. Notice that $C_i \in \mathit{fml}(\mathcal{J})$. The desired result follows from (D2) and (D3).

Case Collect: We must show $\{\bot \leftarrow A_i\}_{i=1}^n \models\!\!\approx \{C \leftarrow A\}$. This follows from the stronger side condition $\{\bot \leftarrow A_i\}_{i=1}^n \models\!\!\approx \{\bot \leftarrow A\}$.

Case Trim: Only the right conclusion is nontrivial. Let $\mathcal{N} = \{\bot \leftarrow A_i\}_{i=1}^n$. It suffices to show $\mathcal{N}_{\mathcal{J}} \cup \{C \leftarrow B\}_{\mathcal{J}} \models\!\!\approx \{C\}$ for every $\mathcal{J} \supseteq A$. Assume $\mathcal{J} \models\!\!\approx \mathcal{N}_{\mathcal{J}} \cup \lfloor \{C \leftarrow B\}_{\mathcal{J}} \rfloor$. By the side condition $\mathcal{N} \cup A \models\!\!\approx \{B\}$, we get $\mathcal{J} \models\!\!\approx \{B\}$. Hence, $\mathcal{J} \models\!\!\approx \mathcal{N}_{\mathcal{J}} \cup \{C\}$, and thus $\mathcal{J} \models\!\!\approx \{C\}$, as required.

Case Approx: The proof is as for the left conclusion of Split.

Case Base: To show $\{C_i \leftarrow A_i\}_{i=1}^n \models\!\!\approx \{D \leftarrow A_1 \cup \cdots \cup A_n\}$, by the definition of $\models\!\!\approx$ on A-formulas, it suffices to show $\{C_1, \ldots, C_n\} \models\!\!\approx \{D\}$. This follows from the soundness of the inferences in *FInf*. □

3.2 The Redundancy Criterion

Next, we lift the base redundancy criterion.

Definition 13. The *splitting redundancy criterion* $\mathit{SRed} = (\mathit{SRed}_{\mathrm{I}}, \mathit{SRed}_{\mathrm{F}})$ is specified as follows. An A-formula $C \leftarrow A \in \mathbf{AF}$ is redundant, written $C \leftarrow A \in \mathit{SRed}_{\mathrm{F}}(\mathcal{N})$, if either of these conditions is met:

(1) $C \in \mathit{FRed}_{\mathrm{F}}(\mathcal{N}_{\mathcal{J}})$ for every propositional interpretation $\mathcal{J} \supseteq A$; or
(2) there exists an A-formula $C \leftarrow B \in \mathcal{N}$ such that $B \subset A$.

An inference $\iota \in \mathit{SInf}$ is redundant, written $\iota \in \mathit{SRed}_{\mathrm{I}}(\mathcal{N})$, if either of these conditions is met:

(3) ι is a Base inference and $\{\iota\}_{\mathcal{J}} \subseteq \mathit{FRed}_{\mathrm{I}}(\mathcal{N}_{\mathcal{J}})$ for every $\mathcal{J}$; or
(4) ι is an Unsat inference and $\bot \in \mathcal{N}$.

Condition (1) is essentially a lifting of $\mathit{FRed}_{\mathrm{F}}$ to A-formulas. It is used both as such and to justify the Split and Collect rules, as we will see below. Condition (2) is used to justify Trim.

We will use *SRed* to justify global A-formula deletion, but also *FRed* for local A-formula deletion in the form of locking. These two redundancy criteria interact nicely via the enabled projection for any $\mathcal{N}$ and $\mathcal{J}$: $\mathit{FRed}_{\mathrm{F}}(\mathcal{N}_{\mathcal{J}}) \subseteq (\mathit{SRed}_{\mathrm{F}}(\mathcal{N}))_{\mathcal{J}} \subseteq \mathit{FRed}_{\mathrm{F}}(\mathcal{N}_{\mathcal{J}}) \cup \mathcal{N}_{\mathcal{J}}$; and if $\bot \notin \mathcal{N}$, then also $\mathit{SRed}_{\mathrm{I}}(\mathcal{N})_{\mathcal{J}} = \mathit{FRed}_{\mathrm{I}}(\mathcal{N}_{\mathcal{J}})$.

Lemma 14. *$\bot \notin SRed_{\mathrm{F}}(\mathcal{N})$ for every $\mathcal{N} \subseteq \mathbf{AF}$.*

Proof. By Lemma 3, condition (1) of the definition of $SRed_{\mathrm{F}}$ cannot apply. Nor can condition (2). □

Lemma 15. *SRed is a redundancy criterion.*

Proof. We will first show that the restriction *ARed* of *SRed* to BASE inferences is a redundancy criterion. Then we will consider UNSAT inferences.

We start by showing that *ARed* is a special case of the redundancy criterion $FRed^{\cap\mathcal{G},\sqsupset}$ of Waldmann et al. [29, Sect. 3]—the *intersection of lifted redundancy criteria with tiebreaker orders*. Then we can simply invoke Theorem 37 and Lemma 19 from their technical report [30].

We define a *tiebreaker order* $\sqsupset$ such that $C \leftarrow A \sqsupset D \leftarrow B$ if and only if $C = D$ and $A \supset B$. The only requirement is that $\sqsupset$ must be well founded, which is the case since the assertion sets of A-formulas are finite. We also define a family of *grounding functions* $\mathcal{G}_\mathcal{J}$ indexed by a propositional model $\mathcal{J}$. Here, "grounding" will mean enabled projection. For A-formulas $\mathcal{C}$, we set $\mathcal{G}_\mathcal{J}(\mathcal{C}) = \{\mathcal{C}\}_\mathcal{J}$. For inferences ι, we set $\mathcal{G}_\mathcal{J}(\iota) = \{\iota\}_\mathcal{J}$.

We must show that $\mathcal{G}_\mathcal{J}$ satisfies the following characteristic properties of grounding function: (G1) $\mathcal{G}_\mathcal{J}(\bot) = \{\bot\}$; (G2) for every $\mathcal{C} \in \mathbf{AF}$, if $\bot \in \mathcal{G}_\mathcal{J}(\mathcal{C})$, then $\mathcal{C} = \bot$; and (G3) for every $\iota \in SInf$, $\mathcal{G}_\mathcal{J}(\iota) \subseteq FRed_{\mathrm{I}}(\mathcal{G}_\mathcal{J}(concl(\iota)))$.

Condition (G1) obviously holds, and (G3) holds by property (R4) of *FRed*. However, (G2) does not hold, a counterexample being $\bot \leftarrow \{a\}$. On closer inspection, Waldmann et al. use (G2) only to prove static completeness (Theorems 27 and 45 in the technical report) but not to establish that $FRed^{\cap\mathcal{G},\sqsupset}$ is a redundancy criterion, so we can proceed. It is a routine exercise to check that *ARed* coincides with $FRed^{\cap\mathcal{G},\sqsupset} = (FRed_{\mathrm{I}}^{\cap\mathcal{G}}, FRed_{\mathrm{F}}^{\cap\mathcal{G},\sqsupset})$, which is defined as follows:

1. $\iota \in FRed_{\mathrm{I}}^{\cap\mathcal{G}}(\mathcal{N})$ if and only if for every propositional interpretation $\mathcal{J}$, $\mathcal{G}_\mathcal{J}(\iota) \subseteq FRed_{\mathrm{I}}(\mathcal{G}_\mathcal{J}(\mathcal{N}))$;
2. $\mathcal{C} \in FRed_{\mathrm{F}}^{\cap\mathcal{G},\sqsupset}(\mathcal{N})$ if and only if for every propositional interpretation $\mathcal{J}$ and every $\mathcal{D} \in \mathcal{G}_\mathcal{J}(\mathcal{C})$, $\mathcal{D} \in FRed_{\mathrm{F}}(\mathcal{G}_\mathcal{J}(\mathcal{N}))$ or there exists $\mathcal{C}' \in \mathcal{N}$ such that $\mathcal{C}' \sqsubset \mathcal{C}$ and $\mathcal{D} \in \mathcal{G}_\mathcal{J}(\mathcal{C}')$.

We also need to check that the consequence relation $\models$ used in *SRed* coincides with the consequence relation $\models_\mathcal{G}^\cap$, which is defined (in our convention with disjunctive consequence relations) as $\mathcal{M} \models_\mathcal{G}^\cap \{\mathcal{C}\}$ if and only if for all $\mathcal{J}$ and $D \in G_\mathcal{J}(\{\mathcal{C}\})$, $\mathcal{G}_\mathcal{J}(\mathcal{M}) \models \{D\}$. Fortunately, after expanding $\mathcal{G}_\mathcal{J}$ this is exactly the definition we used for the lifting of $\models$ to **AF**.

To extend the above result to *SRed*, we must show the second half of conditions (R2) and (R3) as well as (R4) for UNSAT inferences.

(R2) Given an UNSAT inference ι, we must show that if $\mathcal{M} \subseteq \mathcal{N}$ and $\iota \in SRed_{\mathrm{I}}(M)$, then $\iota \in SRed_{\mathrm{I}}(N)$. This holds because if $\bot \in \mathcal{M}$, then $\bot \in \mathcal{N}$.

(R3) Given an UNSAT inference ι, we must show that if $\mathcal{M} \subseteq SRed_{\mathrm{F}}(N)$ and $\iota \in SRed_{\mathrm{I}}(\mathcal{N})$, then $\iota \in SRed_{\mathrm{I}}(\mathcal{N} \setminus \mathcal{M})$. This amounts to proving that if $\bot \in \mathcal{N}$, then $\bot \in \mathcal{N} \setminus \mathcal{M}$, which follows from Lemma 14.

(R4) Given an UNSAT inference ι, we must show that if $\bot \in \mathcal{N}$, then $\iota \in \mathit{SRed}_\mathrm{I}(\mathcal{N})$. This follows from the definition of SRed_I. □

SRed is highly versatile. It can justify the deletion of A-formulas that are propositionally tautological, such as $C \leftarrow \{\mathsf{v}, \neg\mathsf{v}\}$ where $C \neq \bot$. It lifts base subsumption gracefully: If $D \in \mathit{FRed}_\mathrm{F}(\{C_i\}_{i=1}^n)$, then $D \leftarrow A_1 \cup \cdots \cup A_n \in \mathit{SRed}_\mathrm{F}(\{C_i \leftarrow A_i\}_{i=1}^n)$. It also allows other simplifications, as long as the assertions on A-formulas used to simplify a given $C \leftarrow A$ are contained in A. If the base criterion FRed_F supports subsumption (e.g., following the lines of Waldmann et al. [29]), this also extends to A-formulas: $D \leftarrow B \in \mathit{SRed}_\mathrm{F}(\{C \leftarrow A\})$ if D is strictly subsumed by C and $B \supseteq A$, or if $C = D$ and $B \supset A$. Finally, it is strong enough to justify case splits and the other simplification rules presented in Sect. 3.1.

Theorem 16 (Simplification). *For every* SPLIT, COLLECT, *or* TRIM *inference, the conclusions collectively make the premises redundant according to* SRed_F.

Proof. CASE SPLIT: We must show $C \leftarrow A \in \mathit{SRed}_\mathrm{F}(\{\bot \leftarrow \{\neg a_1, \ldots, \neg a_n\} \cup A\} \cup \{C_i \leftarrow \{a_i\}\}_{i=1}^n)$. By condition (1) of the definition of SRed_F, it suffices to show $C \in \mathit{FRed}_\mathrm{F}(\{\bot \leftarrow \{\neg a_1, \ldots, \neg a_n\}\}_\mathcal{J} \cup (\{C_i \leftarrow \{a_i\}\}_{i=1}^n)_\mathcal{J})$ for every $\mathcal{J} \supseteq A$. If $a_i \in \mathcal{J}$ for some i, this follows from SPLIT's side condition $C \in \mathit{FRed}_\mathrm{F}(\{C_i\})$. Otherwise, this follows from (R7), the requirement that $C \in \mathit{FRed}_\mathrm{F}(\{\bot\})$ since $C \neq \bot$.

CASE COLLECT: We must show $C \leftarrow A \in \mathit{SRed}_\mathrm{F}(\{\bot \leftarrow A_i\}_{i=1}^n)$. By condition (1) of the definition of SRed_F, it suffices to show $C \in \mathit{FRed}_\mathrm{F}((\{\bot \leftarrow A_i\}_{i=1}^n)_\mathcal{J})$ for every $\mathcal{J} \supseteq A$. If $A_i \subseteq \mathcal{J}$ for some i, this follows from COLLECT's side condition that $C \neq \bot$ and (R7). Otherwise, from the side condition $\{\bot \leftarrow A_i\}_{i=1}^n \not\approx \{\bot \leftarrow A\}$, we obtain $\emptyset \not\approx \{\bot\}$, which contradicts (D5).

CASE TRIM: We must show $C \leftarrow A \cup B \in \mathit{SRed}_\mathrm{F}(\{\bot \leftarrow A_i\}_{i=1}^n \cup \{C \leftarrow A\})$. This follows directly from condition (2) of the definition of SRed_F. □

Remark 17. Annoyingly, the redundancy criterion *SRed* does not mesh well with α-equivalence. We would expect the A-formula $\mathsf{p}(x) \leftarrow \{a\}$ to be subsumed by $\mathsf{p}(y) \leftarrow \emptyset$, where x, y are variables, but this is not covered by condition (2) of SRed_F because $\mathsf{p}(x) \neq \mathsf{p}(y)$. The simplest solution is to take $\mathbf{F}$ to be the quotient of some set of raw formulas by α-equivalence. An alternative is to generalize the theory so that the projection operator $\mathcal{G}_\mathcal{J}$ generates entire α-equivalence classes (e.g., $\mathcal{G}_\mathcal{J}(\{\mathsf{p}(x)\}) = \{\mathsf{p}(x), \mathsf{p}(y), \mathsf{p}(z), \ldots\}$) or groundings (e.g., $\mathcal{G}_\mathcal{J}(\{\mathsf{p}(x)\}) = \{\mathsf{p}(\mathsf{a}), \mathsf{p}(\mathsf{f}(\mathsf{a})), \ldots\}$). Waldmann et al. describe the second approach [29, Sect. 4].

3.3 Standard Saturation

We will now prove that the splitting calculus is statically complete and therefore dynamically complete. Unfortunately, derivations produced by most practical splitting architectures violate the fairness condition associated with dynamic

completeness. Nevertheless, the standard completeness notions are useful stepping stones, so we start with them.

Lemma 18. *Let $\mathcal{N} \subseteq \mathbf{AF}$ be an A-formula set, and let $\mathcal{J}$ be a propositional interpretation. If $\mathcal{N}$ is saturated w.r.t. SInf, SRed_I, and $\models$, then $\mathcal{N}_\mathcal{J}$ is saturated w.r.t. FInf, FRed_I, and $\models$.*

Proof. Assuming $\iota \in \mathit{FInf}(\mathcal{N}_\mathcal{J})$, we must show $\iota \in \mathit{FRed}_\mathrm{I}(\mathcal{N}_\mathcal{J})$. The argument follows that of the "folklore" Lemma 26 in the technical report of Waldmann et al. [30]. First note that any inference in FInf is lifted, via BASE, in SInf, so that we have $\iota \in (\mathit{SInf}(\mathcal{N}))_\mathcal{J}$. This means that there exists a BASE inference $\iota_0 \in \mathit{SInf}(\mathcal{N})$. By saturation of $\mathcal{N}$, we have $\iota_0 \in \mathit{SRed}_\mathrm{I}(\mathcal{N})$. By definition of SRed_I, $\{\iota_0\}_\mathcal{J} = \{\iota\} \subseteq \mathit{FRed}_\mathrm{I}(\mathcal{N}_\mathcal{J})$, as required. □

Theorem 19 (Static completeness). *Assume $(\mathit{FInf}, \mathit{FRed})$ is statically complete. Then $(\mathit{SInf}, \mathit{SRed})$ is statically complete.*

Proof. Suppose $\mathcal{N} \subseteq \mathbf{AF}$, $\mathcal{N} \models \{\bot\}$, and $\mathcal{N}$ is saturated w.r.t. SInf and SRed_I. We will show $\bot \in \mathcal{N}$.

First, we show $\bot \in \mathcal{N}_\mathcal{J}$ for every $\mathcal{J}$. From $\mathcal{N} \models \{\bot\}$, by the definition of $\models$ on A-formulas, it follows that $N_\mathcal{J} \models \{\bot\}$. Moreover, by Lemma 18, $N_\mathcal{J}$ is saturated w.r.t. FInf and FRed_I. By static completeness of $(\mathit{FInf}, \mathit{FRed})$, we get $\bot \in \mathcal{N}_\mathcal{J}$.

Hence $\mathcal{N}_\bot$ is propositionally unsatisfiable. By compactness of propositional logic, there exists a finite subset $\mathcal{M} \subseteq \mathcal{N}_\bot$ such that $\mathcal{M} \models \{\bot\}$. By saturation w.r.t. UNSAT, we obtain $\bot \in \mathcal{N}$, as required. □

Thanks to the requirements on the redundancy criterion, we obtain dynamic completeness as a corollary:

Corollary 20 (Dynamic completeness). *Assume $(\mathit{FInf}, \mathit{FRed})$ is statically complete. Then $(\mathit{SInf}, \mathit{SRed})$ is dynamically complete.*

Proof. This immediately follows from Theorem 19 by Lemma 6 in the technical report of Waldmann et al. [30]. □

3.4 Local Saturation

The above completeness result, about $\rhd_{\mathit{SRed}_\mathrm{F}}$-derivations, can be extended to prover designs based on the given clause procedure, such as the Otter, DISCOUNT, and Zipperposition loops, as explained by Waldmann et al. [29, Sect. 4]. But it fails to capture a crucial aspect of most splitting architectures. Since $\rhd_{\mathit{SRed}_\mathrm{F}}$-derivations have no notion of current split branch or propositional model, they place no restrictions on which inferences may be performed when. To fully capture splitting, we need to start with a weaker notion of saturation.

If an A-formula set is consistent, it should suffice to saturate w.r.t. a single propositional model. In other words, if no A-formula $\bot \leftarrow A \subseteq \mathcal{J}$ is derivable for some model $\mathcal{J} \models \mathcal{N}_\bot$, the prover will never be able to apply the UNSAT rule to derive $\bot$. It should be allowed to give a verdict of "consistent." We will refer to such model-specific saturations as *local* and to standard saturations as *global*.

Definition 21. A set $\mathcal{N} \subseteq \mathbf{AF}$ is *locally saturated* w.r.t. *SInf* and $SRed_\mathrm{I}$ if either $\bot \in \mathcal{N}$ or there exists a propositional model $\mathcal{J} \models \mathcal{N}_\bot$ such that $\mathcal{N}_\mathcal{J}$ is saturated w.r.t. *FInf* and $FRed_\mathrm{I}$.

Local saturation works in tandem with *strong static completeness*:

Theorem 22 (Strong static completeness). *Assume* $(FInf, FRed)$ *is statically complete. Given a set* $\mathcal{N} \subseteq \mathbf{AF}$ *that is locally saturated w.r.t. SInf and* $SRed_\mathrm{I}$ *and such that* $\mathcal{N} \models \{\bot\}$*, we have* $\bot \in \mathcal{N}$*.*

Proof. We show $\bot \in \mathcal{N}$ by case distinction on the condition by which $\mathcal{N}$ is locally saturated. The first case is vacuous. Otherwise, let $\mathcal{J} \models \mathcal{N}_\bot$. Since $\mathcal{N} \models \{\bot\}$, we have $\mathcal{N}_\mathcal{J} \models \{\bot\}$. By the definition of local saturation and static completeness of $(FInf, FRed)$, we get $\bot \in \mathcal{N}_\mathcal{J}$, contradicting $\mathcal{J} \models \mathcal{N}_\bot$. □

Example 23. Consider the A-clause set $\{\bot \leftarrow \{\neg[\mathsf{p}(x)], \neg[\mathsf{q}(y)]\}, \mathsf{p}(x) \leftarrow \{[\mathsf{p}(x)]\}, \mathsf{q}(y) \leftarrow \{[\mathsf{q}(y)]\}, \neg\mathsf{q}(\mathsf{a})\}$ expressed using AVATAR conventions. It is not globally saturated for resolution, because the conclusion $\bot \leftarrow \{[\mathsf{q}(y)]\}$ of resolving the last two A-clauses is missing, but it is locally saturated with $\mathcal{J} \supseteq \{[\mathsf{p}(x)], \neg[\mathsf{q}(y)]\}$.

Definition 24. A sequence $(\mathcal{N}_i)_i$ of sets of A-formulas is *locally fair* w.r.t. *SInf* and $SRed_\mathrm{I}$ if either $\bot \in \mathcal{N}_i$ for some i or there exists a propositional model $\mathcal{J} \models (\mathcal{N}_\infty)_\bot$ such that $FInf((\mathcal{N}_\infty)_\mathcal{J}) \subseteq \bigcup_i FRed_\mathrm{I}((\mathcal{N}_i)_\mathcal{J})$.

Lemma 25. *Let* $(\mathcal{N}_i)_i$ *be a* $\rhd_{SRed_\mathrm{F}}$*-derivation that is locally fair w.r.t. SInf and* $SRed_\mathrm{I}$*. Then the limit inferior* $\mathcal{N}_\infty$ *is locally saturated w.r.t. SInf and* $SRed_\mathrm{I}$*.*

Proof. The proof is by case distinction on the condition by which $(\mathcal{N}_i)_i$ is locally fair. If $\bot \in \mathcal{N}_i$, then $\bot \in \mathcal{N}_\infty$ by Lemma 14, and $\mathcal{N}_\infty$ is therefore locally saturated. In the remaining case we have $N_i \subseteq N_\infty \cup SRed_\mathrm{F}(N_\infty)$ by Lemma 4 in the technical report of Waldmann et al. [30], and therefore $\bigcup_i FRed_\mathrm{I}((\mathcal{N}_i)_\mathcal{J}) \subseteq \bigcup_i FRed_\mathrm{I}((\mathcal{N}_\infty)_\mathcal{J} \cup FRed_\mathrm{F}((N_\infty)_\mathcal{J})) = \bigcup_i FRed_\mathrm{I}((\mathcal{N}_\infty)_\mathcal{J})$ because we clearly have $SRed_\mathrm{F}(N_\infty)_\mathcal{J} \subseteq FRed_\mathrm{F}((N_\infty)_\mathcal{J}) \cup (N_\infty)_\mathcal{J}$. □

Local fairness works in tandem with *strong dynamic completeness*.

Theorem 26 (Strong dynamic completeness). *Assume* $(FInf, FRed)$ *is statically complete. Given an* $\rhd_{SRed_\mathrm{F}}$*-derivation* $(\mathcal{N}_i)_i$ *that is locally fair w.r.t. SInf and* $SRed_\mathrm{I}$ *and such that* $\mathcal{N}_0 \models \{\bot\}$*, we have* $\bot \in \mathcal{N}_i$ *for some* i*.*

Proof. We connect the dynamic and static points of view along the lines of the proof of Lemma 6 in the technical report of Waldmann et al. [30]. First, we show that the limit inferior is inconsistent: $\mathcal{N}_\infty \models \{\bot\}$. We have $\bigcup_i \mathcal{N}_i \supseteq \mathcal{N}_0 \models \{\bot\}$, and by (R1), it follows that $(\bigcup_i \mathcal{N}_i) \setminus SRed_\mathrm{F}(\bigcup_i \mathcal{N}_i) \models \{\bot\}$. By their Lemma 2, $(\bigcup_i \mathcal{N}_i) \setminus SRed_\mathrm{F}(\bigcup_i \mathcal{N}_i) \subseteq \mathcal{N}_\infty$. Hence $\mathcal{N}_\infty \supseteq (\bigcup_i \mathcal{N}_i) \setminus SRed_\mathrm{F}(\bigcup_i \mathcal{N}_i) \models \{\bot\}$. By Lemma 25, $\mathcal{N}_\infty$ is locally saturated, so by Theorem 22, $\bot \in \mathcal{N}_\infty$. Thus, $\bot \in \mathcal{N}_i$ for some i. □

An alternative proof based on dynamic completeness follows:

Proof. We show $\bot \in \mathcal{N}_i$ for some i by case distinction on the condition by which $(\mathcal{N}_i)_i$ is locally fair. The first case is vacuous. Otherwise, we have $\mathcal{J} \models (\mathcal{N}_\infty)_\bot$. Since $\mathcal{N}_0 \models \{\bot\}$, we have $(\mathcal{N}_0)_\mathcal{J} \models \{\bot\}$. By the definition of local fairness and Theorem 20, we get $\bot \in (\mathcal{N}_i)_\mathcal{J}$ for some i. By Lemma 3 and the definition of $\rhd_{FRed_\mathrm{F}}$, we obtain $\bot \in (\mathcal{N}_\infty)_\mathcal{J}$, contradicting $\mathcal{J} \models (\mathcal{N}_\infty)_\bot$. □

In Sects. 4 to 6, we will review three transition systems of increasing complexity, culminating with an idealized specification of AVATAR. They will be linked by a chain of stepwise refinements, like pearls on a string. All derivations using these systems will correspond to $\rhd_{SRed_\mathrm{F}}$-derivations, and their fairness criteria will imply local fairness. Consequently, by Theorem 26, they will all be complete.

4 Model-Guided Provers

The transition system $\rhd_{SRed_\mathrm{F}}$ provides a very abstract notion of splitting prover. AVATAR and other splitting architectures maintain a model of the propositional clauses, which represents the split tree's current branch. We can capture this abstractly by refining $\rhd_{SRed_\mathrm{F}}$-derivations to incorporate a propositional model.

4.1 The Transition Rules

The states are now pairs $(\mathcal{J}, \mathcal{N})$, where $\mathcal{J}$ is a propositional model and $\mathcal{N} \subseteq \mathbf{AF}$. Initial states have the form $(\mathcal{J}, N)$, where $N \subseteq \mathbf{F}$. The *model-guided prover* MG is defined by the following transition rules:

DERIVE	$(\mathcal{J}, \mathcal{N} \uplus \mathcal{M}) \Longrightarrow_\mathsf{MG} (\mathcal{J}, \mathcal{N} \uplus \mathcal{M}')$	if $\mathcal{M} \subseteq SRed_\mathrm{F}(\mathcal{N} \uplus \mathcal{M}')$
SWITCH	$(\mathcal{J}, \mathcal{N}) \Longrightarrow_\mathsf{MG} (\mathcal{J}', \mathcal{N})$	if $\mathcal{J}' \models \mathcal{N}_\bot$
STRONGUNSAT	$(\mathcal{J}, \mathcal{N}) \Longrightarrow_\mathsf{MG} (\mathcal{J}, \mathcal{N} \cup \{\bot\})$	if $\mathcal{N}_\bot \not\approx \{\bot\}$

The DERIVE rule can add new enabled A-formulas and delete redundant enabled A-formulas. In practice, DERIVE will perform only sound or satisfiability-preserving inferences, but we impose no such restriction. $\mathcal{J}$ should be a model of $\mathcal{N}_\bot$ most of the time; when it is not, SWITCH can be used to switch model or STRONGUNSAT to finish the refutation. Although the condition $\mathcal{J}_i \models (\mathcal{N}_i)_\bot$ might be violated for some i, to make progress we must periodically check it and apply SWITCH as needed.

Lemma 27. *If* $(\mathcal{J}, \mathcal{N}) \Longrightarrow_\mathsf{MG} (\mathcal{J}', \mathcal{N}')$, *then* $\mathcal{N} \rhd_{SRed_\mathrm{F}} \mathcal{N}'$.

Proof. The only rule that deletes A-formulas, DERIVE, exclusively takes out A-formulas that are redundant w.r.t. the next state, as mandated by $\rhd_{SRed_\mathrm{F}}$. □

To develop our intuitions, we will study several examples of $\Longrightarrow_\mathsf{MG}$-derivations. In all the examples in this section, the base calculus is first-order resolution, and $\models$ is entailment for first-order logic with equality.

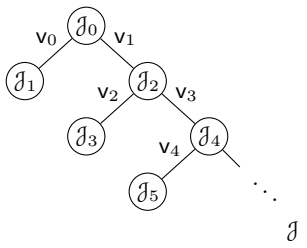

Fig. 1: A split tree with a single infinite branch

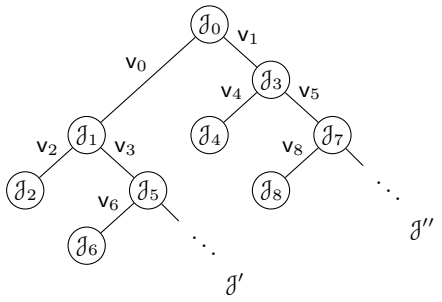

Fig. 2: A split tree with two infinite branches

Example 28. Let us revisit Example 10. Initially, the propositional interpretation is $\mathcal{J}_0 = \{\neg \mathsf{v}_0, \neg \mathsf{v}_1\}$. After the split, we have the A-clauses $\neg \mathsf{p}(\mathsf{a})$, $\neg \mathsf{q}(z, z)$, $\mathsf{p}(x) \leftarrow \{\mathsf{v}_0\}$, $\mathsf{q}(y, \mathsf{b}) \leftarrow \{\mathsf{v}_1\}$, and $\bot \leftarrow \{\neg \mathsf{v}_0, \neg \mathsf{v}_1\}$. The only option is to switch model. We take $\mathcal{J}_1 = \{\mathsf{v}_0, \neg \mathsf{v}_1\}$. We can then derive $\bot \leftarrow \{\mathsf{v}_0\}$. Since $\mathcal{J}_1 \not\models \bot \leftarrow \{\mathsf{v}_0\}$, we switch to $\mathcal{J}_2 = \{\neg \mathsf{v}_0, \mathsf{v}_1\}$, where we derive $\bot \leftarrow \{\mathsf{v}_1\}$. Finally, we detect that the propositional clauses are unsatisfiable and generate $\bot$.

4.2 Fairness

We need a fairness criterion for MG that implies local fairness of the underlying $\rhd_{SRed_{\mathrm{F}}}$-derivation. The latter requires a witness $\mathcal{J}$ but gives us no hint as to where to look for one. This is where basic topology comes into play.

Definition 29. A propositional interpretation $\mathcal{J}$ is a *limit point* in a sequence $(\mathcal{J}_i)_i$ if there exists a subsequence $(\mathcal{J}'_i)_i$ of $(\mathcal{J}_i)_i$ such that $\mathcal{J} = \mathcal{J}'_\infty = \mathcal{J}'^\infty$.

Example 30. Let $(\mathcal{J}_i)_i$ be the sequence such that $\mathcal{J}_{2i} \cap \mathbf{V} = \{\mathsf{v}_1, \mathsf{v}_3, \ldots, \mathsf{v}_{2i-1}\}$ (i.e., $\mathsf{v}_1, \mathsf{v}_3, \ldots, \mathsf{v}_{2i-1}$ are true and the other variables are false) and $\mathcal{J}_{2i+1} = (\mathcal{J}_{2i} \setminus \{\neg \mathsf{v}_{2i}\}) \cup \{\mathsf{v}_{2i}\}$. Although it is not in the sequence, the interpretation $\mathcal{J} \cap \mathbf{V} = \{\mathsf{v}_1, \mathsf{v}_3, \ldots\}$ is a limit point. The associated split tree is depicted in Fig. 1. The direct path from the root to a node labeled $\mathcal{J}_i$ specifies the assertions that are true in $\mathcal{J}_i$. The limit point $\mathcal{J}$ corresponds to the only infinite branch of the tree.

Example 31. Let $(\mathcal{J}_i)_i$ be the sequence such that $\mathcal{J}_0 \cap \mathbf{V} = \emptyset$, $\mathcal{J}_{4i+1} \cap \mathbf{V} = \{\mathsf{v}_0\} \cup \{4j + 3 \mid j < i\}$, $\mathcal{J}_{4i+2} \cap \mathbf{V} = \{\mathsf{v}_0, \mathsf{v}_{4i+2}\} \cup \{4j + 3 \mid j < i\}$, $\mathcal{J}_{4i+3} \cap \mathbf{V} = \{4j + 1 \mid j \leq i\}$, and $\mathcal{J}_{4i+4} \cap \mathbf{V} = \{4j + 1 \mid j \leq i\} \cup \{\mathsf{v}_{4i+4}\}$. This sequence has two limit points: $\mathcal{J}' = \liminf_{i\to\infty} \mathcal{J}_{4i+1}$ and $\mathcal{J}'' = \liminf_{i\to\infty} \mathcal{J}_{4i+3}$. The split tree is depicted in Fig. 2.

Lemma 32. *Let $(\mathcal{J}_i)_i$ be a sequence of propositional interpretations. Then $\mathcal{J}_\infty \subseteq \mathcal{J} \subseteq \mathcal{J}^\infty$ for all of its limit points $\mathcal{J}$.*

Proof. By definition of limit point, there must exist a subsequence $(\mathcal{J}'_i)_i$ of $(\mathcal{J}_i)_i$ such that $\mathcal{J}'_\infty = \mathcal{J}'^\infty = \mathcal{J}$. It is obvious by definition that the limit inferior of

a subsequence must be a superset of the limit inferior of the original sequence, and analogously that the limit superior of a subsequence must be a subset of the limit superior of the original sequence. □

Lemma 33. *Every sequence $(\mathcal{J}_i)_i$ of propositional interpretations has at least one limit point.*

Proof. The set of propositional interpretations is homeomorphic to the set of functions $\mathbf{V} \to \{0, 1\}$ equipped with the product topology. Since $\mathbf{V}$ is countable, this set of functions is a compact metric space, namely the Cantor space. In a compact metric space, every sequence has a convergent subsequence, and thus a limit point in our notation. □

We can nearly as easily supply an elementary proof:

Proof. We construct a subsequence $(\mathcal{J}'_j)_j$ converging to a limit point $\mathcal{J}$ in such a way that $\mathcal{J}'_j$ gets the first j variables right, i.e. that $\mathcal{J}'_j \models \mathsf{v}_k$ if and only if $\mathcal{J} \models \mathsf{v}_k$ for any $k \leq j$. We maintain the invariant that there are infinitely many elements in the sequence $(\mathcal{J}_i)_i$ that agree with this finite prefix.

Assume that we have already defined $\mathcal{J}'_0, \ldots, \mathcal{J}'_j$. Then there are by the induction hypothesis infinitely many elements $\mathcal{J}_i$ such that $\mathcal{J}_i \models \mathsf{v}_k$ if and only if $\mathcal{J}'_j \models \mathsf{v}_k$ for all $k \leq j$. Of these there are infinitely many with $\mathcal{J}_i \models \mathsf{v}_{j+1}$ or infinitely many with $\mathcal{J}_i \models \neg\mathsf{v}_{j+1}$. If there are infinitely many with $\mathcal{J}_i \models \mathsf{v}_{j+1}$, then set $\mathcal{J}'_{j+1} = \mathcal{J}_i$ for one such i, and analogously in the other case. □

Remark 34. Lemma 33 would fail for an uncountable set $\mathbf{V}$ because uncountable products of compact metric spaces are not sequentially compact in general [26]. Take for example $\mathbf{V} = \mathcal{P}(\mathbb{N})$ and define the sequence $(\mathcal{J}_i)_i$ so that $A \in \mathcal{J}_i$ if and only if $i \in A$. Then $(\mathcal{J}_i)_i$ has no limit points. To see this, assume that $\mathcal{J}$ is a limit point with the corresponding subsequence $(\mathcal{J}_{i(j)})_j$ converging to $\mathcal{J}$. Let χ_Q denote the characteristic function of a set Q. A limit point should satisfy $\lim_{j\to\infty} \chi_{\mathcal{J}_{i(j)}}(P) = \chi_{\mathcal{J}}(P)$ for any $P \in \mathbf{V}$. But here we have $\lim_{j\to\infty} \chi_{\mathcal{J}_{i(j)}}(P) = \lim_{j\to\infty} \chi_P(i(j))$. And if we take $P = \{i(2j) \mid j \geq 0\}$, then this subsequence is $1, 0, 1, 0, \ldots$, and it diverges.

Lemma 33 tells us that every sequence has a limit point. No matter how erratically the prover switches branches, it will systematically explore one of them. It then suffices to perform the base *FInf*-inferences fairly in that branch:

Definition 35. An $\Longrightarrow_{\mathsf{MG}}$-derivation $(\mathcal{J}_i, \mathcal{N}_i)_i$ is *fair* if either (1) $\bot \in \mathcal{N}_i$ for some i or (2) $\mathcal{J}_i \models (\mathcal{N}_i)_\bot$ for infinitely many indices i and there exists a limit point $\mathcal{J}$ of $(\mathcal{J}_i)_i$ such that $\mathit{FInf}((\mathcal{N}_\infty)_\mathcal{J}) \subseteq \bigcup_i \mathit{FRed}_\mathrm{I}((\mathcal{N}_i)_\mathcal{J})$.

Fairness of $\Longrightarrow_{\mathsf{MG}}$-derivations is intentionally defined in terms of FRed_I instead of SRed_I. This results in a better notion of fairness, since it allows the prover to ignore formulas and inferences that are locally redundant at the limit point, but not redundant w.r.t. $(\mathit{SInf}, \mathit{SRed})$. For example, the inference $(t = s \leftarrow \{\mathsf{v}_2\}, \mathsf{P}(t) \vee B \leftarrow \{\mathsf{v}_0\}, \mathsf{P}(s) \vee B \leftarrow \{\mathsf{v}_0, \mathsf{v}_2\})$ is locally redundant if $P(s) \leftarrow \{\mathsf{v}_1\}$ has already been derived. But it is not redundant w.r.t. $(\mathit{SInf}, \mathit{SRed})$.

Until $\bot$ is derived, it is impossible in an $\Longrightarrow_{\mathsf{MG}}$-derivation to delay Switch forever or to starve off Derive by performing only Switch transitions. Also note that we make no assumptions about the order in which propositional models are enumerated; the propositional solver is given carte blanche.

We might at first assume that a realistic prover would ensure $\mathit{FInf}((\mathcal{N}_\infty)_\mathcal{J}) \subseteq \bigcup_i \mathit{FRed}_\mathrm{I}((\mathcal{N}_i)_\mathcal{J})$ for all limit points $\mathcal{J}$. However this is not the case, even a prover based on the given-clause procedure with an age-based heuristic might only saturate one of the limit points, as we will see in Section 6.2.

Lemma 36. *Let $(\mathcal{J}_i, \mathcal{N}_i)_i$ be an $\Longrightarrow_{\mathsf{MG}}$-derivation such that $\mathcal{J}_i \models (\mathcal{N}_i)_\bot$ for infinitely many indices i. Then for every $\mathcal{D} \in (\mathcal{N}_\infty)_\bot$, there exists an index i such that $\mathcal{J}_j \models \{\mathcal{D}\}$ for all $j \geq i$.*

Proof. If $\mathcal{D} = \bot \leftarrow A \in (\mathcal{N}_\infty)_\bot$, then $\bot \leftarrow A \in \mathcal{N}_i$ for some k. Since every $\Longrightarrow_{\mathsf{MG}}$-derivation is an $\rhd_{\mathit{SRed}_\mathrm{F}}$-derivation, for every $j \geq k$ we then have $\bot \leftarrow B \in \mathcal{N}_j$ for some $B \subseteq A$ (because $\bot \leftarrow A$ can only become redundant due to such a $\bot \leftarrow B$). Note that $\{\bot \leftarrow B\} \models \bot \leftarrow A$ and hence $(\mathcal{N}_j)_\bot \models \bot \leftarrow A$ for all $j \geq k$. By the assumption there exists an index $i \geq k$ such that $\mathcal{J}_i \models \bot \leftarrow A$. For $j \geq i$, the interpretation only changes in the Switch transition, which has $\mathcal{J}_j \models (\mathcal{N}_j)_\bot$ as a side condition. Since $(\mathcal{N}_j)_\bot \models \bot \leftarrow A$, we have $\mathcal{J}_j \models \bot \leftarrow A$ for all $j \geq i$. □

Lemma 37. *Let $(\mathcal{J}_i, \mathcal{N}_i)_i$ be an $\Longrightarrow_{\mathsf{MG}}$-derivation such that $\mathcal{J}_i \models (\mathcal{N}_i)_\bot$ for infinitely many indices i, and let $\mathcal{J}$ be a limit point of $(\mathcal{J}_i)_i$. Then $\mathcal{J} \models (\mathcal{N}_\infty)_\bot$.*

Proof. Let $\mathcal{C} \in (\mathcal{N}_\infty)_\bot$. By Lemma 36, there exists an index i such that $\mathcal{J}_j \models \{\mathcal{C}\}$ for all $j \geq i$. Let $(\mathcal{J}'_i)_i$ be the subsequence associated with the limit point $\mathcal{J}$, then there also exists an index i' such that $\mathcal{J}'_j \models \{\mathcal{C}\}$ for all $j \geq i'$ and hence $\mathcal{J} \models \{\mathcal{C}\}$. □

Lemma 38. *Let $(\mathcal{J}_i, \mathcal{N}_i)_i$ be a fair $\Longrightarrow_{\mathsf{MG}}$-derivation. Then $(\mathcal{N}_i)_i$ is a $\rhd_{\mathit{SRed}_\mathrm{F}}$-derivation that is locally fair w.r.t. SInf and SRed_I.*

Proof. The case where $\bot \in \mathcal{N}_i$ for some i is trivial. Otherwise, we have that $\mathcal{J}_i \models (\mathcal{N}_i)_\bot$ for infinitely many i and there is a limit point $\mathcal{J}$ such that $\mathit{FInf}((\mathcal{N}_\infty)_\mathcal{J}) \subseteq \bigcup_i \mathit{FRed}_\mathrm{I}((\mathcal{N}_i)_\mathcal{J})$. We take this limit point as the witness for $\mathcal{J}$ in the definition of local fairness. It remains to show that $\mathcal{J} \models (\mathcal{N}_\infty)_\bot$, which follows from Lemma 37. □

Fairness of an $\Longrightarrow_{\mathsf{MG}}$-derivation implies fairness of the underlying $\rhd_{\mathit{SRed}_\mathrm{F}}$-derivation.

A well-behaved propositional solver, as in labeled splitting, always gives rise to a single limit point $\mathcal{J}_\infty$, which can be taken for $\mathcal{J}$ in Definition 35. To achieve this kind of fairness, a splitting prover would perform all inferences from persistently enabled A-formulas—that is, A-formulas that eventually become enabled and remain enabled forever. In a prover based on the given clause procedure, this can be implemented in the standard way, using an age-based selection heuristic [25, Sect. 4]. However, such a strategy is not sufficient if the prover implements

local redundancy elimination as we shall see in the following Sects. 5 and 7.2, even if the propositional solver is well-behaved.

By contrast, an unconstrained solver, as supported by AVATAR, can produce multiple limit points; in particular, the restart feature of modern SAT solvers [18] could lead to this kind of behavior. Then it is more challenging to ensure fairness, as we will see in Sect. 6.

Example 39. Suppose that we leave out $\neg\mathsf{q}(z, z)$ from the initial clause set of Example 10. Then we can still derive $\bot \leftarrow \{\mathsf{v}_0\}$, as in Example 28, but not $\bot \leftarrow \{\mathsf{v}_1\}$. By static completeness of the splitting calculus, we conclude that the A-clause set is consistent.

Example 40. Consider the initial clause set consisting of $\mathsf{p}(x) \vee \mathsf{q}(\mathsf{a})$ and $\neg\mathsf{q}(y) \vee \mathsf{q}(\mathsf{f}(y))$. Without splitting, a resolution prover would diverge attempting to generate infinitely many clauses of the form $\neg\mathsf{p}(x) \vee \mathsf{q}(\mathsf{f}^i(\mathsf{a}))$. By contrast, in a splitting prover, we might split the first clause, yielding the A-clauses $\mathsf{p}(x) \leftarrow \{\mathsf{v}_0\}$, $\mathsf{q}(\mathsf{a}) \leftarrow \{\mathsf{v}_1\}$, and $\bot \leftarrow \{\neg\mathsf{v}_0, \neg\mathsf{v}_1\}$. If we then choose the model $\{\mathsf{v}_1\}$ and commit to it, we will also diverge, although somewhat faster since we do not need to carry around the literal $\neg\mathsf{p}(x)$. On the other hand, if we at any point switch to $\{\mathsf{v}_0\}$, we notice that $\{\mathsf{p}(x)\}$ is saturated and terminate. This illustrates the benefits of employing an unconstrained SAT solver.

Example 41. It is crucial to invoke the SAT solver often enough—in other words, to take Switch and StrongUnsat transitions periodically. Suppose that the inconsistent initial clause set of Example 10 is supplemented by the prolific but unhelpful clauses $\mathsf{r}(\mathsf{a})$ and $\neg\mathsf{r}(x) \vee \mathsf{r}(\mathsf{f}(x))$. We can perform the same split as before, but if we ignore the fairness condition that $\mathcal{J}_i \models (\mathcal{N}_i)_\bot$ must hold infinitely often, we can stick to the interpretation $\{\neg\mathsf{v}_1, \neg\mathsf{v}_2\}$ and derive useless consequences of the form $\mathsf{r}(\mathsf{f}^i(\mathsf{a}))$ forever, thereby failing to generate $\bot$. Similarly, the SAT solver must be invoked eventually after deriving a propositional clause $\bot \leftarrow A$ that conflicts with the current interpretation.

Example 42. Consider the consistent set consisting of $\neg\mathsf{p}(x)$, $\mathsf{p}(\mathsf{a}) \vee \mathsf{q}(\mathsf{a})$, and $\neg\mathsf{q}(y) \vee \mathsf{p}(\mathsf{f}(y)) \vee \mathsf{q}(\mathsf{f}(y))$. Splitting the second clause into $\mathsf{p}(\mathsf{a})$ and $\mathsf{q}(\mathsf{a})$ and resolving $\mathsf{q}(\mathsf{a})$ with the third clause yields $\mathsf{p}(\mathsf{f}(\mathsf{a})) \vee \mathsf{q}(\mathsf{f}(\mathsf{a}))$. This process can be iterated to yield arbitrarily many applications of f. Now suppose that v_{2i} and v_{2i+1} are associated with $\mathsf{p}(\mathsf{f}^i(\mathsf{a}))$ and $\mathsf{q}(\mathsf{f}^i(\mathsf{a}))$, respectively. If we split every emerging clause $\mathsf{p}(\mathsf{f}^i(\mathsf{a})) \vee \mathsf{q}(\mathsf{f}^i(\mathsf{a}))$ and the SAT solver always makes v_{2i} true before v_{2i+1}, we end up with the situation of Example 30 and Fig. 1. For the limit point $\mathcal{J}$, all *FInf*-inferences are performed. Thus, the derivation is fair.

Example 43. We build a clause set from two copies of Example 42, where each clause C from each copy $i \in \{1, 2\}$ is extended to $\neg\mathsf{r}_i \vee C$. We add the clause $\mathsf{r}_1 \vee \mathsf{r}_2$ and split it as our first move. From there, each branch imitates Example 42. A SAT solver might jump back and forth between them, as in Example 31 and Fig. 2. Even if A-clauses get disabled and re-enabled infinitely often, we must select them eventually and perform all nonredundant inferences in at least one of the two limit points ($\mathcal{J}'$ or $\mathcal{J}''$).

5 Locking Provers

With both AVATAR and labeled splitting, an enabled A-clause can be redundant locally, w.r.t. the current interpretation and yet nonredundant globally. Both architectures provide mechanisms to temporarily lock away such A-clauses, to be unlocked when the prover comes back to an interpretation where they are no longer locally redundant. In AVATAR, these conditionally deleted A-clauses are stored in the locked set; in labeled splitting, they are stored in the split stack. We refine the model-guided prover into a locking prover that captures these mechanisms.

5.1 The Transition Rules

The states of a locking derivation are triples $(\mathcal{J}, \mathcal{N}, \mathcal{L})$, where $\mathcal{J}$ is an interpretation, $\mathcal{N} \subseteq \mathbf{AF}$ is a set of A-formulas, and $\mathcal{L} \subseteq \mathbf{A} \times \mathbf{AF}$ is a set of pairs of assertions and A-formulas. Intuitively, $(B,\ C \leftarrow A) \in \mathcal{L}$ means that $C \leftarrow A$ is "locally redundant" in interpretations $\mathcal{J} \supseteq B$. The function $[\![\ ]\!]$ erases the locks: $[\![\mathcal{L}]\!] = \{\mathcal{C} \mid (B, \mathcal{C}) \in \mathcal{L} \text{ for some } B\}$. Initial states have the form $(\mathcal{J}, N, \emptyset)$, where $N \subseteq \mathbf{F}$. The *locking prover* is defined by the following transition rules:

LIFT $(\mathcal{J}, \mathcal{N}, \mathcal{L}) \Longrightarrow_{\mathsf{L}} (\mathcal{J}', \mathcal{N}' \cup [\![\mathcal{U}]\!], \mathcal{L} \setminus \mathcal{U})$
if $(\mathcal{J}, \mathcal{N}) \Longrightarrow_{\mathsf{MG}} (\mathcal{J}', \mathcal{N}')$ and $\mathcal{U} = \{(B, C \leftarrow A) \in \mathcal{L} \mid B \not\subseteq \mathcal{J}' \text{ and } A \subseteq \mathcal{J}'\}$

LOCK $(\mathcal{J}, \mathcal{N} \uplus \{C \leftarrow A\}, \mathcal{L}) \Longrightarrow_{\mathsf{L}} (\mathcal{J}, \mathcal{N}, \mathcal{L} \cup \{(B, C \leftarrow A)\})$
if $B \subseteq \mathcal{J}$ and $C \in \mathit{FRed}_{\mathrm{F}}(\mathcal{N}_{\mathcal{J}'})$ for all $\mathcal{J}' \supseteq A \cup B$

Lemma 44. *Let $(\mathcal{J}_i, \mathcal{N}_i, \mathcal{L}_i)_i$ be an $\Longrightarrow_{\mathsf{L}}$-derivation. Then $(\mathcal{J}_i, \mathcal{N}_i \cup [\![\mathcal{L}_i]\!])_i$ is an $\Longrightarrow_{\mathsf{MG}}$-derivation.*

Proof. Every LIFT transition clearly corresponds to a transition in MG. We can also map the LOCK transition to the DERIVE transition in MG by setting $\mathcal{M} = \mathcal{M}' = \emptyset$. □

5.2 Counterexamples

Locking can cause incompleteness, because an A-formula can be locally redundant at every point in the derivation and yet not be so at any limit point, thereby breaking local saturation. For example, if we have derived $\mathsf{p}(x) \leftarrow \{\neg \mathsf{v}_k\}$ for every k, then $\mathsf{p}(\mathsf{c})$ it is locally redundant in any interpretation $\mathcal{J}$ that contains $\neg \mathsf{v}_k$. If the sequence of interpretations is given by $\mathcal{J}_i = \{\mathsf{v}_1, \ldots, \mathsf{v}_i, \neg \mathsf{v}_{i+1}, \ldots\}$, the clause $\mathsf{p}(\mathsf{c})$ would always be locally redundant and never be considered for inferences. Yet $\mathsf{p}(\mathsf{c})$ might not be locally redundant at the unique limit point $\mathcal{J} = \mathbf{V}$.

Example 45. Consider the inconsistent initial clause set $\{\mathsf{t}(x, y), \mathsf{r}(0), \neg \mathsf{r}(x) \vee \mathsf{r}(\mathsf{s}(x)), \neg \mathsf{r}(x) \vee \neg \mathsf{o}(x) \vee \neg \mathsf{t}(x, y) \vee \mathsf{p}(y), \neg \mathsf{p}(\mathsf{c}), \neg \mathsf{q}(\mathsf{c}), \neg \mathsf{t}(x, z) \vee \neg \mathsf{t}(x, y) \vee \mathsf{p}(x) \vee \mathsf{q}(x)\}$ and ordered resolution with selection as inference system. We always select the maximum negative literal with the precedence $\mathsf{q} \prec \mathsf{p} \prec \mathsf{t} \prec \mathsf{o} \prec \mathsf{r}$. Following an

age-based selection heuristic, the prover will after some steps derive $\mathsf{p}(y) \leftarrow \{\mathsf{v}_0\}$ and $\mathsf{p}(x) \vee \mathsf{q}(x)$ (where $\mathit{fml}(\mathsf{v}_i) = \neg\mathsf{t}(\mathsf{s}^i(0), y) \vee \mathsf{p}(y)$)). The prover switches to a model where v_i is true for $i \geq 0$. The second clause is then clearly locally redundant, so the LOCK transition applies and it can be locked as $(\mathsf{p}(x) \vee \mathsf{q}(x), \{\mathsf{v}_0\})$.

Just before $\mathsf{p}(y) \leftarrow \{\mathsf{v}_0\}$ would be selected for inferences, $\mathsf{p}(y) \leftarrow \{\mathsf{v}_1\}$ is derived. Thus, when the prover switches to a new model after deriving $\bot \leftarrow \{\mathsf{v}_0\}$, where v_i is true if $i \geq 1$, the clause $\mathsf{p}(x) \vee \mathsf{q}(x)$ can be immediately relocked. This process can be repeated indefinitely. The clause $\mathsf{p}(x) \vee \mathsf{q}(x)$, which is necessary for a refutation, is then ignored because it is always locally redundant. It is locked each time the prover selects an A-clause for inferences, due to a different A-clause. But it is not locally redundant at the limit point $\mathcal{J} = \neg\mathbf{V}$.

In the derivation in Example 45, locking is not applied exhaustively: The A-clause $\neg\mathsf{t}(\mathsf{s}^i(0), y) \vee \mathsf{p}(y) \leftarrow \{\mathsf{v}_i\}$ is not locked, even though $\mathsf{p}(y) \leftarrow \{\mathsf{v}_j\}$ has already been derived. Also $\mathsf{p}(x) \vee \mathsf{q}(x)$ is unlocked infinitely often. This situation is unrealistic and would not happen in a practical prover such as Vampire. We could hope that is enough for completeness to forbid these situations. However, this is not the case, as we can see from a more complicated example:

Example 46. Let us construct a concrete example in AVATAR where local redundancy causes incompleteness. The inference system is ordered resolution with selection using the precedence $\mathsf{o} \prec \mathsf{q}_1 \prec \mathsf{q}_2 \prec \mathsf{p}_1 \prec \mathsf{p}_2 \prec \mathsf{t} \prec \mathsf{r}_1 \prec \mathsf{r}_2 \prec \mathsf{k}$, selecting nothing if the clause is positive or of the form $\neg\mathsf{k}(\ldots) \vee \mathsf{k}(\ldots)$ and otherwise selecting the largest negative literal. The initial clauses consist of the following. First we have a splittable clause $\mathsf{q}_1(x) \vee \mathsf{q}_2(y)$. Then we have clauses $\mathsf{k}(0)$ and $\neg\mathsf{k}(x) \vee \mathsf{k}(\mathsf{s}(x))$. (We use the predicate symbol k to delay clause selection in an age-based selection heuristic by adding the literal $\neg\mathsf{k}(\mathsf{s}^n(x))$ to a clause.) We also add the clause $\mathsf{t}(x, y)$, so that we can prevent splitting by adding the literal $\neg\mathsf{t}(x, y)$ to a clause. Finally, we add two sets of clauses, contradicting $\mathsf{q}_1(x)$ and $\mathsf{q}_2(x)$, resp. First: $\mathsf{r}_1(0)$; $\neg\mathsf{o}(x) \vee \neg\mathsf{r}_1(x) \vee \neg\mathsf{t}(x, y) \vee \mathsf{p}_1(y)$; $\neg\mathsf{r}_1(x) \vee \mathsf{r}_1(\mathsf{s}(\mathsf{s}(x)))$; $\neg\mathsf{q}_1(x) \vee \mathsf{p}_1(x)$; $\neg\mathsf{k}(\mathsf{s}(\mathsf{s}(\mathsf{s}(x)))) \vee \neg\mathsf{p}_1(x)$. And second: $r_2(\mathsf{s}(0))$; $\neg\mathsf{o}(x) \vee \neg\mathsf{r}_2(x) \vee \neg\mathsf{t}(x, y) \vee \mathsf{p}_2(y)$; $\neg\mathsf{r}_2(x) \vee \mathsf{r}_2(\mathsf{s}(\mathsf{s}(x)))$; $\neg\mathsf{q}_2(x) \vee \mathsf{p}_2(x)$; $\neg\mathsf{k}(\mathsf{s}(\mathsf{s}(\mathsf{s}(x)))) \vee \neg\mathsf{p}_2(x)$.

This clause set is clearly inconsistent. We will now sketch an infinite derivation corresponding to an age-based selection heuristic which does not derive $\bot$. First we split $\mathsf{q}_1(x) \vee \mathsf{q}_2(y)$ into $\mathsf{q}_1(x) \leftarrow \mathsf{a}$ and $\mathsf{q}_2(x) \leftarrow \mathsf{b}$ (with the additional propositional clause $\bot \leftarrow \neg\mathsf{a}, \neg\mathsf{b}$). The derivation then uses the derivations $\mathcal{J}_i$ defined as follows: $\mathcal{J}_i \models \mathsf{v}_j$ if and only if $j < i$, $\mathcal{J}_i \models \mathsf{w}_j$ if and only if $j \in \{i, i+1\}$, $\mathcal{J}_i \models \mathsf{a}$ if and only if i is even, and $\mathcal{J}_i \models \mathsf{b}$ if and only if i is odd. The assertions denote $\mathit{fml}(\mathsf{v}_i) = \neg\mathsf{o}(\mathsf{s}^i(0))$, $\mathit{fml}(\mathsf{w}_{2i}) = \neg\mathsf{q}(\mathsf{s}^{2i}, x) \vee \mathsf{p}_1(x)$, $\mathit{fml}(\mathsf{w}_{2i+1}) = \neg\mathsf{q}(\mathsf{s}^{2i+1}, x) \vee \mathsf{p}_2(x)$. The derivation thus alternates between two families of interpretations (even and odd indices), which create two different limit points.

In the first family $\mathcal{J}_{2i}$, the A-clause $\neg\mathsf{q}_2(x) \vee \mathsf{p}_2(x)$ is locally redundant given $\mathsf{q}_2(x) \leftarrow \mathsf{w}_{2i+1}$, and we can lock it using the assertion w_{2i+1}. Similarly, we can lock $\neg\mathsf{q}_1(x) \vee \mathsf{p}_1(x)$ in the second family $\mathcal{J}_{2i+1}$ using the assertion w_{2i+2}.

Note that $\neg\mathsf{q}_2(x) \vee \mathsf{p}_2(x)$ is only unlocked in interpretations $\mathcal{J}_{2i}$. In those interpretations, $\mathsf{q}_2(x) \leftarrow \mathsf{b}$ is disabled and hence no inferences can be performed

with $\neg \mathsf{q}_2(x) \vee \mathsf{p}_2(x)$. And the same holds for $\neg \mathsf{q}_1(x) \vee \mathsf{p}_1(x)$, it is only unlocked when no inferences can be performed with it.

The derivation hence never performs inferences with $\mathsf{q}_1(x) \leftarrow \mathsf{a}$ or $\mathsf{q}_2(x) \leftarrow \mathsf{b}$. Removing these A-clauses makes the set satisfiable, and hence the derivation does not derive $\bot$.

Given the right sequence of propositional interpretations returned by the SAT solver, the derivation in Example 46 could potentially happen in a prover such as Vampire. It is difficult to exclude that the SAT solver used by Vampire could produce this sequence, or generally to characterize the sequence of models produced by CDCL solvers in such a concrete way. This derivation is also strongly fair—that is, every inference possible infinitely often, perhaps intermittently, must eventually be made redundant. Thus strong fairness is not a sufficient criterion for completeness either.

5.3 Fairness

Our solution is as follows. Let $(\mathcal{J}_i, \mathcal{N}_i, \mathcal{L}_i)_i$ be an $\Longrightarrow_{\mathsf{L}}$-derivation, let $(\mathcal{J}'_j)_j$ be a subsequence of $(\mathcal{J}_i)_i$, and let $(\mathcal{N}'_j)_j$ be the corresponding subsequence of $(\mathcal{N}_i)_i$. For fairness, we now consider $\mathcal{N}'_\infty$, the A-formulas persistent in the unlocked subsequence $(\mathcal{N}'_j)_j$. By contrast, with no A-formulas locked away, fairness of $\Longrightarrow_{\mathsf{MG}}$-derivations could use $\mathcal{N}_\infty$.

Definition 47. An $\Longrightarrow_{\mathsf{L}}$-derivation $(\mathcal{J}_i, \mathcal{N}_i, \mathcal{L}_i)_i$ is *fair* if either (1) $\bot \in \bigcup_i \mathcal{N}_i$ or (2) $\mathcal{J}_i \models (\mathcal{N}_i)_\bot$ for infinitely many indices i and there exists a subsequence $(\mathcal{J}'_j)_j$ converging to a limit point $\mathcal{J}$ such that $\mathit{FInf}((\mathcal{N}'_\infty)_\mathcal{J} \cup ((\limsup_{j\to\infty} \llbracket \mathcal{L}'_j \rrbracket)_\mathcal{J} \setminus \llbracket \mathcal{L}'^\infty \rrbracket)_\mathcal{J}) \subseteq \bigcup_i \mathit{FRed}_{\mathrm{I}}((\mathcal{N}_i \cup \llbracket \mathcal{L}_i \rrbracket)_\mathcal{J})$, where $(\mathcal{N}'_j)_j$ and $(\mathcal{L}'_j)_j$ correspond to $(\mathcal{J}'_j)_j$.

Fairness of an $\Longrightarrow_{\mathsf{L}}$-derivation implies fairness of the corresponding $\Longrightarrow_{\mathsf{MG}}$-derivation. The condition on the sets $\mathcal{L}'_j$ ensures that inferences from A-formulas that are locked infinitely often, but not infinitely often with the same lock, are redundant at the limit point. This is subtle, but if we know that each A-formula is locked at most finitely often, then $\limsup_{j\to\infty} \llbracket \mathcal{L}'_j \rrbracket = \llbracket \mathcal{L}'^\infty \rrbracket$ and the inclusion in the definition above simplifies to $\mathit{FInf}((\mathcal{N}'_\infty)_\mathcal{J}) \subseteq \bigcup_i \mathit{FRed}_{\mathrm{I}}((\mathcal{N}_i \cup \llbracket \mathcal{L}_i \rrbracket)_\mathcal{J})$.

Lemma 48. *Let $(\mathcal{J}_i, \mathcal{N}_i, \mathcal{L}_i)_i$ be a fair $\Longrightarrow_{\mathsf{L}}$-derivation. Then $(\mathcal{J}_i, \mathcal{N}_i \cup \llbracket \mathcal{L}_i \rrbracket)_i$ is a fair $\Longrightarrow_{\mathsf{MG}}$-derivation.*

Proof. We have already shown that it is an $\Longrightarrow_{\mathsf{MG}}$-derivation in Lemma 44, it only remains to show fairness. Case (1) is trivial. In case (2) we need to show that $\mathit{FInf}((\liminf_{i\to\infty} \mathcal{N}_i \cup \llbracket \mathcal{L}_i \rrbracket)_\mathcal{J}) \subseteq \bigcup_i \mathit{FRed}_{\mathrm{I}}((\mathcal{N}_i \cup \llbracket \mathcal{L}_i \rrbracket)_\mathcal{J})$. So assume that $\iota \in \mathit{FInf}((\liminf_{j\to\infty} \mathcal{N}'_j \cup \llbracket \mathcal{L}'_j \rrbracket)_\mathcal{J})$. If all premises of ι are in $(\mathcal{N}'_\infty)_\mathcal{J} \cup ((\limsup_{j\to\infty} \llbracket \mathcal{L}'_j \rrbracket)_\mathcal{J} \setminus (\llbracket \mathcal{L}'^\infty \rrbracket)_\mathcal{J})$ then the inference is redundant by the definition of fairness of $\Longrightarrow_{\mathsf{L}}$-derivations.

Otherwise, one of the premises $\lfloor \mathcal{C} \rfloor$ is not in that set. Since $\mathcal{C}$ is a premise and not in $\mathcal{N}'_\infty$, we have $\mathcal{C} \in \limsup_{j\to\infty} \llbracket \mathcal{L}'_i \rrbracket$. Therefore $(B, \lfloor \mathcal{C} \rfloor \leftarrow A) \in \mathcal{L}'^\infty$ for some $A \subseteq \mathcal{J}$ and B. We have $B \subseteq \mathcal{J}$ (otherwise it would be unlocked), and thus $\lfloor \mathcal{C} \rfloor \in \mathit{FRed}((\mathcal{N}_i)_\mathcal{J})$ for some i by the side condition of the Lock transition and then ι is redundant because *FRed* is reduced. □

6 AVATAR-Based Provers

AVATAR was unveiled in 2014 by Voronkov [28], although it was reportedly present in Vampire already in 2012. Since then, he and his colleagues studied many options and extensions [4, 20]. A second implementation, in Lean's `super` tactic, is due to Ebner [10]. Here we attempt to capture AVATAR's essence.

The abstract AVATAR-based prover we define in this section extends the locking prover L with a given clause procedure [15]. A-formulas are moved in turn from the passive to the active set, where inferences are performed. The heuristic for choosing the next *given* A-formula to move is guided by timestamps indicating when the A-formulas were derived, to ensure fairness.

6.1 The Transition Rules

Let $\mathbf{TAF} = \mathbf{AF} \times \mathbb{N}$ be the set of *timestamped A-formulas*. Given a subset $\mathcal{N} \subseteq \mathbf{TAF}$, we define $\lbag \mathcal{N} \rbag = \{\mathcal{C} \mid (\mathcal{C}, t) \in \mathcal{N} \text{ for some } t\}$, and we overload existing notations to erase timestamps as necessary. Accordingly, $\lfloor \mathcal{N} \rfloor = \lfloor \lbag \mathcal{N} \rbag \rfloor$, $\mathcal{N}_\bot = \lbag \mathcal{N} \rbag_\bot$, and $\mathcal{N}_\mathcal{J} = \lbag \mathcal{N} \rbag_\mathcal{J}$. Note that we use a new set of calligraphic letters (e.g., $\mathcal{C}, \mathcal{N}$) to range over timestamped A-formulas and A-formulas sets. We say that $\mathcal{N}$ is enabled in $\mathcal{J}$ if and only if $\lbag \mathcal{N} \rbag$ is enabled in $\mathcal{J}$. We also define $\lbag (\mathcal{C}_1, \ldots, \mathcal{C}_n, \mathcal{D}) \rbag = (\lbag \mathcal{C}_1 \rbag, \ldots, \lbag \mathcal{C}_n \rbag, \lbag \mathcal{D} \rbag)$ for **TAF**-inferences ι. Using the saturation framework [29, Sect. 3], we lift a calculus $(\mathit{SInf}, \mathit{SRed})$ on **AF** to a calculus $(\mathit{TSInf}, \mathit{TSRed})$ on **TAF** with the tiebreaker order $<$ on timestamps. We then have $\lbag \mathit{TSInf}(\mathcal{N}) \rbag = \mathit{SInf}(\lbag \mathcal{N} \rbag)$, $\lbag \mathit{TSRed}_\mathrm{I}(\mathcal{N}) \rbag = \mathit{SRed}_\mathrm{I}(\lbag \mathcal{N} \rbag)$, $\lbag \mathit{TSRed}_\mathrm{F}(\mathcal{N}) \rbag \supseteq \mathit{SRed}_\mathrm{F}(\lbag \mathcal{N} \rbag)$, and $(\mathcal{C}, t + k) \in \mathit{TSRed}_\mathrm{F}(\{(\mathcal{C}, t)\})$ for any $k > 0$; in other words, if an A-formula appears with two timestamps, the later version is redundant. Traditionally, provers use the active or passive status as tiebreaker: An active clause may subsume a passive copy of itself, but not the other way around. Timestamps are more fine-grained.

A state is a tuple $(\mathcal{J}, \mathcal{A}, \mathcal{P}, \mathcal{Q}, \mathcal{L})$ consisting of an interpretation $\mathcal{J}$, a set of *active* A-formulas $\mathcal{A} \subseteq \mathbf{TAF}$, a set of *passive* A-formulas $\mathcal{P} \subseteq \mathbf{TAF}$, a set of disabled or propositional A-formulas $\mathcal{Q} \subseteq \mathbf{TAF}$, and a set of locked A-formulas $\mathcal{L} \subseteq \mathbf{TAF} \times \mathcal{P}_\mathrm{fin}(\mathbf{A})$ such that (1) $\mathcal{A}_\bot = \mathcal{P}_\bot = \emptyset$, (2) $\mathcal{A} \cup \mathcal{P}$ is enabled in $\mathcal{J}$, and (3) $\mathcal{Q}_\mathcal{J} \subseteq \{\bot\}$. For every $\mathcal{L} \subseteq \mathbf{TAF} \times \mathbf{A}$, we define $\lbag \mathcal{L} \rbag = \{(B, \lbag \mathcal{C} \rbag) \mid (B, \mathcal{C}) \in \mathcal{L}\} \subseteq \mathbf{AF} \times \mathbf{A}$.

The input clauses are first put in the passive $\mathcal{P}$ set. Once a clause is selected for inferences and all inferences with it have been made redundant, it is moved to the active set $\mathcal{A}$. Inferences such as Split produce propositional clauses and clauses that are not enabled in the current model, these are put into $\mathcal{Q}$. When switching to a new model, the prover moves the newly enabled clauses from $\mathcal{Q}$ to $\mathcal{P}$, and disabled clauses from $\mathcal{A}$ to $\mathcal{Q}$ to preserve the state invariant.

The division of nonactive A-formulas into the sets $\mathcal{P}$ and $\mathcal{Q}$ is done for notational convenience (e.g., $\mathcal{P}$ is a separate set because fairness will be stated in terms of $\mathcal{A}$ and $\mathcal{P}$). In a practical implementation, this division can and will be different: The set $\mathcal{Q}$ would typically be stored in two different data structures: the propositional clauses in $\mathcal{Q}_\bot$ are directly passed to the SAT solver and might not

even stored by the prover itself (unless required for proof output). The remaining A-formulas $\mathcal{Q} \setminus \mathcal{Q}_\bot$ are those that need to moved back into $\mathcal{P}$ when the prover switches to an interpretation where they are enabled. These might be stored in the same data structure as the set of locked A-formulas $\mathcal{L}$, which also need to be reactivated depending on the interpretation. They could also be stored in the same data structure as $\mathcal{P}$, with the prover checking on every access whether an A-formula is enabled in the current interpretation or not.

The *AVATAR-based prover* AV is defined as the following transitions:

Infer	$(\mathcal{J}, \mathcal{A}, \mathcal{P} \uplus \{\mathcal{C}\}, \mathcal{Q}, \mathcal{L}) \Longrightarrow_{\mathsf{AV}} (\mathcal{J}, \mathcal{A} \cup \{\mathcal{C}\}, \mathcal{P}', \mathcal{Q}', \mathcal{L})$ if $\mathit{TSInf}(\mathcal{A}, \{\mathcal{C}\}) \subseteq \mathit{TSRed}_{\mathrm{I}}(\mathcal{A} \cup \{\mathcal{C}\} \cup \mathcal{P}' \cup \mathcal{Q}')$, $\mathcal{P} \subseteq \mathcal{P}'$, and $\mathcal{Q} \subseteq \mathcal{Q}'$
Process	$(\mathcal{J}, \mathcal{A}, \mathcal{P}, \mathcal{Q}, \mathcal{L}) \Longrightarrow_{\mathsf{AV}} (\mathcal{J}, \mathcal{A}', \mathcal{P}', \mathcal{Q}', \mathcal{L})$ if $\mathcal{A} \supseteq \mathcal{A}'$ and $(\mathcal{A} \backslash \mathcal{A}') \cup (\mathcal{P} \backslash \mathcal{P}') \cup (\mathcal{Q} \backslash \mathcal{Q}') \subseteq \mathit{TSRed}_{\mathrm{F}}(\mathcal{A}' \cup \mathcal{P}' \cup \mathcal{Q}')$
Switch	$(\mathcal{J}, \mathcal{A}, \mathcal{P}, \mathcal{Q}, \mathcal{L}) \Longrightarrow_{\mathsf{AV}} (\mathcal{J}', \mathcal{A}', \mathcal{P}' \cup \lVert \mathcal{U} \rVert, \mathcal{Q}', \mathcal{L} \setminus \mathcal{U})$ if $\mathcal{J} \not\models \mathcal{Q}_\bot$, $\mathcal{J}' \models \mathcal{Q}_\bot$, $\mathcal{A}' = \{\mathcal{C} \in \mathcal{A} \mid \mathcal{C}$ is enabled in $\mathcal{J}'\}$, $\mathcal{U} = \{(B, (C \leftarrow A, t)) \in \mathcal{L} \mid B \not\subseteq \mathcal{J}'$ and $A \subseteq \mathcal{J}'\}$, and $\mathcal{A} \cup \mathcal{P} \cup \mathcal{Q} = \mathcal{A}' \cup \mathcal{P}' \cup \mathcal{Q}'$
StrongUnsat	$(\mathcal{J}, \mathcal{A}, \mathcal{P}, \mathcal{Q}, \mathcal{L}) \Longrightarrow_{\mathsf{AV}} (\mathcal{J}, \mathcal{A}, \mathcal{P}, \mathcal{Q} \cup \{(\bot, t)\}, \mathcal{L})$ if $\mathcal{Q}_\bot \not\approx \bot$
LockA	$(\mathcal{J}, \mathcal{A} \uplus \{(C \leftarrow A, t)\}, \mathcal{P}, \mathcal{Q}, \mathcal{L}) \Longrightarrow_{\mathsf{AV}}$ $(\mathcal{J}, \mathcal{A}, \mathcal{P}, \mathcal{Q}, \mathcal{L} \cup \{(B, (C \leftarrow A, t))\})$ if $B \subseteq \mathcal{J}$ and $C \in \mathit{FRed}_{\mathrm{F}}((\mathcal{A} \cup \mathcal{P})_{\mathcal{J}'})$ for every $\mathcal{J}' \supseteq A \cup B$

There is also a LockP rule that is identical to LockA except that it starts in the state $(\mathcal{J}, \mathcal{A}, \mathcal{P} \uplus \{(C \leftarrow A, t)\}, \mathcal{Q}, \mathcal{L})$. An AV-derivation is *well timestamped* if every A-formula introduced by a rule is assigned a unique timestamp.

Lemma 49. *Let $(\mathcal{J}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{Q}_i, \mathcal{L}_i)_i$ be an $\Longrightarrow_{\mathsf{AV}}$-derivation. Then $(\mathcal{J}_i, \lbag \mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \rbag, \lbag \mathcal{L}_i \rbag)_i$ is an $\Longrightarrow_{\mathsf{L}}$-derivation.*

Proof. The transitions map directly to the corresponding transitions in $\Longrightarrow_{\mathsf{L}}$; both Infer and Process map to a Lift of Derive. □

Lemma 50. *Let $(\mathcal{J}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{Q}_i, \mathcal{L}_i)_i$ be an $\Longrightarrow_{\mathsf{AV}}$-derivation such that $\mathcal{A}_0 = \emptyset$. Then $\mathit{TSInf}(\mathcal{A}_i) \subseteq \mathit{TSRed}_{\mathrm{I}}(\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \lVert \mathcal{L}_i \rVert)$ for all i.*

Proof. The invariant is preserved by each transition. □

6.2 Counterexamples

In contrast with nonsplitting provers, for AV fairness w.r.t. formulas does not imply fairness w.r.t. inferences. To ensure fairness in a nonsplitting prover, it suffices to select the oldest formula for inferences infinitely often; for example, provers can alternate between choosing the oldest and choosing the heuristically best formula. In splitting provers, such a strategy is incomplete, and we need a stronger fairness criterion. A problematic scenario involves two premises $\mathcal{C}, \mathcal{D}$ of

a binary inference ι and four transitions repeated forever, possibly with other steps interleaved: Infer makes $\mathcal{C}$ active; Switch disables it; Infer makes $\mathcal{D}$ active; Switch disables it. Even though $\mathcal{C}$ and $\mathcal{D}$ are selected in a strongly fair fashion, ι is never performed. We need an even stronger fairness criterion.

Example 51. Let the prover alternate between heuristic and age-based selection. Take two copies of the clause set $\{\mathsf{p}(0), \neg\mathsf{p}(x) \vee \mathsf{p}(\mathsf{s}(x)) \vee \mathsf{q}(x), \neg\mathsf{p}(x), \neg\mathsf{q}(x)\}$, one with the assertion $\{\mathsf{a}\}$, the other with the assertion $\{\mathsf{b}\}$, plus the propositional clause $\bot \leftarrow \{\neg\mathsf{a}, \neg\mathsf{b}\}$. The prover then chooses the A-clause $\mathsf{p}(0) \leftarrow \{\mathsf{a}\}$ for inferences, followed by $\neg\mathsf{p}(x) \vee \mathsf{p}(\mathsf{s}(x)) \vee \mathsf{q}(x) \leftarrow \{\mathsf{a}\}$. By resolution and splitting, it derives $\mathsf{p}(\mathsf{s}(0)) \leftarrow \{\mathsf{v}_0\}$, $\mathsf{q}(0) \leftarrow \{\mathsf{v}_1\}$, and $\bot \leftarrow \{\mathsf{a}, \neg\mathsf{v}_0, \neg\mathsf{v}_1\}$. It then switches to another model, which makes b true. There it selects $\neg\mathsf{p}(x) \vee \mathsf{p}(\mathsf{s}(x)) \vee \mathsf{q}(x) \leftarrow \{\mathsf{b}\}$ for inferences, deriving analogous A-clauses as in the a-branch.

If the models alternate between the a and b-branches, $\mathsf{p}(0) \leftarrow \{\dots\}$ or $\neg\mathsf{p}(x) \vee \mathsf{p}(\mathsf{s}(x)) \vee \mathsf{q}(x) \leftarrow \{\dots\}$ will always be the oldest A-clause after switching to a new model. If we are allowed to alternate the age-based selection with a heuristic selection, then we can cause the prover to switch model after selecting at most four A-clauses for inferences: if an A-clause $\mathsf{q}(\dots) \leftarrow \{\dots\}$ is enabled, heuristically select both that A-clause and $\neg\mathsf{q}(x) \leftarrow \{v\}$ for inferences (with $v \in \{\mathsf{a}, \mathsf{b}\}$ depending on the branch). Otherwise an A-clause of the form $\mathsf{p}(\dots) \leftarrow \{\dots\}$ is enabled, and we heuristically select it for inferences. Together with $\neg\mathsf{p}(x) \vee \mathsf{p}(\mathsf{s}(x)) \vee \mathsf{q}(x) \leftarrow \{v\}$ this will derive $\bot \leftarrow \{\dots\}$.

With this strategy, the prover will never select $\neg\mathsf{p}(x)$ for inferences, and thus never derive $\bot$.

In Example 51, the prover did not derive $\bot$ because it never performed an inference between $\mathsf{p}(0) \leftarrow \{\mathsf{a}\}$ and $\neg\mathsf{p}(x)$ (and analogously for b), even though these two A-clauses are enabled infinitely often. It is not sufficient for completeness to forbid this situation either. As we have seen in Example 46, there are strongly fair $\Longrightarrow_{\mathsf{L}}$-derivations that do not derive $\bot$ from an inconsistent initial clause set.

This counterexample cannot occur in Vampire, because as far as we are aware, Vampire exclusively uses (combinations of) age-weight selection strategies. However the example requires a highly unrestricted heuristic selection.

Unrelated to completeness, we might expect that an $\Longrightarrow_{\mathsf{AV}}$-derivation saturates every limit point. This is not the case either, even with age-based selection:

Example 52. Take the union of two copies of the clause set $\{\mathsf{p}(0), \neg\mathsf{p}(x) \vee \mathsf{p}(\mathsf{s}(x)) \vee \mathsf{q}(\mathsf{s}(x)), \neg\mathsf{q}(x)\}$, one with the assertion $\{\mathsf{a}\}$, the other with the assertion $\{\neg\mathsf{a}\}$, This A-clause set is consistent and we will sketch a derivation with two limit points $\mathcal{J} = \{\mathsf{a}\} \cup \bigcup_i \{\mathsf{v}_i, \neg\mathsf{w}_i\}$ and $\mathcal{J}' = \{\neg\mathsf{a}\} \cup \bigcup_i \{\mathsf{v}_i, \neg\mathsf{w}_i\}$, where $\mathit{fml}(\mathsf{v}_i) = \mathsf{p}(\mathsf{s}^i(0))$ and $\mathit{fml}(\mathsf{w}_i) = \mathsf{q}(\mathsf{s}^i(0))$.

On the first subsequence converging to $\mathcal{J}$, a is true, w_i is false for all i, and v_i is true if $\mathsf{p}(\mathsf{s}^i(0)) \leftarrow \{\dots\}$ has been derived before. In this subsequence, the prover will produce A-clauses of the form $\mathsf{p}(\mathsf{s}^i(0)) \vee \mathsf{q}(\mathsf{s}^i(0)) \leftarrow \{\mathsf{a}, \dots\}$, which can subsequently be split into $\mathsf{p}(\mathsf{s}^i(0)) \leftarrow \{\mathsf{v}_i\}$, $\mathsf{q}(\mathsf{s}^i(0)) \leftarrow \{\mathsf{w}_i\}$, and $\bot \leftarrow \{\neg\mathsf{v}_i, \neg\mathsf{w}_i, \mathsf{a}, \dots\}$.

Pick now a model where a is false, w_i is false for all i, and v_i is true if $\mathsf{p}(\mathsf{s}^i(0)) \leftarrow \{\dots\}$ has been derived before. Let j be such that v_j is false in the model. On the first subsequence $\neg\mathsf{a}$ never appears as an assertion in a derived A-clause, so this interpretation remains a model of the propositional clauses. If the prover runs long enough on the first subsequence, the clause $\mathsf{p}(\mathsf{s}^j(0)) \leftarrow \{\mathsf{v}_j\}$ eventually makes it into the active set. After that, switch to the model, derive $\bot \leftarrow \{\neg\mathsf{v}_j, \neg\mathsf{w}_j, \neg\mathsf{a}, \dots\}$, and switch back to the first subsequence.

The A-clause $\neg\mathsf{q}(x) \leftarrow \{\neg\mathsf{a}\}$ is never considered for inferences, and thus $\mathcal{J}'$ is not saturated; in other words, the persistent A-clauses enabled at $\mathcal{J}'$ do not form a saturated set.

6.3 Fairness

Definition 53. An $\Longrightarrow_{\mathsf{AV}}$-derivation $(\mathcal{J}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{Q}_i, \mathcal{L}_i)_i$ is *fair* if (1) $\bot \in \wr \bigcup_i \mathcal{Q}_i \int$ or (2) $\mathcal{J}_i \models (\mathcal{Q}_i)_\bot$ for infinitely many indices i and there exists a subsequence $(\mathcal{J}'_j)$ converging to a limit point $\mathcal{J}'_\infty$ such that (3) $\liminf_{j\to\infty} \mathit{TSInf}(\mathcal{A}'_j, \mathcal{P}'_j) = \emptyset$ and (4) $(\limsup_{j\to\infty} [\![\mathcal{L}'_j]\!])_{\mathcal{J}} \setminus [\![\mathcal{L}'^{\infty}]\!]_{\mathcal{J}} \subseteq \bigcup_i \mathit{FRed}_{\mathrm{F}}((\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup [\![\mathcal{L}_i]\!])_{\mathcal{J}})$.

Condition (3) states that all inferences involving passive A-formulas are redundant at the limit point. It would not suffice to simply require $\mathcal{P}'_\infty = \emptyset$ because A-formulas can move back and forth between the sets $\mathcal{A}$, $\mathcal{P}$, and $\mathcal{Q}$, as we saw in Example 51. Condition (4) is an additional condition on locks similar to the one in Definition 47.

Lemma 54. *Let $(\mathcal{J}_i, \mathcal{A}_i, \mathcal{Q}_i, \mathcal{P}_i, \mathcal{L}_i)_i$ be a fair $\Longrightarrow_{\mathsf{AV}}$-derivation such that $\mathcal{A}_0 = \emptyset$. Then the $\Longrightarrow_{\mathsf{L}}$-derivation $(\mathcal{J}_i, \wr \mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \int, \wr \mathcal{L}_i \int)_i$ is fair as well.*

Proof. First consider for the locking condition a $C \in (\limsup_{j\to\infty} \wr [\![\mathcal{L}'_j]\!] \int)_{\mathcal{J}} \setminus (\limsup_{j\to\infty} [\![\mathcal{L}'_j]\!])_{\mathcal{J}}$. Hence there is a sequence $((C \leftarrow A, t_k), B_k) \in [\![\mathcal{L}''_k]\!]$ where $\mathcal{L}''$ is a subsequence of $\mathcal{L}'$, $A \subseteq \mathcal{J}$, and $(C \leftarrow A, t_k) \notin \limsup_{j\to\infty} [\![\mathcal{L}'_j]\!]$ for all k. Either $(C \leftarrow A, t_0) \in \liminf_{j\to\infty} \mathcal{A}'_j \cup \mathcal{P}'_j$ or $(C \leftarrow A, t_0) \in \bigcup_i \mathit{TSRed}_{\mathrm{F}}(\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i)$. In both cases, inferences from C are redundant.

By a similar argument we obtain $(\liminf_{j\to\infty} \wr \mathcal{A}'_j \cup \mathcal{P}'_j \int)_{\mathcal{J}} \setminus (\liminf_{j\to\infty} \mathcal{A}'_j \cup \mathcal{P}'_j)_{\mathcal{J}} \subseteq \bigcup_i \mathit{FRed}_{\mathrm{F}}((\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup [\![\mathcal{L}_i]\!])_{\mathcal{J}})$. If now $\bot \in \bigcup_i \mathcal{Q}_i$, then we clearly

have $\Longrightarrow_{\mathsf{L}}$-fairness. Otherwise it suffices to show:

$$
\begin{aligned}
&\mathit{FInf}((\liminf_{j\to\infty} \mathcal{A}'_j \cup \mathcal{P}'_j \cup \mathcal{Q}'_j)_{\mathcal{J}}) \\
&= \mathit{FInf}((\liminf_{j\to\infty} \mathcal{A}'_j \cup \mathcal{P}'_j)_{\mathcal{J}}) \\
&= (\mathit{TSInf}(\liminf_{j\to\infty} \mathcal{A}'_j \cup \mathcal{P}'_j))_{\mathcal{J}} \\
&= (\liminf_{j\to\infty} \mathit{TSInf}(\mathcal{A}'_j \cup \mathcal{P}'_j))_{\mathcal{J}} \\
&= (\liminf_{j\to\infty} \mathit{TSInf}(\mathcal{A}'_j) \cup \underbrace{\mathit{TSInf}(\mathcal{A}'_j, \mathcal{P}'_j)}_{\to\emptyset})_{\mathcal{J}} \\
&\subseteq (\liminf_{j\to\infty} \mathit{TSRed}_{\mathrm{I}}(\mathcal{A}'_j \cup \mathcal{P}'_j \cup \mathcal{Q}'_j \cup [\![\mathcal{L}'_j]\!]))_{\mathcal{J}} \\
&\subseteq \bigcup_i (\mathit{TSRed}_{\mathrm{I}}(\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup [\![\mathcal{L}_i]\!]))_{\mathcal{J}} \\
&= \bigcup_i \mathit{FRed}_{\mathrm{I}}((\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup [\![\mathcal{L}_i]\!])_{\mathcal{J}})
\end{aligned}
$$

□

The $\Longrightarrow_{\mathsf{AV}}$ relation is concrete enough so that we can show that typical clause selection strategies are fair (assuming the restriction on locking we already required for $\Longrightarrow_{\mathsf{L}}$-derivations) and avoid the counterexamples in this section. Many selection strategies are combinations of basic strategies, such as choosing the smallest formula by weight or the oldest by age. We capture such strategies using selection orders $\lessdot$. Intuitively, $\mathcal{C} \lessdot \mathcal{D}$ if the prover will always select $\mathcal{C}$ before $\mathcal{D}$ if both are present. That is, the prover always chooses one of the $\lessdot$-minimal A-formulas. (Typically, there is more than one.) We use two selection orders: $\lessdot_{\mathbf{TAF}}$, on timestamped A-formulas, must be followed infinitely often; $\lessdot_{\mathbf{F}}$, on base formulas, must be followed otherwise.

Definition 55. Let X be a set. A *selection order* $\lessdot$ on X is an irreflexive and transitive relation such that $\{y \mid y \not\gtrdot x\}$ is finite for all $x \in X$.

Example 56. Let $X \subseteq \mathbf{TAF}$ be such that X only contains finitely many A-formulas with the same timestamp. Define $\lessdot_{\mathrm{age}}$ on X so that $(\mathcal{C}, t) \lessdot_{\mathrm{age}} (\mathcal{C}', t')$ if and only if $t < t'$. Then $\lessdot_{\mathrm{age}}$ is a selection order corresponding to age-based selection.

Remark 57. Every selection order is a well-founded relation. But not every well-founded relation is a selection order. A well-order is a selection order if and only if its order type is less than $\omega{+}1$. The ordinal $\omega{+}1 = \{0 < 1 < 2 < \ldots < \omega\}$ is not a selection order since $\{y \mid y \not\gtrdot \omega\}$ is infinite. Even well-founded relations of low rank need not be selection orders: The empty relation $\emptyset \subseteq \mathbb{N}\times\mathbb{N}$ is irreflexive, transitive, and well-founded (with rank zero), but not a selection order.

Selection orders on **TAF** also generalize the mechanism, outlined by Bachmair and Ganzinger in a footnote [2, Sect. 4.3] and elaborated by Schlichtkrull et al. [25, Sect. 4], of using an $\mathbb{N}$-valued weight function that is strictly monotone in the timestamp.

Example 58. Let $\mathbf{F}$ be the set of first-order clauses in a fixed signature. Define the selection order $\preccurlyeq_{\mathrm{nv}}$ on $\mathbf{F}$ by $C' \preccurlyeq_{\mathrm{nv}} C$ if and only if $|C'| \leq |C|$ where $|C|$ denotes the sum of the number of nonvariable positions in C. Then $\preccurlyeq_{\mathrm{nv}}$ is a selection order because there is only a finite number of first-order clauses with at most n nonvariable positions for any n. This selection order corresponds to a simple weight-based selection scheme.

The intersection of two orders $\preccurlyeq_1$ and $\preccurlyeq_2$ corresponds to the nondeterministic alternation between them; the prover may choose either a $\preccurlyeq_1$-minimal or a $\preccurlyeq_2$-minimal A-formula, at its discretion.

Lemma 59. *Let $\preccurlyeq_1$ and $\preccurlyeq_2$ be selection orders on X. Then $\preccurlyeq_1 \cap \preccurlyeq_2$ is a selection order as well.*

Proof. Irreflexivity and transitivity are preserved by intersections, and note that $\{y \mid \text{not } (x \prec_1 y \text{ and } x \prec_2 y)\} = \{y \mid x \not\prec_1 y\} \cup \{y \mid x \not\prec_2 y\}$ is finite as a union of two finite sets. □

Lemma 60. *Let $\prec$ be a selection order on an infinite set X. Then for all x and y there exists a z such that $x \prec z$ and $y \prec z$.*

Proof. The set $X \setminus \{z \mid x \prec z \text{ and } y \prec z\} = \{z \mid x \not\prec z\} \cup \{z \mid y \not\prec z\}$ is finite, and therefore $\{z \mid x \prec z \text{ and } y \prec z\}$ is infinite and in particular nonempty. □

To ensure completeness of the given clause procedure, we must restrict the inferences that the prover may perform; otherwise, it could derive infinitely many A-formulas with different assertions, causing it to switch between two branches of the split tree without making progress as in Example 51. Given $\mathcal{N} \subseteq \mathbf{AF}$, let $\lceil \mathcal{N} \rceil = \{A \mid C \leftarrow A \in \mathcal{N} \text{ for some } C\}$.

Definition 61. A function $F : \mathcal{P}(\mathbf{AF}) \to \mathcal{P}(\mathbf{AF})$ is *strongly finitary* if $\lfloor F(\mathcal{N}) \rfloor$ and $\bigcup \lceil F(\mathcal{N}) \rceil \setminus \bigcup \lceil \mathcal{N} \rceil$ are finite for any $\mathcal{N} \subseteq \mathbf{AF}$ such that $\lfloor \mathcal{N} \rfloor$ is finite.

Intuitively, a strongly finitary function F returns finitely many base formulas and finitely many new assertions, although it may return infinitely many A-formulas. Clearly, $F(\mathcal{N})$ is finite for any finite $\mathcal{N} \subseteq \mathbf{AF}$. If $\mathit{FInf}(N)$ is finite for any finite $N \subseteq \mathbf{F}$, then $\mathit{concl} \circ \mathit{SInf}$ is strongly finitary. Deterministic Split rules, such as AVATAR's, are also strongly finitary. On the other hand, a function F mapping $\bot \leftarrow \{a_i\}$ to $\mathit{fml}(a_i)$ is in general not strongly finitary, since $\lfloor F(\mathbf{AF}_\bot) \rfloor = \mathit{fml}(\mathbf{A})$ can be infinite even though $\lfloor \mathbf{AF}_\bot \rfloor = \{\bot\}$ is finite.

We can extend a strongly finitary F to sets of base formulas by taking $F_{\mathbf{F}}(N) = \lfloor F(N \times \mathcal{P}_{\mathrm{fin}}(\mathbf{A})) \rfloor$ for any $N \subseteq \mathbf{F}$ and to sets of timestamped A-formulas by taking $F_{\mathbf{TAF}}(\mathcal{N}) = F(\wr \mathcal{N} \int) \times \mathbb{N}$ for any $\mathcal{N} \subseteq \mathbf{TAF}$. The function $F_{\mathbf{F}}$ is then finitary, mapping finite sets to finite sets. Moreover, if F and G are strongly finitary, then so is $\mathcal{N} \mapsto F(\mathcal{N}) \cup G(\mathcal{N})$.

Simplification rules used by the prover must be restricted even more to ensure completeness, because they can lead to new splits and assertions, and hence switching to new models. For example, simplifying $\mathsf{p}(x * 0) \vee \mathsf{p}(x)$ to $\mathsf{p}(0) \vee \mathsf{p}(x)$

transforms an unsplittable clause into a splittable one. Even for the standard order on first-order clauses there can be infinitely many clauses that are smaller that a given clause. For example, with the lexicographic path order, the set $\{u' \mid u' \prec u\}$ is typically infinite for a term u. If simplification were to produce infinitely many new splittable clauses, the prover might split and switch models forever without making progress.

Example 62. Even if $\prec$ is a well-founded order on $\mathbf{F}$, and I is a set of binary inferences such that $C_2 \prec C_1$ and $D \prec C_1$ for all $(C_2, C_1, D) \in I$, simplification with I can still produce infinitely many base formulas. This is because in an AV-prover, the same base formula may be re-derived infinitely often (for example due to switching between two families of interpretations).

As a slightly abstract example, consider $\mathbf{F} = \mathbb{N} \cup \{\infty\}$ with $0 \prec 1 \prec 2 \prec \ldots \prec \infty$ and let $I = \{(n, \infty, n+1) \mid n \in \mathbb{N}\}$. If the prover then re-derives ∞ infinitely often, it might simplify ∞ in a different way each time, the first time to 1, then to 2, and so on. We hence need to ensure that, in the entire derivation, each formula is simplified only in a finite number of ways.

Definition 63. Let $\prec$ be a well-founded relation on $\mathbf{F}$, and let $\preceq$ be its reflexive closure. A function $S : \mathbf{AF} \to \mathcal{P}(\mathbf{AF})$ is a *strongly finitary simplification bound* for $\prec$ if $\mathcal{N} \mapsto \bigcup_{\mathcal{C} \in \mathcal{N}} S(\mathcal{C})$ is strongly finitary and $\lfloor \mathcal{C}' \rfloor \preceq \lfloor \mathcal{C} \rfloor$ for all $\mathcal{C}' \in S(\mathcal{C})$.

The prover may simplify an A-formula $\mathcal{C}$ to $\mathcal{C}'$ only if $\mathcal{C}' \in S(\mathcal{C})$. It may also delete $\mathcal{C}$. Strongly finitary simplification bounds are closed under unions, allowing the combination of simplification techniques based on $\prec$. For superposition, a natural choice for $\prec$ is the clause order. The key property of strongly finitary simplification bounds is that if we saturate a finite set of A-formulas w.r.t. simplifications, the saturation is also finite. This is crucial to bound the number of A-formulas derived by the prover and thus the number of possible model changes: if the prover only selects a finite set of A-formulas for inferences, then simplification will only derive finitely many A-formulas as well, no matter how often an A-formula is derived again:

Lemma 64. *Let S be a strongly finitary simplification bound. For every $\mathcal{C} \in \mathbf{AF}$, let $S^*(\mathcal{C}) = \bigcup_{i=0}^{\infty} S^i(\mathcal{C})$, where S^i denotes the ith iterate of S. Then S^* is also a strongly finitary simplification bound.*

Proof. Let $N \subseteq \mathbf{F}$. It suffices to show that $S^*_{\mathbf{F}}(N)$ is finite as well. We define a sequence of finite sets $M_i \subseteq \mathbf{F}$ by $M_0 = N$ and $M_{i+1} = M_i \cup S_{\mathbf{F}}(M_i)$. Clearly $S_{\mathbf{F}}(M_\infty) \cup N \subseteq M_\infty = S^*_{\mathbf{F}}(N)$. To show that M_∞ is finite, consider the sequence $M_{i+1} \setminus M_i = S_{\mathbf{F}}(M_i \setminus M_{i-1}) \setminus M_i$, which is decreasing in the multiset extension of $\prec$, and therefore eventually constant. It follows that $M_{i+1} \setminus M_i = \emptyset$ for large enough i, and that $S^*_{\mathbf{F}}(N) = M_\infty = \bigcup_i M_i = M_0 \cup \bigcup_i (M_{i+1} \setminus M_i)$ is finite. □

Example 65. Let $\mathbf{F}$ be the set of first-order clauses and $S(C \leftarrow A) = \{C' \leftarrow A' \mid C'$ is a subclause of C and $A' \subseteq A\}$. Then S is a strongly finitary simplification bound, because (1) $C' \preceq C$ if C' is a subclause of C and (2) each clause has only finitely many subclauses. This S covers many simplification techniques,

including elimination of duplicate literals (simplifying $C \vee L \vee L$ into $C \vee L$), deletion of resolved literals (simplifying $C \vee u \not\approx u$ into C), and subsumption resolution (simplifying $C\sigma \vee D\sigma \vee L\sigma$ to $C\sigma \vee D\sigma$ given the side premise $C \vee \neg L$) Removing redundant clauses is possible with every S.

Example 66. If the Knuth–Bendix order [14] is used as the term order and all weights are positive, then $S(C \leftarrow A) = \{C' \leftarrow A' \mid C' \prec C \text{ and } A' \subseteq A\}$ is a strongly finitary simplification bound. This can be used to cover demodulation.

Example 67. The simplification rules Collect, Trim, and StrongUnsat from Sect. 3.1 are all strongly finitary simplification bounds. In a practical implementation, Split will deterministically split $C \in \mathbf{F}$ into $C_1, \ldots, C_n$ and use the same assertions $a_i \in asn(C_i)$ every time. Under these conditions, Split is also a strongly finitary simplification bound.

For other term orders, the S in Example 66 is not strongly finitary, and proving that demodulation is a strongly finitary simplification bound is much more involved. In this case the S even depends on the derivation.

Example 68. If unit equations are only removed by demodulation, reflexivity deletion, or subsumption, then the one-step demodulations possible at any point in the derivation are a strongly finitary simplification bound. (By Lemma 64, this implies that many-step demodulation is also a strongly finitary simplification bound then.) Assume that the provers performs the demodulation in a postorder traversal (i.e., subterms first), always rewriting using the oldest available equation. We also assume that if $l \approx r$ is an existing (ordered) equation and the prover derives $l \approx r'$, that $l \approx r'$ is not used for demodulation (but for example instead simplified to $r \approx r'$).

We will show that for every term t, there is only a finite number of terms t' that are simplified from t in one step. The term t' will typically be different over the course of the derivation. However, we can assign a decreasing well-founded measure to the rewrite step, ensuring finiteness. Consider a demodulation step transforming $C[l\sigma]$ to $C[r\sigma]$ using an equation $l \approx r$, with $r\sigma \prec l\sigma$. Let i be the index of $l\sigma$ in $C[l\sigma]$ in a postorder traversal, $|l|$ be the number of nonvariable positions in l, $u = 1$ if the equation is unorientable ($l \not\succ r$) and $u = 0$ otherwise. Then the tuple $(i, |l|, u, r\sigma)$ decreases or stays the same in the lexicographic order as we move along the derivation.

If the prover derives a new ordered equation $l' \approx r'$, then it is possible that it applies at an earlier position in C, thus decreasing the i. Otherwise it applies at the same position as $l \approx r$ previously, and the prover rewrites using the older $l \approx r$ first, keeping the tuple unchanged. If the equation $l \approx r$ is subsumed by $l' \approx r'$ and deleted, then $|l'| < |l|$. (Note that if $l \approx r$ is subsumed by $l \approx r'$, then r and r' are identical because all variables that occur in r, r' also occur in l.) If $l \approx r$ is simplified to $t \approx r$ by $l' \approx r'$ and $l \not\approx l'$, then $|l'| < |l|$. If $l \approx r$ is simplified to $t \approx r$ by $l \approx r'$ and $l \approx r$ is unorientable, then $|l'| = |l|$ and u decreases. If $l \approx r$ is simplified to $l \approx t$ by $l' \approx r'$, then $|l|$ stays the same, u might decrease, and $r\sigma \succ t\sigma$.

Equipped with the above definitions, we introduce a fairness criterion that is more concrete and easier to apply than the definition of fairness of $\Longrightarrow_{\mathsf{AV}}$-derivations. We could refine AV further and use this criterion to show the completeness of an imperative procedure such as Voronkov's extended Otter loop [28, Fig. 3], thus showing that AVATAR as implemented in Vampire is complete if locking is sufficiently restricted.

Lemma 69. *Let I be a strongly finitary function, and let S be a strongly finitary simplification bound. Then a well-timestamped $\Longrightarrow_{\mathsf{AV}}$-derivation $(\mathcal{J}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{Q}_i, \mathcal{L}_i)_i$ is fair if all of the following conditions hold:*

1. *$\preccurlyeq_{\mathbf{TAF}}$ is a selection order on $\bigcup_i \mathcal{P}_i$, and $\preccurlyeq_{\mathbf{F}}$ is a selection order on $\mathbf{F}$;*
2. *$\mathcal{A}_0 = \mathcal{L}_0 = \emptyset$ and $\mathcal{P}_0 \cup \mathcal{Q}_0$ is finite;*
3. *for every* Infer *transition, either $\mathcal{C}$ is $\preccurlyeq_{\mathbf{TAF}}$-minimal in $\mathcal{P}$ or $\lfloor \mathcal{C} \rfloor$ is $\preccurlyeq_{\mathbf{F}}$-minimal in $\lfloor \mathcal{P} \rfloor$;*
4. *for every* Infer *transition, $\mathcal{P}' \cup \mathcal{Q}' \subseteq I_{\mathbf{TAF}}(\mathcal{A} \cup \{\mathcal{C}\})$;*
5. *for every* Process *transition, $\mathcal{P}' \cup \mathcal{Q}' \subseteq S^*_{\mathbf{TAF}}(\mathcal{A} \cup \mathcal{P} \cup \mathcal{Q} \cup \llbracket \mathcal{L} \rrbracket)$;*
6. *if $\mathcal{J}_i \not\models (\mathcal{Q}_i)_\bot$, then eventually* Switch *or* StrongUnsat *occurs;*
7. *if $\mathcal{P}_i \neq \emptyset$, then eventually* Infer, Switch *or* StrongUnsat *occurs;*
8. *there are infinitely many indices i such that either $\mathcal{P}_i = \emptyset$ or* Infer *chooses a $\preccurlyeq_{\mathbf{TAF}}$-minimal $\mathcal{C}$ at i;*
9. *$(\limsup_{j\to\infty} \llbracket \mathcal{L}'_j \rrbracket_{\mathcal{J}}) \setminus \llbracket \mathcal{L}'^{\infty} \rrbracket_{\mathcal{J}} \subseteq \bigcup_i \mathit{FRed}_{\mathrm{F}}((\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \llbracket \mathcal{L}_i \rrbracket)_{\mathcal{J}})$ for every subsequence converging to a limit point.*

Proof. If $\bot \in \bigcup_i \wr \mathcal{Q}_i \int$, then the derivation is trivially fair. Else the StrongUnsat transition never occurs, and therefore Switch is eventually applied if the propositional clauses are not satisfied by the interpretation. Hence $\mathcal{J}_i \models \mathcal{Q}_i$ for infinitely many i. It remains to construct a subsequence $(\mathcal{J}'_j, \mathcal{A}'_j, \mathcal{P}'_j, \mathcal{Q}'_j, \mathcal{L}'_j)_j$ such that $(\mathcal{J}'_j)_j$ converges to a limit point and $\liminf_{j\to\infty} \mathit{TSInf}(\mathcal{A}'_j, \mathcal{P}'_j) = \emptyset$.

Case 1: The set of $\preccurlyeq_{\mathbf{F}}$-minimal A-formulas selected in an Infer transition for some state j is unbounded in $\preccurlyeq_{\mathbf{F}}$, that is, for every $C \in \mathbf{F}$, there is a Infer transition from state j such that the selected A-formula $\mathcal{S}_j$ is $\preccurlyeq_{\mathbf{F}}$-minimal in $\mathcal{P}_j$, and $C \preccurlyeq_{\mathbf{F}} \lfloor \mathcal{S} \rfloor$. In this case, these Infer transitions clearly form an infinite subsequence. By Lemma 32, we can further refine this subsequence of Infer transitions into a subsequence $(\mathcal{J}'_j, \mathcal{A}'_j, \mathcal{P}'_j, \mathcal{Q}'_j, \mathcal{L}'_j)_j$ where $(\mathcal{J}'_j)_j$ converges to a limit point. Assume towards a contradiction that $\iota \in \liminf_{j\to\infty} \mathit{TSInf}(\mathcal{A}'_j, \mathcal{P}'_j)$. By Lemma 60, there is a j such that $\lfloor \mathcal{C} \rfloor \preccurlyeq_{\mathbf{F}} \lfloor \mathcal{S}'_j \rfloor$ for every $\mathcal{C} \in \mathit{prems}(\iota)$. Therefore $\mathit{prems}(\iota) \subseteq \mathcal{A}'_j$ by the $\preccurlyeq_{\mathbf{F}}$-minimality requirement on the Infer transition, a contradiction.

Case 2: The set of $\preccurlyeq_{\mathbf{TAF}}$-minimal A-formulas selected in an Infer transition for some state j is unbounded in $\preccurlyeq_{\mathbf{TAF}}$; this case is analogous to case 1.

Case 3: Neither case 1 nor case 2 apply. Then the set of $\preccurlyeq_{\mathbf{F}}$-minimal formulas selected in an Infer transitions is bounded and hence finite since $\preccurlyeq_{\mathbf{F}}$ is a selection order. Similarly, the set of $\preccurlyeq_{\mathbf{TAF}}$-minimal **TAF**-formulas selected in an Infer transitions is finite as well. Let $\mathcal{T}$ be the set of A-formulas selected

in an INFER transition. So $\lfloor \mathcal{T} \rfloor$ and therefore $\bigcup_i \lfloor \mathcal{A}_i \rfloor$ are both finite. The set $S^*_{\mathbf{F}}(I_{\mathbf{F}}(\bigcup_i \lfloor \mathcal{A}_i \rfloor) \cup \lfloor \mathcal{P}_0 \rfloor \cup \lfloor \mathcal{Q}_0 \rfloor)$ is then finite, and therefore $\bigcup_i \lfloor \mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \llbracket \mathcal{L}_i \rrbracket \rfloor$ is finite as well. Since both S^* and I are strongly finitary, only a finite number of new assertions are introduced, and $\bigcup_i \wr \mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \llbracket \mathcal{L}_i \rrbracket \int$ is also finite. Thus $(\bigcup_i \mathcal{Q}_i)_\perp$ is also finite, and only a finite number of SWITCH transitions can occur. Thus there is an index N such that no SWITCH transitions occur at states $i > N$.

Now take the whole derivation as subsequence. We have $\mathcal{P}_\infty = \emptyset$ because there are infinitely many states j with a INFER transition such that a $\lessdot_{\mathbf{TAF}}$-minimal A-formula $\mathcal{S}_j$ is selected. After the initial N steps, every A-formula is selected only once (that is, $\mathcal{S}_i \neq \mathcal{S}_j$ if $i \neq j$), because once it has been selected, it can only be removed from the active set if it becomes redundant or locked. (There are no SWITCH transitions.) In either case, the A-formula is removed from the passive set for the rest of the derivation. Newly derived formulas have a different timestamp due to the well-timestampedness requirement. Therefore, once an A-formula is in the active set, it will not come back into the passive set, and we have $\liminf_{i \to \infty} \mathit{TSInf}(\mathcal{A}_i, \mathcal{P}_i) = \emptyset$. □

A slight complication is that in AVATAR as implemented by Vampire, A-clauses $C \leftarrow \{[C]\}$ are generated on a per-need basis, when switching model. However, we can imagine that the A-clauses were there all along, in the passive set.

Recall the abstact counterexample from Sect. 6.2, in which the A-formulas $\mathcal{C}$ and $\mathcal{D}$ were selected and disabled in turn. Intuitively, selection orders, together with the restrictions on the inferences, ensure that the prover will follow roughly the same steps whenever it is in a model that enables $\mathcal{C}$ and $\mathcal{D}$. Since there are only finitely many formulas that it can select for inferences before $\mathcal{C}$ or $\mathcal{D}$, the prover will eventually repeat itself and thus make progress.

7 Application to Other Architectures

AVATAR may be the most natural application of our framework, but it is not the only one. Below we complete the picture by studying splitting without backtracking, labeled splitting, and SMT with quantifiers.

7.1 Splitting without Backtracking

Before the invention of AVATAR, Riazanov and Voronkov [23] had already experimented with splitting in Vampire in a lighter variant without backtracking. They based their work on ordered resolution O with selection [2], but the same ideas work with superposition. Weidenbach [31, end of Sect. 4.5] independently outlined the same technique at about the same time.

The basic idea of splitting without backtracking is to extend the signature Σ with a countable set $\mathbb{P}$ of nullary predicate symbols disjoint from Σ and to augment the base calculus with a binary splitting rule that replaces a Σ-clause $C \vee D$ with two $\Sigma_{\mathbb{P}}$-clauses $C \vee \mathsf{p}$ and $D \vee \neg\mathsf{p}$, where C and D share no variables and

$\mathsf{p} \in \mathbb{P}$. Riazanov and Voronkov require that the precedence $\prec$ makes all $\mathbb{P}$-literals smaller than the Σ-literals. Binary splitting then qualifies as a simplification rule. They show that their rule and a few variants are satisfiability-preserving. They do not show refutational completeness, but this is obvious since the rule is a simplification.

Riazanov and Voronkov also extended the selection function of the base calculus to support $\mathbb{P}$-literals. They present two such extensions. The *blocking* function allows for the selection of $\mathbb{P}$-literals in clauses that contain Σ-literals, whereas the *parallel* function selects only maximal $\mathbb{P}$-literals in pure $\mathbb{P}$-clauses and otherwise imitates the original selection function. Parallel selection cleanly separates the $\mathbb{P}$- and the Σ-literals. Bachmair and Ganzinger proved O statically complete, and this also obviously extends to ordered resolution with this extension, which we denote by $\mathsf{O}_{\mathbb{P}}$, since it is an instance of the same calculus.

Remarkably, the calculus $\mathsf{O}_{\mathbb{P}}$ is closely related to an instance of our framework. Let $\mathbf{F}$ be the set of Σ-clauses, with the empty clause as $\bot$. Let $\mathsf{O} = (\mathit{FInf}, \mathit{FRed})$, where FInf is the set of ordered resolution inferences on $\mathbf{F}$ with some selection function and FRed is the standard redundancy criterion [2, Sect. 4.2.2], and similarly $\mathsf{O}_{\mathbb{P}} = (\mathit{FPInf}, \mathit{FPRed})$. We use the notion of entailment from Example 2 for the base relations $\models$ and $\mathrel{|\!\approx}$ for both calculi. We take $\mathbf{V} = \mathbb{P}$ for defining $\mathbf{AF}$. The properties (D1)–(D5) and (R1)–(R7) are verified for $\models$ and FRed, respectively. This gives us a splitting calculus $\mathsf{LA} = (\mathit{SInf}, \mathit{SRed})$, whose name stands for *lightweight AVATAR*. Lightweight AVATAR amounts to the splitting architecture Cruanes implemented in Zipperposition and confusingly called "AVATAR" [9, Sect. 2.5]. Binary splitting can be realized in LA as the following simplification rule:

$$\frac{C \vee D \leftarrow A}{C \leftarrow A \cup \{a\} \quad D \leftarrow A \cup \{\neg a\}}\ \textsc{BinSplit}$$

with the side conditions that $a \in \mathit{asn}(C)$ and $C \vee D$ is splittable into C, D. By Theorem 19, LA is complete.

Like splitting without backtracking but unlike the real AVATAR, Cruanes's architecture is not guided by a propositional model. It is essentially an instance of LA with the following features: It is based on superposition instead of ordered resolution, it performs branch-specific simplifications (a special case of subsumption demodulation [13]), and a SAT solver is used to detect propositional unsatisfiability and to eliminate assertions that are implied at the SAT solver's top level.

The calculi $\mathsf{O}_{\mathbb{P}}$ and LA are very similar but not identical. $\mathsf{O}_{\mathbb{P}}$ has a slightly stronger notion of inference redundancy, because its order $\prec$ can access not only the Σ-literals but also the $\mathbb{P}$-literals, whereas with LA the $\mathbb{P}$-literals are invisible to the base calculus. To see this, consider the set consisting of the $\Sigma_{\mathbb{P}}$-clauses

$$\mathsf{q} \qquad \mathsf{b} \vee \mathsf{p} \vee \neg\mathsf{q} \qquad \neg\mathsf{a} \vee \mathsf{p} \vee \neg\mathsf{q} \qquad \mathsf{a} \vee \mathsf{p} \vee \neg\mathsf{q}$$

where $\mathbb{P} = \{\mathsf{a}, \mathsf{b}\}$. Given the precedence $\mathsf{a} \prec \mathsf{b} \prec \mathsf{p} \prec \mathsf{q}$, an ordered resolution inference is possible between the first two clauses, with $\mathsf{b} \vee \mathsf{p}$ as its conclusion. This

inference is redundant according to $FPRed_{\mathrm{I}}$, because the conclusion is entailed by the first, third, and fourth clauses taken together, all of which are $\prec$-smaller than the main premise $\mathsf{b} \vee \mathsf{p} \vee \neg \mathsf{q}$. However, the corresponding **AF**-inference is not redundant according to $SRed_{\mathrm{I}}$, because the assertions are simply truncated by the projection operator $(\ )_{\mathcal{J}}$ and are not compared. Without the assertions, the third and fourth clauses are equal to—but not smaller than—the main premise, and the inference is not redundant. Note that the set is not saturated: Inferences are possible to derive $\neg \mathsf{a} \vee \mathsf{p}$ and $\mathsf{a} \vee \mathsf{p}$, which make $\mathsf{b} \vee \mathsf{p}$ redundant.

Another dissimilarity is that LA can detect unsatisfiability immediately using a SAT solver, whereas splitting without backtracking generally needs many propositional resolution steps to achieve the same result. Correspondingly, on satisfiable problems, LA also allows smaller saturated sets. For example, while the A-clause set $\{\bot \leftarrow \{\mathsf{a}, \neg \mathsf{b}\}, \bot \leftarrow \{\mathsf{b}\}\}$ is saturated, its $\mathsf{O}_{\mathbb{P}}$ counterpart is subject to an inference between $\neg \mathsf{a} \vee \mathsf{b}$ and $\neg \mathsf{b}$.

As positive results, we will show that $\mathsf{O}_{\mathbb{P}}$ and LA share the same notion of entailment and $\mathsf{O}_{\mathbb{P}}$'s redundancy criterion is stronger than LA's, yet saturation w.r.t. LA guarantees saturation w.r.t. $\mathsf{O}_{\mathbb{P}}$, up to the natural correspondence between A-clauses and $\Sigma_{\mathbb{P}}$-clauses. More precisely, a $\Sigma_{\mathbb{P}}$-clause can be written as $C \vee L_1 \vee \cdots \vee L_n$, where C is a Σ-clause and the L_i's are $\mathbb{P}$-literals. Let α be a bijective mapping such that $\alpha(C \vee L_1 \vee \cdots \vee L_n) = C \leftarrow \{\neg L_1, \ldots, \neg L_n\}$ is the corresponding A-clause. We overload the operator $\lfloor\ \rfloor$ to erase the $\mathbb{P}$-literals: $\lfloor C \vee L_1 \vee \cdots \vee L_n \rfloor = \lfloor \alpha(C \vee L_1 \vee \cdots \vee L_n) \rfloor = C$. Moreover, let $\mathcal{G}$ denote the function that returns all ground instances of a clause, clause set, or inference according to Σ (or, equivalently, $\Sigma_{\mathbb{P}}$), which is assumed to contain at least one constant.

Lemma 70. *Given the $\Sigma_{\mathbb{P}}$-clause sets M, N, we have $M \models N$ if and only if $\alpha(M) \models \alpha(N)$.*

Proof. Since both entailments are defined via grounding, it suffices to consider the case where M and N are ground.

For the forward direction, we must show that $(\alpha(M))_{\mathcal{J}} \models \lfloor N \rfloor$ for some $\mathcal{J}$ in which $\alpha(N)$ is enabled. Let $\mathcal{K}$ be a Σ-model of $\{\alpha(M)\}_{\mathcal{J}}$. We will show that at least one clause in $\lfloor N \rfloor$ is true in $\mathcal{K}$. We start by showing that $\mathcal{K} \cup \mathcal{J}$ is a $\Sigma_{\mathbb{P}}$-model of M. Let $C \in M$. If $\alpha(C)$ is enabled in $\mathcal{J}$, then $\lfloor C \rfloor \in (\alpha(M))_{\mathcal{J}}$. Thus $\mathcal{K} \models \lfloor C \rfloor$ and finally $\mathcal{K} \cup \mathcal{J} \models \{C\}$. Otherwise, $\alpha(C)$ contains an assertion that is false in $\mathcal{J}$, which means that C contains the complementary $\mathbb{P}$-literal, which is true in $\mathcal{J}$, and we have $\mathcal{K} \cup \mathcal{J} \models \{C\}$. Either way, $\mathcal{K} \cup \mathcal{J} \models M$ and hence, since $M \models N$, one of the clauses in N is true in $\mathcal{K} \cup \mathcal{J}$. Since $\alpha(N)$ is enabled in $\mathcal{J}$, all $\mathbb{P}$-literals occurring in N are false in $\mathcal{K} \cup \mathcal{J}$. Therefore, each clause in N must contain a true Σ-literal in $\mathcal{K}$, which means that the corresponding clause in $\lfloor N \rfloor$ must also be true in $\mathcal{K}$.

For the backward direction, we must show that $M \models N$. Let $\mathcal{K} \cup \mathcal{J}$ be a $\Sigma_{\mathbb{P}}$-model of M, where $\mathcal{K}$ is a Σ-interpretation and $\mathcal{J}$ is a $\mathbb{P}$-interpretation. We will show that a clause in N is true in $\mathcal{K} \cup \mathcal{J}$. If $\alpha(N)$ is disabled in $\mathcal{J}$, there exists a $\mathbb{P}$-literal in some clause from N that is true in $\mathcal{K} \cup \mathcal{J}$, which suffices to make

the entire clause true. Otherwise, N is enabled in $\mathcal{J}$ and then $(\alpha(M))_{\mathcal{J}} \models \lfloor N \rfloor$. Since $\mathcal{K} \cup \mathcal{J} \models M$, we have $\mathcal{K} \models (\alpha(M))_{\mathcal{J}}$. Hence, one of the clauses in $\lfloor N \rfloor$ is true in $\mathcal{K}$, and its counterpart in N is also true in $\mathcal{K} \cup \mathcal{J}$. □

Lemma 71. *Given an inference ι over $\Sigma_{\mathbb{P}}$-clauses, if $\alpha(\iota)$ is a* Base *inference, then $\iota \in \mathit{FPInf}$.*

Proof. Let $\iota = (C_n, \ldots, C_1, C_0)$ and assume $\alpha(\iota)$ is a Base inference. By the definition of ordered resolution, none of the Σ-clauses $\lfloor C_i \rfloor$, for $i \in \{1, \ldots, n\}$, can be $\bot$. Thus, the selected literals in the premises coincide with those chosen by the parallel selection function on the $\Sigma_{\mathbb{P}}$-clauses C_i and so $\iota \in \mathit{FPInf}$. □

Lemma 72. (1) *Given a $\Sigma_{\mathbb{P}}$-clause C, if $\alpha(C) \in \mathit{SRed}_{\mathrm{F}}(\alpha(N))$, then $C \in \mathit{FPRed}_{\mathrm{F}}(N)$.* (2) *Given an inference ι over $\Sigma_{\mathbb{P}}$-clauses, if $\alpha(\iota) \in \mathit{SRed}_{\mathrm{I}}(\alpha(N))$, then $\iota \in \mathit{FPRed}_{\mathrm{I}}(N)$.*

Proof. For (2), let $\iota' \in \mathcal{G}(\iota)$ and let $C_n, \ldots, C_1$ be ι''s premises and C_0 be its conclusion. Let $\mathcal{E} = \{C \leftarrow A \in \alpha(\mathcal{G}(N)) \mid C \prec \lfloor C_1 \rfloor\}$ and $\mathcal{F} = \{C \in N \mid \mathcal{G}(C) \prec C_1\}$. By the definition of standard redundancy, assuming that $\{\alpha(C_n), \ldots, \alpha(C_2)\} \cup \mathcal{E} \models \{\alpha(C_0)\}$ we need to show that $\{C_n, \ldots, C_2\} \cup \mathcal{F} \models \{C_0\}$. By Lemma 70, this amounts to showing that $\{\alpha(C_n), \ldots, \alpha(C_2)\} \cup \alpha(\mathcal{F}) \models \{\alpha(C_0)\}$. By (D3), this follows from our assumption if $\mathcal{E} \subseteq \alpha(\mathcal{F})$. This subset inclusion holds because if $C \prec \lfloor C_1 \rfloor$, then we have $C \vee D \prec C_1$ for every $\mathbb{P}$-clause D, since $\mathbb{P}$-literals are smaller than Σ-literals.

For (1), essentially the same line of argumentation applies, with $n = 1$ and $C_1 = C_0 = C$. □

Lemma 73 (Saturation). *Let N be a set of $\Sigma_{\mathbb{P}}$-clauses. If N is saturated w.r.t. $\mathsf{O}_{\mathbb{P}}$, then $\alpha(N)$ is saturated w.r.t.* LA.

Proof. Let $\iota = (\alpha(C_n), \ldots, \alpha(C_1), \alpha(C_0)) \in \mathit{SInf}$ be an inference with premises in $\alpha(N)$. We will show that it is redundant w.r.t. $\alpha(N)$.

Case Base: We need to show that $\{\iota\}_{\mathcal{J}} \subseteq \mathit{FRed}_{\mathrm{I}}((\alpha(\mathcal{G}(N)))_{\mathcal{J}})$ for all propositional interpretations $\mathcal{J}$. The case where ι is disabled in $\mathcal{J}$ is trivial. Otherwise, let θ be a substitution such that $\iota\theta \in \mathcal{G}(\iota)$. We must show that $\{\lfloor C_n\theta \rfloor, \ldots, \lfloor C_2\theta \rfloor\} \cup \lfloor \mathcal{E} \rfloor \models \{\lfloor C_0\theta \rfloor\}$, where $\mathcal{E} = \{\alpha(C) \mid C \in \mathcal{G}(N) \text{ and } \lfloor C \rfloor \prec \lfloor C_1\theta \rfloor\}$. Because the premises' assertions are contained in the conclusion's, this is equivalent to showing that $\{\alpha(C_n\theta), \ldots, \alpha(C_2\theta)\} \cup \mathcal{E} \models \{\alpha(C_0\theta)\}$.

By Lemma 71, there exists an inference $(C_n, \ldots, C_1, C_0) \in \mathit{FPInf}$. Since N is saturated, the inference is redundant—i.e., $\{C_n\theta, \ldots, C_2\theta\} \cup \mathcal{F} \models \{C_0\theta\}$, where $\mathcal{F} = \{C \mid C \in \mathcal{G}(N) \text{ and } C \prec C_1\theta\}$. If $\alpha(\mathcal{F}) \subseteq \mathcal{E}$, we can invoke Lemma 70 to conclude. However, in the general case, we have only that $\alpha(\mathcal{F} \setminus \mathcal{F}_{\mathrm{eq}}) \subseteq \mathcal{E}$, where $\mathcal{F}_{\mathrm{eq}} = \{C \in \mathcal{F} \mid \lfloor C \rfloor = \lfloor C_1\theta \rfloor\}$, and thus there might be models of $\mathcal{E}$ that are models of $\mathcal{F} \setminus \mathcal{F}_{\mathrm{eq}}$ but not of $\mathcal{F}_{\mathrm{eq}}$. Fortunately, we can show that $\{C_n\theta, \ldots, C_2\theta\} \cup (\mathcal{F} \setminus \mathcal{F}_{\mathrm{eq}}) \models \{C_0\theta\}$. We proceed by removing from $\mathcal{F}$ each clause $D \in \mathcal{F}_{\mathrm{eq}}$ in turn and by showing that the entailment is preserved by each step. Finally, we invoke Lemma 70. A slight complication is that $\mathcal{F}_{\mathrm{eq}}$ may be

infinite. However, by compactness, only a finite subset $\mathcal{F}_{\text{eq}}^{\text{fin}} \subseteq \mathcal{F}_{\text{eq}}$ is needed to have the desired entailment.

Let $D \in N$ be a clause that generalizes the ground, $\succ$-largest clause in $\mathcal{F}_{\text{eq}}^{\text{fin}}$. Then there exists an inference $(C_n, \ldots, C_2, D, D_0) \in \mathit{FPInf}$ such that $\lfloor C_0\theta \rfloor \in \lfloor \mathcal{G}(D_0) \rfloor$ and the $\mathbb{P}$-literals of D_0 are the union of those of $C_n, \ldots, C_2, D$. By renaming the variables in D and D_0, we can ensure that $\lfloor D\theta \rfloor = \lfloor C_1\theta \rfloor$ and $\lfloor D_0\theta \rfloor = \lfloor C_0\theta \rfloor$. Now, to prove the desired entailment, assume that $\mathcal{J}$ is a model of $\{C_n\theta, \ldots, C_2\theta\} \cup (\mathcal{F} \setminus \{D\theta\})$. Since N is saturated, $\{C_n\theta, \ldots, C_2\theta\} \cup \{C \in \mathcal{G}(N) \mid C \prec D\theta\} \models \{D_0\theta\}$. Since we are proceeding from largest to smallest clause, we have $\{C \in \mathcal{G}(N) \mid C \prec D\theta\} \subseteq \mathcal{F} \setminus \{D\theta\}$, even if some clauses have been removed from $\mathcal{F}$ already. Thus, in both cases, $\mathcal{J} \models \{D_0\theta\}$. If $\mathcal{J}$ makes a Σ-literal of $D_0\theta$ true, $\mathcal{J}$ makes the same literal in $C_0\theta$ true. Otherwise, either $\mathcal{J}$ makes one of the $\mathbb{P}$-literals of $C_n\theta, \ldots, C_2\theta$ true, satisfying $C_0\theta$ for the same reason, or it makes one of the $\mathbb{P}$-literals of D true and then $\mathcal{J} \models \{C_n\theta, \ldots, C_2\theta\} \cup \mathcal{F}$, which as noted above implies $\mathcal{J} \models \{C_0\theta\}$ by the saturation of N. In both cases, $\mathcal{J} \models \{C_0\theta\}$.

Case Unsat: The inference derives $\bot$ from a set of $\mathbb{P}$-clauses $(\alpha(A_i))_{i=1}^n$ such that $\{\alpha(A_i)\}_i$ is propositionally unsatisfiable—i.e., $\{A_i\}_i \models \{\bot\}$ in $\mathsf{O}_{\mathbb{P}}$. Since $\mathsf{O}_{\mathbb{P}}$ is complete and $N \supseteq \{A_i\}_i$ is saturated, we have $\bot \in N$ and hence $\bot \in \alpha(N)$. Therefore, ι is redundant also in this case. □

7.2 Labeled Splitting

Labeled splitting, as originally described by Fietzke and Weidenbach [11] and implemented in SPASS, is a first-order resolution-based calculus with binary splitting that traverses the split tree in a depth-first way, using an elaborate backtracking mechanism inspired by CDCL [18]. It works on states $(\Psi, \mathcal{N})$, where Ψ is a stack storing the current state of the split tree and $\mathcal{N}$ is a set of *labeled clauses*—clauses annotated with finite sets of natural numbers.

We model labeled splitting as an instance of the locking prover L based on the splitting calculus $\mathsf{LS} = (\mathit{SInf}, \mathit{SRed})$ induced by the resolution calculus $\mathsf{R} = (\mathit{FInf}, \mathit{FRed})$, where $\models$ and $\mathrel{|\!\approx}$ are as in Example 2 and $\mathbf{V} = \bigcup_{i \in \mathbb{N}} \{\mathsf{l}_i, \mathsf{r}_i, \mathsf{s}_i\}$. A-clauses correspond to labeled clauses. Splits are identified by unique *split levels*. Given a split on $C \vee D$ with level k, $\mathsf{l}_k \in \mathit{asn}(C)$ and $\mathsf{r}_k \in \mathit{asn}(D)$ represent the left and right branch. In practice, the prover would dynamically extend *fml* so that $\mathit{fml}(\mathsf{l}_k) = C$ and $\mathit{fml}(\mathsf{r}_k) = D$.

When splitting, if we simply added the propositional clause $\bot \leftarrow \{\neg\mathsf{l}_k, \neg\mathsf{r}_k\}$, we would always need to consider either $C \leftarrow \{\mathsf{l}_k\}$ or $D \leftarrow \{\mathsf{r}_k\}$, depending on the interpretation. However, labeled splitting can undo splits when backtracking. Yet fairness would require us to perform inferences with either C or D even when labeled splitting would not. We solve this as follows. Let $\top = \sim\bot$. We introduce the propositional variable $\mathsf{s}_k \in \mathit{asn}(\top)$ so that we can enable or disable the split as we wish. The StrongUnsat rule then knows that s_k is true and that the case distinction is exhaustive, but we can still switch to propositional models that disable both C and D. A-clauses are then split using the following binary variant of Split:

$$\frac{C \vee D \leftarrow A}{\bot \leftarrow \{\neg \mathsf{l}_k, \neg \mathsf{r}_k, \mathsf{s}_k\} \quad C \leftarrow A \cup \{\mathsf{l}_k\} \quad D \leftarrow A \cup \{\mathsf{r}_k\}}\ \textsc{SoftSplit}$$

where C and D share no variables and k is the next split level. Unlike AVATAR, labeled splitting keeps the premise and might split it again with another level. We rely on locking to ensure that the premise is not split within either branch.

To emulate the original, the locking prover based on the LS calculus must repeatedly apply the following three steps in any order until saturation:

1. Apply Base (via Lift and Derive) to perform an inference from the enabled A-clauses. If an enabled $\bot \leftarrow A$ is derived with $A \subseteq \bigcup_i \{\mathsf{l}_i, \mathsf{r}_i\}$, apply Switch or StrongUnsat.
2. Apply Base to simplify or delete an enabled A-clause. Use Lock if necessary to remove the original A-clause. If an enabled $\bot \leftarrow A$ is derived, apply Switch or StrongUnsat.
3. Apply SoftSplit with split level k on an A-clause $\mathcal{C}$. Then use Switch to enable the left branch and apply Lock on $\mathcal{C}$ with s_k as the lock.

Disabled A-clauses are the ones that occur in branches other than the current one and in disabled splits. As such, they should not be used when exploring the current branch.

Switch is powerful enough to support all of Fietzke and Weidenbach's backtracking rules, but to explore the tree in the same order as they do, we must choose the new model carefully. If a left branch is closed, the model must be updated so as to disable the splits that were not used to close this branch and to enable the right branch. If a right branch is closed, the split must be disabled, and the model must switch to the right branch of the closest enabled split above it with an enabled left branch. If a right branch is closed but there is no split above with an enabled left branch, the entire tree has been visited. Then, a propositional clause $\bot \leftarrow A$ with $A \subseteq \bigcup_i \{\mathsf{s}_i\}$ is $\models$-entailed by the A-clause set, and StrongUnsat can finish the refutation by exploiting $\mathit{fml}(\mathsf{s}_i) = \top$.

We illustrate the strategy on an example.

Example 74. Let $N = \{\mathsf{p}(x) \vee \mathsf{q}(y), \neg\mathsf{p}(x) \vee \mathsf{q}(y), \neg\mathsf{q}(x), \mathsf{r}(x) \vee \mathsf{s}(y)\}$ be a set of clauses. It is unsatisfiable due to the first three clauses. Let $\mathcal{N}_0 = N$ be the input set and $\mathcal{J}_0 = \{\neg v \mid v \in \mathbf{V}\}$ be the initial model.

A split in the tree is achieved by the succession of two Lift—one of SoftSplit and one of Switch, and a Lock rule applications. Here, to start with, the first clause is split into $\mathsf{p}(x) \leftarrow \mathsf{l}_0$, $\mathsf{q}(y) \leftarrow \mathsf{r}_0$ and $\bot \leftarrow \{\mathsf{s}_0, \neg\mathsf{l}_0, \neg\mathsf{r}_0\}$ by SoftSplit, then a Switch replaces $\mathcal{J}_0$ with $\{\mathsf{s}_0, \mathsf{l}_0\} \cup \mathcal{J}_0 \setminus \{\neg\mathsf{s}_0, \neg\mathsf{l}_0\}$. This has the effect of enabling $\mathsf{p}(x) \leftarrow \mathsf{l}_0$, that is in the explored branch. Then Lock is invoked and removes $\mathsf{p}(x) \vee \mathsf{q}(y)$ from $\mathcal{N}_0$ for as long as s_0 is enabled. By splitting in a similar

fashion the clauses $\mathsf{r}(x) \vee \mathsf{s}(y)$ and then $\neg\mathsf{p}(x) \vee \mathsf{q}(y)$, the resulting state is

$$\begin{aligned}
\mathcal{J}_1 &= \{\mathsf{s}_0, \mathsf{l}_0, \mathsf{s}_1, \mathsf{l}_1, \mathsf{s}_2, \mathsf{l}_2\} \cup \mathcal{J}_0 \setminus \{\neg\mathsf{s}_0, \neg\mathsf{l}_0, \neg\mathsf{s}_1, \neg\mathsf{l}_1, \neg\mathsf{s}_2, \neg\mathsf{l}_2\} \\
\mathcal{N}_1 &= \{\neg\mathsf{q}(x),\ \mathsf{p}(x) \leftarrow \mathsf{l}_0,\ \mathsf{r}(x) \leftarrow \mathsf{l}_1,\ \neg\mathsf{p}(x) \leftarrow \mathsf{l}_2\} \\
&\quad \cup \{\bot \leftarrow \mathsf{s}_0, \neg\mathsf{l}_0, \neg\mathsf{r}_0,\ \bot \leftarrow \mathsf{s}_1, \neg\mathsf{l}_1, \neg\mathsf{r}_1,\ \bot \leftarrow \mathsf{s}_2, \neg\mathsf{l}_2, \neg\mathsf{r}_2\} \\
&\quad \cup \{\mathsf{q}(y) \leftarrow \mathsf{r}_0,\ \mathsf{s}(y) \leftarrow \mathsf{r}_1,\ \mathsf{q}(y) \leftarrow \mathsf{r}_2\} \\
\mathcal{L}_1 &= \{(\mathsf{s}_0, \mathsf{p}(x) \vee \mathsf{q}(x)), (\mathsf{s}_1, \mathsf{r}(x) \vee \mathsf{s}(y)), (\mathsf{s}_2, \neg\mathsf{p}(x) \vee \mathsf{q}(y))\}
\end{aligned}$$

where the last three clauses listed in $\mathcal{N}_1$ are disabled, and thus currently unusable for inferences.

The first backtracking step happens after a Lift of Base produces the A-clause $\bot \leftarrow \mathsf{l}_0, \mathsf{l}_2$ from $\mathsf{p}(x) \leftarrow \mathsf{l}_0$ and $\neg\mathsf{p}(x) \leftarrow \mathsf{l}_2$. At that point Switch is called. It disables s_1 because this split was not useful in closing the branch and it swaps exploring branch l_2 for branch r_2, leaving the rest of the model unchanged. The resulting model is thus $\mathcal{J}_2 = \{\mathsf{s}_0, \mathsf{l}_0, \mathsf{s}_2, \mathsf{r}_2\} \cup \mathcal{J}_0 \setminus \{\neg\mathsf{s}_0, \neg\mathsf{l}_0, \neg\mathsf{s}_2, \neg\mathsf{r}_2\}$. This model disables $\neg\mathsf{p}(x) \leftarrow \mathsf{l}_2$ and enables $\mathsf{q}(y) \leftarrow \mathsf{r}_2$. It also unlocks $\mathsf{r}(x) \vee \mathsf{s}(y)$.

The second backtracking step is reached after $\bot \leftarrow \mathsf{r}_2$ is added to the set of A-clauses by resolving $\neg\mathsf{q}(x)$ and $\mathsf{q}(y) \leftarrow \mathsf{r}_2$. Since both branches of the split s_2 have now been closed, the Switch rule is invoked, producing the model $\mathcal{J}_3 = \{\mathsf{s}_0, \mathsf{r}_0\} \cup \mathcal{J}_0 \setminus \{\neg\mathsf{s}_0, \neg\mathsf{r}_0\}$. This has the effect of unlocking the clause $\neg\mathsf{p}(x) \vee \mathsf{q}(y)$ and now only $\mathsf{q}(y) \leftarrow \mathsf{r}_0$ is enabled in addition to the unlocked input clauses.

This branch is immediately closed by the generation of $\bot \leftarrow \mathsf{r}_0$ from $\mathsf{q}(y) \leftarrow \mathsf{r}_0$ and $\neg\mathsf{q}(x)$. The resulting set $\mathcal{N}_4$ includes in particular $\bot \leftarrow \mathsf{r}_0$, $\bot \leftarrow \mathsf{r}_2$ and $\bot \leftarrow \mathsf{l}_0, \mathsf{l}_2$, obtained by Infer steps as well as $\bot \leftarrow \mathsf{s}_0, \neg\mathsf{l}_0, \neg\mathsf{r}_0$ and $\bot \leftarrow \mathsf{s}_2, \neg\mathsf{l}_2, \neg\mathsf{r}_2$, obtained by Split steps. Hence it entails $\bot \leftarrow \mathsf{s}_0, \mathsf{s}_2$. Since $\mathit{fml}(\mathsf{s}_0) = \mathit{fml}(\mathsf{s}_2) = \sim\bot$, StrongUnsat's condition holds, so this rule can be invoked and derive $\bot$, concluding the refutation.

By following this strategy, LS closely simulates the original calculus, i.e., it is possible to add and remove (or at least disable) exactly the same elements to the A-clause set as is done in the original and in the same order. A subtle difference lies in the backtracking rule: Labeled splitting can backtrack to branches where $\bot$ is enabled, while we require that the prover directly switches to a model where all propositional clauses are satisfied. But what about fairness?

The above strategy helps achieve fairness, because it ensures that there exists exactly one limit point. The strategy also uses locks in a friendly way. This means we can simplify the notion of fairness for $\Longrightarrow_{\mathsf{L}}$-derivations considerably and obtain a criterion that is almost identical to, but slightly more liberal than, Fietzke and Weidenbach's—thereby re-proving the completeness of labeled splitting.

For terminating derivations, their fairness criterion coincides with ours—both require that the final A-clause set is locally saturated and all propositional clauses are satisfied by the interpretation. For diverging derivations, Fietzke and Weidenbach construct a limit subsequence $(\Phi'_i, \mathcal{N}'_i)_i$ of the derivation $(\Phi_i, \mathcal{N}_i)_i$ and require that every persistent inference in it be made redundant, exactly as we do for $\Longrightarrow_{\mathsf{L}}$-derivations. The subsequence consists of all states that lie on the split tree's unique infinite branch. Therefore, this subsequence converges to a limit

point of the full derivation. Locks are well behaved, with $\limsup_{j\to\infty} \|\mathcal{L}'_j\| = \|\mathcal{L}'^{\infty}\|$, because with the strategy above, once an A-clause is enabled on the rightmost branch, it remains enabled forever. Our definition of fairness allows more subsequences, although this is difficult to exploit without bringing in all the theoretical complexity of AVATAR.

Example 75. Alternating age-based and unrestricted heuristic selection is not complete for labeled splitting, just like it is incomplete for AVATAR (Example 51). Start with the clause set $\{\mathsf{p}(\mathsf{0}, y), \neg\mathsf{p}(x, y) \vee \mathsf{p}(\mathsf{s}(x), y), \neg\mathsf{p}(x, y) \vee \mathsf{q}(y) \vee \mathsf{r}(x), \neg\mathsf{q}(x) \vee \mathsf{s}(x), \neg\mathsf{s}(x), \mathsf{q}(\mathsf{0})\}$, and always select the negative literal if there is one. The prover begins by deriving $\mathsf{p}(\mathsf{s}(\mathsf{0}), y)$ and $\mathsf{q}(y) \vee \mathsf{r}(\mathsf{0})$ using the age-based heuristic. Then it heuristically selects $\mathsf{q}(y) \vee \mathsf{r}(\mathsf{0})$ for inferences and splits it. In the left branch where $\mathsf{q}(y)$ is enabled, $\mathsf{q}(\mathsf{0})$ is locally redundant and locked. Before age-based selection allows the prover to derive $\bot$ from $\mathsf{q}(y), \neg\mathsf{q}(x) \vee \mathsf{s}(x)$, and $\neg\mathsf{s}(x)$, it will also have derived $\mathsf{p}(\mathsf{s}(\mathsf{s}(\mathsf{0})), y)$ and $\mathsf{q}(y) \vee \mathsf{r}(\mathsf{s}(\mathsf{0}))$. When the prover switches back to the right branch, it can heuristically select this clause and split it, continuing as before.

In this way, only split inferences are performed on the rightmost branch. The clause $\mathsf{q}(\mathsf{0})$, which is necessary for a refutation, is never selected for inferences (most of the time it is even locally redundant).

7.3 SMT with Quantifiers

Satisfiability modulo theories (SMT) solvers based on DPLL(T) [18] combine a SAT solver with theory solvers, each responsible for reasoning about a specific quantifier-free theory (e.g., equality, linear integer arithmetic). In the classical setup, the theories are decidable, and the SMT solver is a decision procedure for the union of the theories. Some SMT solvers, including CVC4 [3], veriT [8], and Z3 [16], also support quantified formulas via instantiation at the expense of decidability.

Complete instantiation strategies have been developed for various fragments of first-order logic [12, 21, 22]. In particular, enumerative quantifier instantiation [21] is complete under some conditions. An SMT solver following such a strategy ought to be refutationally complete, but this has never been proved. Although SMT is quite different from the architectures we have studied so far, we can instantiate our framework to show the completeness of an abstract SMT solver. The model-guided prover MG will provide a suitable starting point, since we will need neither L's locking mechanism nor AV's given clause procedure.

Let $\mathbf{F}$ be the set of first-order Σ-formulas with a distinguished falsehood $\bot$. We represent the SMT solver's underlying SAT solver by the Unsat rule and complement it with an inference system *FInf* that clausifies formulas, detects inconsistencies up to theories excluding quantifiers, and instantiates quantifiers. For *FRed*, we take an arbitrary instance of the standard redundancy criterion. Its only purpose is to split disjunctions destructively. We define the "theories with quantifiers" calculus $\mathsf{TQ} = (\mathit{FInf}, \mathit{FRed})$. For the consequence relations $\models$ and $\mathrel{|\!\approx}$, we use entailment in the supported theories including quantifiers.

The clausification rules work on logical symbols outside quantifiers; they derive C and D from a premise $C \wedge D$, among others. The theory rules can derive $\bot$ from some finite formula set N if $N \models \{\bot\}$, ignoring quantifiers; this triggers a model switch. Finally, the instantiation rules derive formulas $\mathsf{p}(t)$ from premises $\forall x.\, \mathsf{p}(x)$, where t is some ground term; the instantiation strategy determines which ground terms must be tried and in which order. A lot of complexity hidden in *FInf*—such as purification and theory-specific data structures and algorithms—is taken as a black box.

As with AVATAR, the initial problem is expressed using Σ-formulas. We use the same approximation function as in AVATAR to represent formulas as assertions (Example 8). Abusing terminology slightly, let us call an A-formula $C \leftarrow A$ a *subunit* if C is not a disjunction. Whenever a disjunction $C \vee D \leftarrow A$ emerges, we immediately apply Split. This delegates clausal reasoning to the SAT solver. It then suffices to assume that TQ is complete for subunits.

Theorem 76 (Dynamic completeness). *Assume* TQ *is statically complete for subunit sets. Let* $(\mathcal{J}_i, \mathcal{N}_i)_i$ *be a fair* $\Longrightarrow_{\mathsf{MG}}$*-derivation based on* TQ*. If* $\mathcal{N}_0 \models \{\bot\}$ *and* $\mathcal{N}_\infty$ *contains only subunits, then* $\bot \in \mathcal{N}_j$ *for some* j*.*

Proof. The proof is analogous to that of Theorem 26. Because we only have conditional static completeness of $(\mathit{FInf}, \mathit{FRed})$, we need the assumption that $\mathcal{N}_\infty$ contains only subunits. □

Care must be taken to design a practical fair strategy. Like AVATAR-based provers, SMT solvers will typically not perform all *SInf*-inferences, not even up to SRed_I. Given $\mathsf{a} \approx \mathsf{b} \leftarrow \{\mathsf{v}_0\}$, $\mathsf{b} \approx c \leftarrow \{\mathsf{v}_1\}$, $\mathsf{a} \approx \mathsf{d} \leftarrow \{\mathsf{v}_2\}$, $\mathsf{c} \approx \mathsf{d} \leftarrow \{\mathsf{v}_3\}$, and $\mathsf{a} \not\approx \mathsf{c} \leftarrow \{\mathsf{v}_4\}$, it will find only one of the conflicts $\bot \leftarrow \{\mathsf{v}_0, \mathsf{v}_1, \mathsf{v}_4\}$ or $\bot \leftarrow \{\mathsf{v}_2, \mathsf{v}_3, \mathsf{v}_4\}$ but not both. This leaves us in a similar predicament as with locking: A theory conflict might be nonredundant at the limit point, even though it is redundant at every point of the derivation. The SMT solver just happened to choose the wrong conflict every time.

Example 77. Consider $\mathcal{N}_0 = \{\forall x\, (x \leq 0 \vee \mathsf{a} > x), \mathsf{a} = 0, \mathsf{a} + 3 < 2\}$. Eagerly applying quantifier instantiation, we get the instances $0 \leq 0 \leftarrow \{[0 \leq 0]\}$, $\mathsf{a} > 0 \leftarrow \{[\mathsf{a} > 0]\}$, $\bot \leftarrow \{\neg[0 \leq 0], \neg[\mathsf{a} > 0]\}$, $1 \leq 0 \leftarrow \{[1 \leq 0]\}$, $\mathsf{a} > 1 \leftarrow \{[\mathsf{a} > 1]\}$, $\bot \leftarrow \{\neg[1 \leq 0], \neg[\mathsf{a} > 1]\}$, etc. The solver then starts in a model where $[\mathsf{a} > 0], [\mathsf{a} > 1]$, etc. are true. Here it can derive the conflict $\bot \leftarrow \{[\mathsf{a} > 0]\}$. Then it switches to the next model where $[\mathsf{a} > 0]$ is false, but $[\mathsf{a} > 1], [\mathsf{a} > 2]$, etc. are true, and derive the conflict $\bot \leftarrow \{[\mathsf{a} > 1]\}$.

Continuing in this way, all conflicts are of the form $[\mathsf{a} > n]$ for some n. However, at the limit point—where $[\mathsf{a} > n]$ is false for all n—none of these conflicts is enabled. The only conflict which exists at the limit point is between $\mathsf{a} = 0$ and $\mathsf{a} + 3 < 2$, and the solver never found it because it stumbled upon a different conflict first.

For decidable theories, a practical fair strategy is to first clausify and detect theory conflicts and to instantiate quantifiers only if no other rules are applicable. A similar case distinction as in Lemma 69 works to establish fairness for

this strategy: first consider the case where quantifier instantiation is invoked infinitely often. Hence there is an infinite subsequence $(\mathcal{J}'_j, \mathcal{N}'_j)_j$ of states such that (1) no $\mathcal{N}'_j$ has a theory conflict, and (2) $(\mathcal{J}'_j)_j$ converges to a limit point. For fairness of the $\Longrightarrow_{\mathsf{MG}}$-derivation we need to show that $\iota \in \mathit{FInf}((\mathcal{N}_\infty)_\mathcal{J})$ implies $\iota \in \mathit{FRed}_\mathrm{I}((\mathcal{N}_i)_\mathcal{J})$ for all ι. If $\mathit{concl}(\iota) = \bot$, then there is a theory conflict and because ι has only finitely many premises, these premises are in $\mathcal{N}'_j$ for some j, a contradiction to the strategy. Quantifier instantiation accounts for the remaining inferences ι with $\mathit{concl}(\iota) \neq \bot$. Here, it suffices to ensure that A-formulas that are enabled infinitely often at a quantifier instantiation step are also fully instantiated.

In the other case quantifier instantiation is only invoked finitely often (either because every encountered model had a theory conflict, or because there was nothing to instantiate). This case reduces to the completeness of the SMT solver without quantifier instantiation, which must be established separately. And just as with AV-provers, it is possible that not all limit points are saturated.

There is also the question of model soundness. If the SMT solver starts with the Σ-formula set $\mathcal{N}_0$ and terminates in a state $(\mathcal{J}_i, \mathcal{N}_i)$, with $\mathcal{J}_i \models (\mathcal{N}_i)_\bot$, we would like the solver to generated a model of $(\mathcal{N}_i)_{\mathcal{J}_i}$, from which a model of $\mathcal{N}_0$ can be derived. This is possible if the solver performs only sound inferences and applies Approx systematically. Then $(\mathcal{N}_i)_{\mathcal{J}_i}$ is fully exposed to the propositional level, and $\mathit{fml}(\mathcal{J}_i)$ is a theory model of $\mathcal{N}_{\mathcal{J}_i}$ and therefore of $\mathcal{N}_0$.

Example 78. Consider the initial clause set $\mathcal{N}_0 = \{\forall x\, (\mathsf{f}(x) = 0 \vee \mathsf{g}(x) = 0), \mathsf{f}(1) = 1, \mathsf{g}(1) = 1\}$. The SMT solver first considers the propositional model $\mathcal{J}_0 = \neg\mathbf{V}$. There is no theory conflict, so the solver uses quantifier instantiation and clausification to derive $\mathcal{N}_1 = \mathcal{N}_0 \cup \{\mathsf{f}(0) = 0 \leftarrow \{\mathsf{v}_0\}, \mathsf{g}(0) = 0 \leftarrow \{\mathsf{v}_1\}, \bot \leftarrow \{\neg\mathsf{v}_0, \neg\mathsf{v}_1\}\}$. We have $\mathcal{J}_0 \not\models \bot \leftarrow \{\neg\mathsf{v}_0, \neg\mathsf{v}_1\}$, so the solver now switches to the new propositional model $\mathcal{J}_2 = (\mathcal{J}_0 \setminus \{\neg\mathsf{v}_0\}) \cup \{\mathsf{v}_0\}$. There is still no theory conflict, so we perform another quantifier instantiation step yielding $\mathcal{N}_3 = \mathcal{N}_1 \cup \{\mathsf{f}(1) = 0 \leftarrow \{\mathsf{v}_2\}, \mathsf{g}(1) = 0 \leftarrow \{\mathsf{v}_3\}, \bot \leftarrow \{\neg\mathsf{v}_2, \neg\mathsf{v}_3\}\}$. We have $\mathcal{J}_2 \not\models \bot \leftarrow \{\neg\mathsf{v}_2, \neg\mathsf{v}_3\}$, so the solver now switches to the new propositional model $\mathcal{J}_4 = (\mathcal{J}_2 \setminus \{\neg\mathsf{v}_2\}) \cup \{\mathsf{v}_2\}$. Now the solve derives a theory conflict $\bot \leftarrow \{\mathsf{v}_2\}$, and switches to $\mathcal{J}_6 = (\mathcal{J}_4 \setminus \{\neg\mathsf{v}_3\}) \cup \{\mathsf{v}_3\}$. In this propositional model there is also a conflict, $\bot \leftarrow \{\mathsf{v}_3\}$, and the prover terminates using the Unsat rule.

Our mathematization of AVATAR and SMT with quantifiers exposes their dissimilarities. With SMT, splitting is mandatory, and there is no subsumption or simplification, locking, or active and passive sets. And of course, theory inferences are n-ary and quantifier instantiation is unary, whereas superposition is binary. Nevertheless, their completeness follows from the same principles.

8 Conclusion

Our splitting framework captures splitting calculi and provers in a general way, independently of the base calculus. Users can conveniently derive a dynamic

refutational completeness result for a splitting prover based on a given statically refutationally complete calculus. As we developed the framework, we faced some tension between constraining the SAT solver's behavior and the saturation prover's. We preferred to constrain the prover, because the prover is typically easier to modify than an off-the-shelf SAT solver. To our surprise, we discovered counterexamples related to locking, formula selection, and simplification, which may affect Vampire's AVATAR implementation, depending on the SAT solver used. We proposed some restrictions, but alternatives could be investigated.

We found that labeled splitting can be seen as a variant of AVATAR where the SAT solver follows a strict strategy and propositional variables are not reused across branches. A benefit of the strict strategy is that its locking preserves completeness. As for the relation between AVATAR and SMT, there are some glaring differences, including that splitting is necessary to support disjunctions in SMT but fully optional in AVATAR. For future work, we could try to complete the picture by considering other related architectures [5–7, 17].

Acknowledgment. Petar Vukmirović greatly helped us design the abstract notions related to A-formulas. Giles Reger patiently explained AVATAR and revealed some of its secrets. Simon Cruanes did the same regarding lightweight AVATAR. Simon Robillard, Andrei Voronkov, Uwe Waldmann, Christoph Weidenbach discussed splitting with us. Haniel Barbosa, Pascal Fontaine, Andy Reynolds, and Cesare Tinelli explained some fine points of SMT. Natarajan Shankar pointed us to his work on the Shostak procedure. Ahmed Bhayat, Mark Summerfield, Dmitriy Traytel, Petar Vukmirović, and the anonymous reviewers suggested textual improvements. We thank them all.

This 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). The research has also received funding from the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) under the Vidi program (project No. 016.Vidi.189.037, Lean Forward).

References

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

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

[3] Barrett, C., Conway, C.L., Deters, M., Hadarean, L., Jovanovic, D., King, T., Reynolds, A., Tinelli, C.: CVC4. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 171–177. Springer (2011)

[4] Bjø[r]ner, N., Reger, G., Suda, M., Voronkov, A.: AVATAR modulo theories. In: Benzmüller, C., Sutcliffe, G., Rojas, R. (eds.) GCAI 2016. EPiC Series in Computing, vol. 41, pp. 39–52. EasyChair (2016)

[5] Bonacina, M.P., Graham-Lengrand, S., Shankar, N.: Satisfiability modulo theories and assignments. In: de Moura, L. (ed.) CADE-26. LNCS, vol. 10395, pp. 42–59. Springer (2017)

[6] Bonacina, M.P., Lynch, C., de Moura, L.: On deciding satisfiability by DPLL($\Gamma + T$) and unsound theorem proving. In: Schmidt, R.A. (ed.) CADE-22. LNCS, vol. 5663, pp. 35–50. Springer (2009)

[7] Bonacina, M.P., Plaisted, D.A.: SGGS theorem proving: An exposition. In: Schulz, S., de Moura, L., Konev, B. (eds.) PAAR-2014. EPiC Series in Computing, vol. 31, pp. 25–38. EasyChair (2014)
[8] Bouton, T., de Oliveira, D.C.B., Déharbe, D., Fontaine, P.: veriT: An open, trustable and efficient SMT-solver. In: Schmidt, R.A. (ed.) CADE-22. LNCS, vol. 5663, pp. 151–156. Springer (2009)
[9] Cruanes, S.: Extending Superposition with Integer Arithmetic, Structural Induction, and Beyond. Ph.D. thesis, École polytechnique (2015)
[10] Ebner, G., Ullrich, S., Roesch, J., Avigad, J., de Moura, L.: A metaprogramming framework for formal verification. Proc. ACM Program. Lang. 1(ICFP), 34:1–34:29 (2017)
[11] Fietzke, A., Weidenbach, C.: Labelled splitting. Ann. Math. Artif. Intell. 55(1–2), 3–34 (2009)
[12] Ge, Y., de Moura, L.: Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 306–320. Springer (2009)
[13] Gleiss, B., Kovács, L., Rath, J.: Subsumption demodulation in first-order theorem proving. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020, Part I. LNCS, vol. 12166, pp. 297–315. Springer (2020)
[14] Knuth, D.E., Bendix, P.B.: Simple word problems in universal algebras. In: Leech, J. (ed.) Computational Problems in Abstract Algebra. pp. 263–297. Pergamon Press (1970)
[15] McCune, W., Wos, L.: Otter—the CADE-13 competition incarnations. J. Autom. Reason. 18(2), 211–220 (1997)
[16] de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer (2008)
[17] de Moura, L., Jovanović, D.: A model-constructing satisfiability calculus. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) VMCAI 2013. LNCS, vol. 7737, pp. 1–12. Springer (2013)
[18] Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). J. ACM 53(6), 937–977 (2006)
[19] Nipkow, T., Klein, G.: Concrete Semantics: With Isabelle/HOL. Springer (2014)
[20] Reger, G., Suda, M., Voronkov, A.: Playing with AVATAR. In: Felty, A.P., Middeldorp, A. (eds.) CADE-25. LNCS, vol. 9195, pp. 399–415. Springer (2015)
[21] Reynolds, A., Barbosa, H., Fontaine, P.: Revisiting enumerative instantiation. In: Beyer, D., Huisman, M. (eds.) TACAS 2018. LNCS, vol. 10806, pp. 112–131. Springer (2018)
[22] Reynolds, A., Tinelli, C., Goel, A., Krstic, S.: Finite model finding in SMT. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 640–655. Springer (2013)
[23] Riazanov, A., Voronkov, A.: Splitting without backtracking. In: Nebel, B. (ed.) IJCAI 2001. pp. 611–617. Morgan Kaufmann (2001)
[24] Robinson, J.A.: A machine-oriented logic based on the resolution principle. J. ACM 12(1), 23–41 (1965)
[25] Schlichtkrull, A., Blanchette, J.C., Traytel, D.: A verified prover based on ordered resolution. In: Mahboubi, A., Myreen, M.O. (eds.) CPP 2019. pp. 152–165. ACM (2019)
[26] Steen, L.A., Seebach, J.A.: Counterexamples in Topology. Holt, Rinehart and Winston (1970)

[27] Sutcliffe, G.: The TPTP problem library and associated infrastructure—from CNF to TH0, TPTP v6.4.0. J. Autom. Reason. 59(4), 483–502 (2017)
[28] Voronkov, A.: AVATAR: The architecture for first-order theorem provers. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 696–710. Springer (2014)
[29] 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 (2020)
[30] Waldmann, U., Tourret, S., Robillard, S., Blanchette, J.: A comprehensive framework for saturation theorem proving (technical report). Technical report (2020), `https://matryoshka-project.github.io/pubs/saturate_report.pdf`
[31] Weidenbach, C.: Combining superposition, sorts and splitting. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. II, pp. 1965–2013. Elsevier and MIT Press (2001)