Unifying Splitting

Gabriel Ebner(ID) · **Jasmin Blanchette**(ID) ·
Sophie Tourret(ID)

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 that 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 [26] and superposition [1] is that they avoid case analyses. 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_i that share no variables. Splitting approaches include splitting with backtracking [31, 32], splitting without backtracking [25], labeled splitting [13], and AVATAR [28].

The AVATAR architecture, which is based on a satisfiability (SAT) solver, is of particular interest because it is so successful. Voronkov reported that an AVATAR-enabled Vampire could solve 421 TPTP problems that had never been solved before by any system [28, Sect. 9], a mind-boggling number. Intuitively, 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

---

Gabriel Ebner · Jasmin Blanchette
Vrije Universiteit Amsterdam, Amsterdam, the Netherlands
E-mail: gebner@gebner.org, j.c.blanchette@vu.nl

Jasmin Blanchette · Sophie Tourret
Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
E-mail: {jasmin.blanchette,sophie.tourret}@inria.fr

Jasmin Blanchette · Sophie Tourret
Max-Planck-Institut für Informatik, Saarland Informatics Campus, Saarbrücken, Germany
E-mail: {jasmin.blanchette,stourret}@mpi-inf.mpg.de

generalizes traditional saturation provers and yet degenerates to a SAT solver if the problem is propositional.

To illustrate the approach, we follow the key steps of an AVATAR-enabled resolution prover on the initial clause set containing $\neg p(a)$, $\neg q(z,z)$, and $p(x) \lor q(y,b)$. The disjunction can be split into $p(x) \leftarrow \{[p(x)]\}$ and $q(y,b) \leftarrow \{[q(y,b)]\}$, 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 $[p(x)] \lor [q(y,b)]$. Suppose $\mathcal{J}$ makes $[p(x)]$ true and $[q(y,b)]$ false. Then resolving $p(x) \leftarrow \{[p(x)]\}$ with $\neg p(a)$ produces $\bot \leftarrow \{[p(x)]\}$, meaning that $[p(x)]$ must be false. Next, the SAT solver makes $[p(x)]$ false and $[q(y,b)]$ true. Resolving $q(y,b) \leftarrow \{[q(y,b)]\}$ with $\neg q(z,z)$ yields $\bot \leftarrow \{[q(y,b)]\}$, meaning that $[q(y,b)]$ must be false. Since both disjuncts of $[p(x)] \lor [q(y,b)]$ are false, the SAT solver 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 perhaps 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 the work of 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 [21] 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 of the form $C \leftarrow A$, where C is a base clause or formula and A is a set of propositional literals, called assertions (Sect. 2).

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 major 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). This layer is also dynamically complete.

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 satisfiability modulo theories (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.

A shorter version of this article was presented at CADE-28 [11]. This article extends the conference paper with more explanations, examples, counterexamples, and proofs. We strengthened the definition of consequence relation to require compactness, which allowed us to simplify property (D4). The property (D4) from the conference paper is proved as Lemma 5. The definition of strongly finitary was also changed to include a stronger condition on the introduced assertions, which is needed for the proof of Lemma 72.

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 $\perp$ denoting falsehood. A *consequence relation* $\models$ over $\mathbf{F}$ is a relation $\models \subseteq (\mathscr{P}(\mathbf{F}))^2$ with the following properties for all sets $M, M', N, N' \subseteq \mathbf{F}$ and formulas $C, D \in \mathbf{F}$:

(D1) $\{\perp\} \models \emptyset$;
(D2) $\{C\} \models \{C\}$;
(D3) if $M' \subseteq M$ and $N' \subseteq N$, then $M' \models N'$ implies $M \models N$;
(D4) if $M \models N \cup \{C\}$ and $M' \cup \{C\} \models N'$, then $M \cup M' \models N \cup N'$;
(D5) if $M \models N$, then there exist finite sets $M' \subseteq M$ and $N' \subseteq N$ such that $M' \models N'$.

The intended interpretation of $M \models N$ is conjunctive on the left but disjunctive on the right: "$\bigwedge M \to \bigvee N$." The disjunctive interpretation of N will be useful to define splittability abstractly in Sect. 3.1. Property (D4) is called the cut rule, and (D5) is called compactness.

For their saturation framework, Waldmann et al. instead consider a fully conjunctive version of the consequence relation, with different properties. The incompatibility can easily be repaired: Given a consequence relation $\models$, we can obtain a consequence relation $\models'$ in their sense by defining $M \models' N$ if and only if $M \models \{C\}$ for every $C \in N$. The two versions differ only when the right-hand side is not a singleton. The conjunctive version $\models'$ can then be used when interacting with the saturation framework.

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 set $M \models N$ if and only if

$$\{C \in \mathbf{F} \mid \quad C \in M\} \cup \{C \in \mathbf{F} \mid \sim C \in N\}$$
$$\models \{C \in \mathbf{F} \mid \sim C \in M\} \cup \{C \in \mathbf{F} \mid \quad C \in N\}$$

We write $M \mathrel{\vDash\!\!\!\dashv} N$ for the conjunction of $M \models N$ and $N \models M$.

Lemma 1 *Let $C \in \mathbf{F}_{\sim}$. Then $\{C\} \cup \{\sim C\} \models \{\perp\}$.*

Proof This holds by (D2) and (D3) due to the definition of $\models$ on $\mathbf{F}_{\sim}$. $\quad\square$

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

Example 1 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 $L_1,\dots,L_n$ written $C = L_1 \vee \cdots \vee L_n$. The clause's variables are implicitly quantified universally. 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 a posteriori need not be performed 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 $(C_n,\dots,C_1,D) \in \mathbf{F}^{n+1}$. The formulas $C_n,\dots,C_1$ are the *premises*, and D is the *conclusion*. Define $prems(\iota) = \{C_n,\dots,C_1\}$ and $concl(\iota) = \{D\}$. An inference ι is *sound* w.r.t. $\approx\!\!\!\mid$ if $prems(\iota) \approx\!\!\!\mid concl(\iota)$. An *inference system* Inf is a set of $\mathbf{F}$-inferences.

Given $N \subseteq \mathbf{F}$, we let $Inf(N)$ denote the set of all inferences in Inf whose premises are included in N, and $Inf(N,M) = Inf(N \cup M) \setminus Inf(N \setminus M)$ 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 $Red = (Red_I, Red_F)$, where $Red_I : \mathscr{P}(\mathbf{F}) \to \mathscr{P}(Inf)$ and $Red_F : \mathscr{P}(\mathbf{F}) \to \mathscr{P}(\mathbf{F})$ enjoy the following properties for all sets $M, N \subseteq \mathbf{F}$:

(R1) if $N \models \{\bot\}$, then $N \setminus Red_F(N) \models \{\bot\}$;
(R2) if $M \subseteq N$, then $Red_F(M) \subseteq Red_F(N)$ and $Red_I(M) \subseteq Red_I(N)$;
(R3) if $M \subseteq Red_F(N)$, then $Red_F(N) \subseteq Red_F(N \setminus M)$ and $Red_I(N) \subseteq Red_I(N \setminus M)$;
(R4) if $\iota \in Inf$ and $concl(\iota) \in N$, then $\iota \in Red_I(N)$.

Inferences in $Red_I(N)$ and formulas in $Red_F(N)$ are said to be *redundant* w.r.t. N. Red_I indicates which inferences need not 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 to 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 (Inf, Red) forms a *calculus*. A set $N \subseteq \mathbf{F}$ is *saturated* w.r.t. Inf and Red_I if $Inf(N) \subseteq Red_I(N)$. The calculus (Inf, 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 2 *Assume that the calculus (Inf, Red) is statically complete. Then $\bot \notin Red_F(N)$ for every $N \subseteq \mathbf{F}$.*

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

Remark 3 Given a redundancy criterion (Red_I, Red_F), where $\bot \notin Red_F(\mathbf{F})$, we can make it stricter as follows. Define Red'_I such that $\iota \in Red'_I(N)$ if and only if either $\iota \in Red_I(N)$ or $\bot \in N$. Define Red'_F such that $C \in Red'_F(N)$ if and only if either $C \in Red_F(N)$ or else both $\bot \in N$ and $C \neq \bot$. Obviously, $Red' = (Red'_I, Red'_F)$ 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 (Inf, Red) is statically complete, then (Inf, Red') is also statically complete. (In the last case, the condition $\bot \notin Red_F(\mathbf{F})$ holds by Lemma 2.)

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 $Inf(N_\infty) \subseteq \bigcup_i Red_I(N_i)$ and *strongly fair* if $\limsup_{i \to \infty} Inf(N_i) \subseteq \bigcup_i Red_I(N_i)$. 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 $\rhd \subseteq X^2$ (pronounced "triangle"), a $\rhd$-*derivation* is a 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. Abusing language, and departing slightly 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 $\rhd_{Red_F} \subseteq (\mathscr{P}(\mathbf{F}))^2$ be the relation such that $M \rhd_{Red_F} N$ if and only if $M \setminus N \subseteq Red_F(N)$. Note that it is reflexive and hence supports stuttering. The relation is also transitive due to (R3). We could additionally require soundness ($M \approx\!\!\!| N$) or at least consistency preservation ($M \not\approx\!\!\!| \{\bot\}$ implies $N \not\approx\!\!\!| \{\bot\}$), but this is unnecessary for proving completeness.

The calculus (Inf, Red) is *dynamically (refutationally) complete* (w.r.t. $\models$) if for every $\rhd_{Red_F}$-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 $fml(\mathsf{v}) \in \mathbf{F}$ is associated with each propositional variable $\mathsf{v} \in \mathbf{V}$. Intuitively, v approximates $fml(\mathsf{v})$ at the propositional level. This definition is extended so that $fml(\neg\mathsf{v}) = {\sim} fml(\mathsf{v})$. 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 variable $\mathsf{v} \in \mathbf{V}$, exactly one of $\mathsf{v} \in \mathcal{J}$ and $\neg\mathsf{v} \in \mathcal{J}$ holds. We lift fml to sets in an elementwise fashion: $fml(\mathcal{J}) = \{fml(a) \mid a \in \mathcal{J}\}$. In the rest of this article, we will often implicitly lift functions elementwise to sets. The condition on the variables ensures that $\mathcal{J}$ is propositionally consistent. $\mathcal{J}$ might nevertheless be inconsistent for $\models$, which takes into account the semantics of the formulas $fml(\mathsf{v})$ associated with the variables v; for example, we might have $\mathcal{J} = \{\mathsf{v}_0, \mathsf{v}_1\}$, $fml(\mathsf{v}_0) = \mathsf{p}(x)$, and $fml(\mathsf{v}_1) = \neg\mathsf{p}(\mathsf{a})$, and $\mathcal{J} \models \{\bot\}$.

An *A-formula* over a set $\mathbf{F}$ of *base formulas* and an assertion set $\mathbf{A}$ is a pair $\mathscr{C} = (C, A) \in \mathbf{AF} = \mathbf{F} \times \mathscr{P}_{\text{fin}}(\mathbf{A})$, 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 $a_1 \wedge \cdots \wedge a_n \rightarrow C$. 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, $\mathscr{N}_\perp$ is the set consisting of all A-formulas of the form $\perp \leftarrow A \in \mathscr{N}$, where $A \in \mathscr{P}_{\text{fin}}(\mathbf{A})$. Since $\perp \leftarrow \{a_1, \ldots, a_n\}$ can be read as $\neg a_1 \vee \cdots \vee \neg a_n$, we call such A-formulas *propositional clauses*. (In contrast, we call a variable-free base formula such as $\mathsf{p} \vee \mathsf{q}$ a ground clause when $\mathbf{F}$ is first-order logic.) The set $\mathscr{N}_\perp$ represents the clauses considered by the SAT solver in the original AVATAR [28]. Note the use of calligraphic letters (e.g., $\mathscr{C}, \mathscr{N}$) to range over A-formulas and sets of A-formulas.

Model-guided provers only consider A-formulas 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 $\mathscr{N} \subseteq \mathbf{AF}$, the *enabled projection* $\mathscr{N}_\mathcal{J} \subseteq \lfloor \mathscr{N} \rfloor$ consists of the projections $\lfloor \mathscr{C} \rfloor$ of all A-formulas $\mathscr{C}$ enabled in $\mathcal{J}$. Analogously, the *enabled projection* $Inf_\mathcal{J} \subseteq \lfloor Inf \rfloor$ of a set Inf of $\mathbf{AF}$-inferences consists of the projections $\lfloor \iota \rfloor$ of all inferences $\iota \in Inf$ whose premises are all enabled in $\mathcal{J}$.

A propositional interpretation $\mathcal{J}$ is a *propositional model* of $\mathscr{N}_\perp$, written $\mathcal{J} \models \mathscr{N}_\perp$, if $\perp \notin (\mathscr{N}_\perp)_\mathcal{J}$. (i.e., $(\mathscr{N}_\perp)_\mathcal{J} = \emptyset$). Moreover, we write $\mathcal{J} \approx \mathscr{N}_\perp$ if $\perp \notin (\mathscr{N}_\perp)_\mathcal{J}$ or $fml(\mathcal{J}) \approx \{\perp\}$. A set $\mathscr{N}_\perp$ is *propositionally satisfiable* if there exists an interpretation $\mathcal{J}$ such that $\mathcal{J} \models \mathscr{N}_\perp$. In contrast to consequence relations, propositional modelhood $\models$ interprets the set $\mathscr{N}_\perp$ conjunctively: $\mathcal{J} \models \mathscr{N}_\perp$ is informally understood as $\mathcal{J} \models \bigwedge \mathscr{N}_\perp$.

Given consequence relations $\models$ and $\approx$, we lift them from $\mathscr{P}(\mathbf{F})$ to $\mathscr{P}(\mathbf{AF})$: $\mathscr{M} \models \mathscr{N}$ if and only if $\mathscr{M}_\mathcal{J} \models \lfloor \mathscr{N} \rfloor$ for every $\mathcal{J}$ in which $\mathscr{N}$ is enabled, and $\mathscr{M} \approx \mathscr{N}$ if and only if $fml(\mathcal{J}) \cup \mathscr{M}_\mathcal{J} \approx \lfloor \mathscr{N} \rfloor$ for every $\mathcal{J}$ in which $\mathscr{N}$ 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, $\approx$ captures a stronger semantics: For example, thanks to $fml(\mathcal{J})$ among the premises for $\approx$, the A-formula $fml(a) \leftarrow \{a\}$ is always a $\approx$-tautology. Also note that assuming $\emptyset \not\models \emptyset$, then $\models \subseteq \approx$ on sets that contain exclusively propositional clauses. When needed, we use $\approx_\mathbf{F}$ to denote $\approx$ on $\mathscr{P}(\mathbf{F})$ and analogously for $\approx_\mathbf{AF}$, as well as $\models_\mathbf{F}$ and $\models_\mathbf{AF}$.

Lemma 4 *The relations $\models$ and $\approx$ on $\mathscr{P}(\mathbf{AF})$ are consequence relations.*

Proof We consider only $\approx$; the proof for $\models$ is analogous. For (D1), we need to show $fml(\mathcal{J}) \cup \{\perp\} \approx \emptyset$ for every $\mathcal{J}$ because $\emptyset$ is always enabled. This follows from (D1) and (D3). For (D2), we need to show $fml(\mathcal{J}) \cup \{C \leftarrow A\}_\mathcal{J} \approx \lfloor \{C \leftarrow A\} \rfloor$, assuming $C \leftarrow A$ is enabled in $\mathcal{J}$. Hence it suffices to show $fml(\mathcal{J}) \cup \{C\} \approx \{C\}$, which follows from (D2) and (D3). For (D3), it suffices to show $fml(\mathcal{J}) \cup \mathscr{M}_\mathcal{J} \approx \lfloor \mathscr{N} \rfloor$ assuming that $\mathscr{N}$ is enabled in $\mathcal{J}$, and $fml(\mathcal{J}) \cup \mathscr{M}'_\mathcal{J} \approx \lfloor \mathscr{N}' \rfloor$ for every $\mathscr{M}' \subseteq \mathscr{M}$ and $\mathscr{N}' \subseteq \mathscr{N}$. This follows from (D3) and monotonicity of $\lfloor \rfloor$ and $(\ )_\mathcal{J}$. For (D4), we need to show $fml(\mathcal{J}) \cup (\mathscr{M} \cup \mathscr{M}')_\mathcal{J} \approx \lfloor \mathscr{N} \cup \mathscr{N}' \rfloor$, assuming $fml(\mathcal{J}) \cup \mathscr{M}_\mathcal{J} \approx \lfloor \mathscr{N} \rfloor \cup \{C\}$ if $C \leftarrow A$ is enabled in $\mathcal{J}$, $fml(\mathcal{J}) \cup \mathscr{M}'_\mathcal{J} \cup \{C \leftarrow A\}_\mathcal{J} \approx \lfloor \mathscr{N}' \rfloor$, and $\mathscr{N} \cup \mathscr{N}'$ is enabled in $\mathcal{J}$. This follows directly from (D4) if $C \leftarrow A$ is enabled in $\mathcal{J}$, and from (D3) if $C \leftarrow A$ is not enabled.

Finally, we show the compactness of $\approx_\mathbf{AF}$ (D5), using the compactness of propositional logic. First we consider the case where $\mathscr{N}$ is never enabled. Then the set of assertions in $\mathscr{N}$, seen as conjunctions of propositional literals, is unsatisfiable. By compactness, there exists a finite subset of these assertions that is also unsatisfiable, i.e., there is a finite subset $\mathscr{N}'$ of $\mathscr{N}$ that is also never enabled. Thus for any finite subset $\mathscr{M}'$ of $\mathscr{M}$, $\mathscr{M}' \approx \mathscr{N}'$ as wanted.

Otherwise, there is at least one $\mathcal{J}$ enabling $\mathcal{N}$. By abuse of notation, we write $\mathcal{N}_A$ even if $A \subseteq \mathbf{A}$ is not an interpretation. For every interpretation $\mathcal{J}$ in which $\mathcal{N}$ is enabled, there exist by compactness of $\mathrel{\approx}_\mathbf{F}$ finite sets $\mathcal{J}' \subseteq \mathcal{J}$, $\mathscr{M}^{\mathcal{J}} \subseteq \mathscr{M}$, and $\mathscr{N}^{\mathcal{J}} \subseteq \mathscr{N}$ such that $fml(\mathcal{J}') \cup \mathscr{M}^{\mathcal{J}}{}_{\mathcal{J}'} \mathrel{\approx} \lfloor \mathscr{N}^{\mathcal{J}} \rfloor$. Define $E = \{\bot \leftarrow \{\neg a\} \mid a \in A \text{ for some } C \leftarrow A \in \mathscr{N}\}$. Note that $\mathcal{J} \models E$ if and only if $\mathscr{N}$ is enabled in $\mathcal{J}$. This observation implies that the sets of propositional clauses E and $\{\bot \leftarrow \mathcal{J}' \mid \mathcal{J} \text{ interpretation where } \mathscr{N} \text{ is enabled}\} \cup E$ are respectively propositionally satisfiable and propositionally unsatisfiable. By compactness, there exists a finite unsatisfiable subset $\{\bot \leftarrow \mathcal{J}'_1, \ldots, \bot \leftarrow \mathcal{J}'_n\} \cup E'$ of the latter set.

Let $\mathscr{M}' = \bigcup_i \mathscr{M}^{\mathcal{J}_i}$ and $\mathscr{N}' = \bigcup_i \mathscr{N}^{\mathcal{J}_i} \cup \mathscr{N}''$ where $\mathcal{J}_i$ is any of the interpretations enabling $\mathscr{N}$ that is at the origin of the existence of this $\mathcal{J}'_i$ and $\mathscr{N}''$ is a finite subset of $\mathscr{N}$ such that all assertions in E' also occur negated in $\mathscr{N}''$. Note that both $\mathscr{M}'$ and $\mathscr{N}'$ are finite sets. It now suffices to show $\mathscr{M}' \mathrel{\approx} \mathscr{N}'$. Thus let $\mathcal{J}$ be an interpretation in which $\mathscr{N}'$ is enabled. Then $\mathcal{J} \models E'$ because all assertions in E' also appear negated in $\mathscr{N}'' \subseteq \mathscr{N}'$. Thus, since $\{\bot \leftarrow \mathcal{J}'_1, \ldots, \bot \leftarrow \mathcal{J}'_n\} \cup E'$ is unsatisfiable, there must exist an index k such that $\mathcal{J} \not\models \bot \leftarrow \mathcal{J}'_k$, that is, $\mathcal{J}'_k \subseteq \mathcal{J}$. We have $fml(\mathcal{J}'_k) \cup \mathscr{M}^{\mathcal{J}_k}{}_{\mathcal{J}'_k} \mathrel{\approx} \lfloor \mathscr{N}^{\mathcal{J}_k} \rfloor$ by construction, and thus $fml(\mathcal{J}) \cup \bigcup_i \mathscr{M}^{\mathcal{J}_i}{}_{\mathcal{J}} \mathrel{\approx} \bigcup_i \lfloor \mathscr{N}^{\mathcal{J}_i} \rfloor \cup \lfloor \mathscr{N}'' \rfloor$ by (D3). $\qquad\square$

Given sets $M, N \subseteq \mathscr{P}(\mathbf{F})$, the expression $M \models N$ can refer to either the base consequence relation on $\mathscr{P}(\mathbf{F})$ or the lifted consequence relation on $\mathscr{P}(\mathbf{AF})$ (since $\mathbf{F} \subseteq \mathbf{AF}$). Fortunately, there is no ambiguity. First, let us show a preparatory lemma:

Lemma 5 *Let $\models$ be a consequence relation on $\mathbf{F}$, and $M, N \subseteq \mathbf{F}$. If $M' \models N'$ for all $M' \supseteq M$ and $N' \supseteq N$ such that $M' \cup N' = \mathbf{F}$, then $M \models N$.*

Proof By contraposition, we assume that $M \not\models N$, and we need to find $M' \supseteq M$ and $N' \supseteq N$ such that $M' \cup N' = \mathbf{F}$ and $M' \not\models N'$. We apply Zorn's lemma to obtain a maximal element (M', N') of the set $\{(M', N') \mid M \subseteq M', N \subseteq N' \text{ and } M' \not\models N'\}$ with the order $(M_1, N_1) \leq (M_2, N_2)$ if and only if $M_1 \subseteq M_2$ and $N_1 \subseteq N_2$. Compactness of $\models$, together with (D3), guarantees that every chain in this set has an upper bound; for nonempty chains, this is the pairwise union of all the elements in the chain. It remains to show that $M' \cup N' = \mathbf{F}$. Assume to the contrary that $C \notin M' \cup N'$ for some C. Due to the maximality of (M', N'), we necessarily have $M' \cup \{C\} \models N'$ and $M' \models N' \cup \{C\}$. Applying the cut rule for $\models$, we get $M' \models N'$, a contradiction. $\qquad\square$

Lemma 6 *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 we must show that $M \mathrel{\approx}_\mathbf{F} N$ if and only if $M \mathrel{\approx}_\mathbf{AF} N$. First assume that $M \mathrel{\approx}_\mathbf{F} N$. Then clearly $fml(\mathcal{J}) \cup M \mathrel{\approx}_\mathbf{F} N$ for any $\mathcal{J}$ by (D3) and thus $M \mathrel{\approx}_\mathbf{AF} N$. Assuming $M \mathrel{\approx}_\mathbf{AF} N$, we show $M \mathrel{\approx}_\mathbf{F} N$ using Lemma 5. It thus suffices to show that $M' \mathrel{\approx}_\mathbf{F} N'$ for every $M' \supseteq M$ and $N' \supseteq N$ such that $M' \cup N' = \mathbf{F}$. Set $\mathcal{J} = \{\mathsf{v} \mid fml(\mathsf{v}) \in M'\} \cup \{\neg\mathsf{v} \mid fml(\mathsf{v}) \notin M'\}$. Then $fml(\mathcal{J}) \cup M \cup {\sim}N \subseteq M' \cup {\sim}N'$. By the assumption $M \mathrel{\approx}_\mathbf{AF} N$ we have $fml(\mathcal{J}) \cup M \mathrel{\approx}_\mathbf{F} N$ and thus $M' \mathrel{\approx}_\mathbf{F} N'$ via (D3). $\qquad\square$

Aside from resolving ambiguity, Lemma 6 justifies the use of splitting in provers without compromising soundness or completeness: When we prove a completeness theorem that claims that a given prover derives $\bot$ from any initial $\models_\mathbf{AF}$-unsatisfiable set $M \subseteq \mathbf{AF}$, Lemma 6 allows us to conclude that it also derives $\bot$ when starting from any initial $\models_\mathbf{F}$-unsatisfiable set $M \subseteq \mathbf{F}$.

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

Remark 7 Our propositional interpretations are always total. We could also consider partial interpretations—that is, $\mathcal{J} \subseteq \mathbf{A}$ such that at most one 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 be simulated by total ones: For every variable v in the partial interpretation, we can use two variables v^+ and v^- in the total interpretation and interpret v^+ as true if v is true and 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.

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 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 $fml(\mathsf{v}) = C$ 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 $\bumpeq$-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 exhaustive (according to $\bumpeq$).

3 Splitting Calculi

Let $\mathbf{F}$ be a set of base formulas equipped with $\bot$, $\models$, and $\bumpeq$. The consequence relation $\bumpeq$ is assumed to be nontrivial: (D6) $\emptyset \not\bumpeq \emptyset$. Let $\mathbf{A}$ be a set of assertions over V, and let $\mathbf{AF}$ be the set of A-formulas over $\mathbf{F}$ and $\mathbf{A}$. Let $(FInf, FRed)$ be a base calculus for $\mathbf{F}$-formulas, where $FRed$ is a redundancy criterion that additionally satisfies

(R5) $Inf(\mathbf{F}, Red_{\mathrm{F}}(N)) \subseteq Red_{\mathrm{I}}(N)$ for every $N \subseteq \mathbf{F}$;
(R6) $\bot \notin FRed_{\mathrm{F}}(N)$ for every $N \subseteq \mathbf{F}$;
(R7) $C \in FRed_{\mathrm{F}}(\{\bot\})$ 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 2), and (R7) can be made without loss of generality (Remark 3). Bachmair and Ganzinger's redundancy criterion for superposition [1, Sect. 4.3] meets (R1)–(R7).

From a base calculus, we will define an induced *splitting calculus* $(SInf, SRed)$. We will show that the splitting calculus is sound w.r.t. $\bumpeq$ and that it is statically and dynamically complete w.r.t. $\models$. Furthermore, we will show two stronger results that take into account the switching of propositional models that characterizes most splitting architectures: strong static completeness and strong dynamic completeness.

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:

$$\frac{(C_i \leftarrow A_i)_{i=1}^n}{D \leftarrow A_1 \cup \cdots \cup A_n} \text{ BASE} \qquad\qquad \frac{(\bot \leftarrow A_i)_{i=1}^n}{\bot} \text{ UNSAT}$$

For BASE, the side condition is $(C_n, \ldots, C_1, D) \in FInf$. For UNSAT, the side condition is that $\{\bot \leftarrow A_1, \ldots, \bot \leftarrow A_n\}$ is propositionally unsatisfiable.

In addition, the following optional inference rules can be used if desired; the completeness proof does not depend on their application. Rules identified by double bars, such as SPLIT, are simplifications; they replace their premises with their conclusions in the current A-formula set. The premises' removal is justified by $SRed_F$, defined in Sect. 3.2.

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

In the SPLIT rule, we require that $C \neq \bot$ is splittable into $C_1, \ldots, C_n$ and that $a_i \in asn(C_i)$ for each i. A formula C is *splittable* into formulas $C_1, \ldots, C_n$ if $n \geq 2$, $\{C\} \approx \{C_1, \ldots, C_n\}$ and $C \in FRed_F(\{C_i\})$ for each i.

SPLIT performs an n-way case analysis on C. Each case C_i is approximated by an assertion a_i. The first conclusion expresses that the cases are 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. For example, the clause $p(x) \vee q(x)$ is not splittable because of the shared variable x, whereas $p(x) \vee q(y)$ can be split into $\{p(x), q(y)\}$.

$$\frac{(\bot \leftarrow A_i)_{i=1}^n \quad C \leftarrow A}{(\bot \leftarrow A_i)_{i=1}^n} \text{ COLLECT} \qquad \frac{(\bot \leftarrow A_i)_{i=1}^n \quad C \leftarrow A \cup B}{(\bot \leftarrow A_i)_{i=1}^n \quad C \leftarrow B} \text{ TRIM}$$

For COLLECT, we require $C \neq \bot$ and $\{\bot \leftarrow A_i\}_{i=1}^n \approx \{\bot \leftarrow A\}$. For TRIM, we require $C \neq \bot$ and $\{\bot \leftarrow A_i\}_{i=1}^n \cup \{\bot \leftarrow A\} \approx \{\bot \leftarrow B\}$.

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.

$$\frac{(\bot \leftarrow A_i)_{i=1}^n}{\bot} \text{ STRONGUNSAT} \qquad \frac{C \leftarrow A}{\bot \leftarrow \{\neg a\} \cup A} \text{ APPROX} \qquad \frac{}{C \leftarrow A} \text{ TAUTO}$$

For STRONGUNSAT, we require $\{\bot \leftarrow A_i\}_{i=1}^n \approx \{\bot\}$. For APPROX, we require $a \in asn(C)$. For TAUTO, we require $\approx \{C \leftarrow A\}$.

STRONGUNSAT is a variant of UNSAT that uses $\approx$ instead of $\models$. A splitting prover may choose to apply STRONGUNSAT if desired, but only UNSAT is necessary for completeness. In practice, $\approx$-entailment can be much more expensive to decide, or even be undecidable. A splitting prover could invoke an SMT solver [4] ($\approx$) with a time limit, falling back on a SAT solver ($\models$) if necessary.

APPROX can be used to make any derived A-formula visible to $\approx$. It is similar to a one-way split. TAUTO, which asserts a $\approx$-tautology, allows communication in the other direction, from the SMT or 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 Sect. 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. We are not aware of any prover that implements this idea, although a similar idea is described for ground $C(x)$ in the context of labeled splitting [13, Sect. 2].

Example 12 Consider a splitting calculus whose propositional solver is an SMT solver supporting linear arithmetic. Suppose that we are given the inconsistent clause set $\{\mathsf{c} > 0, \mathsf{c} < 0\}$. Two applications of APPROX make these clauses visible to the SMT solver, as the propositional clause set $\{\bot \leftarrow \neg(\mathsf{c} > 0), \bot \leftarrow \neg(\mathsf{c} < 0)\}$. Then the SMT solver, modeled by STRONGUNSAT, detects the unsatisfiability.

The splitting inference system commutes nicely with the enabled projection:

Lemma 13 $(\mathit{SInf}(\mathscr{N}))_{\mathcal{J}} = \mathit{FInf}(\mathscr{N}_{\mathcal{J}})$ *if* $\bot \notin \mathscr{N}_{\mathcal{J}}$.

Proof The condition $\bot \notin \mathscr{N}_{\mathcal{J}}$ rules out the UNSAT inferences. It remains to show that the enabled projection of a BASE inference is an FInf-inference from enabled premises, and vice versa. □

Theorem 14 (Soundness) *The rules* UNSAT, SPLIT, COLLECT, TRIM, STRONGUNSAT, APPROX, *and* TAUTO *are sound w.r.t.* $\approx\!\!\!|$. *Moreover, if every rule in FInf is sound w.r.t.* $\approx\!\!\!|$ *(on $\mathscr{P}(\mathbf{F})$), then the rule* BASE *is sound w.r.t.* $\approx\!\!\!|$ *(on $\mathscr{P}(\mathbf{AF})$).*

Proof CASES UNSAT, STRONGUNSAT, TAUTO: Trivial.

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

For the right conclusions, we must show $\mathit{fml}(\mathcal{J}) \cup \lfloor \{C \leftarrow A\}_{\mathcal{J}} \rfloor \approx\!\!\!| \{C_i\}$ 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 $\{\bot \leftarrow A_i\}_{i=1}^{n} \approx\!\!\!| \{C \leftarrow A\}$. This follows from the stronger side condition $\{\bot \leftarrow A_i\}_{i=1}^{n} \approx\!\!\!| \{\bot \leftarrow A\}$.

CASE TRIM: Only the right conclusion is nontrivial. Let $\mathscr{N} = \{\bot \leftarrow A_i\}_{i=1}^{n}$. It suffices to show $\mathscr{N}_{\mathcal{J}} \cup \{C \leftarrow A\}_{\mathcal{J}} \approx\!\!\!| \{C\}$ for every $\mathcal{J} \supseteq B$. Assume $\mathcal{J} \approx\!\!\!| \mathscr{N}_{\mathcal{J}} \cup \{C \leftarrow A\}_{\mathcal{J}}$. By the side condition $\mathscr{N} \cup \{\bot \leftarrow A\} \approx\!\!\!| \{\bot \leftarrow B\}$, we get $\mathscr{N}_{\mathcal{J}} \cup \{\bot \leftarrow A\}_{\mathcal{J}} \approx\!\!\!| \{\bot\}$, meaning that either $\mathscr{N}_{\mathcal{J}} \approx\!\!\!| \{\bot\}$ or $\mathcal{J} \supseteq A$. The first case is trivial. In the other case, $\mathcal{J} \approx\!\!\!| \mathscr{N}_{\mathcal{J}} \cup \{C\}$ and thus $\mathcal{J} \approx\!\!\!| \{C\}$, as required.

CASE APPROX: The proof is as for the left conclusion of SPLIT.

CASE BASE: To show $\{C_i \leftarrow A_i\}_{i=1}^{n} \approx\!\!\!| \{D \leftarrow A_1 \cup \cdots \cup A_n\}$, by the definition of $\approx\!\!\!|$ on $\mathscr{P}(\mathbf{AF})$, it suffices to show $\{C_1, \ldots, C_n\} \approx\!\!\!| \{D\}$. This follows from the soundness of the inferences in FInf. □

3.2 The Redundancy Criterion

Next, we lift the base redundancy criterion.

Definition 15 The *splitting redundancy criterion SRed* $= (SRed_I, SRed_F)$ is specified as follows. An A-formula $C \leftarrow A \in \mathbf{AF}$ is redundant w.r.t. $\mathcal{N}$, written $C \leftarrow A \in SRed_F(\mathcal{N})$, if either of these conditions is met:

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

An inference $\iota \in SInf$ is redundant w.r.t. $\mathcal{N}$, written $\iota \in SRed_I(\mathcal{N})$, if either of these conditions is met:

(3) ι is a BASE inference and $\{\iota\}_{\mathfrak{J}} \subseteq FRed_I(\mathcal{N}_{\mathfrak{J}})$ for every $\mathfrak{J}$; or
(4) ι is an UNSAT inference and $\bot \in \mathcal{N}$.

Condition (1) lifts $FRed_F$ 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 $SRed_F$ to justify global A-formula deletion, but also $FRed_F$ for local A-formula deletion in the locking prover. Note that $SRed$ is not reduced. Inference redundancy partly commutes with the enabled projection:

Lemma 16 $(SRed_I(\mathcal{N}))_{\mathfrak{J}} \subseteq FRed_I(\mathcal{N}_{\mathfrak{J}})$ if $\bot \notin \mathcal{N}$.

Proof Since $\bot \notin \mathcal{N}$, condition (4) of the definition of $SRed_I$ cannot apply. The inclusion then follows directly from condition (3) applied to the interpretation $\mathfrak{J}$.   □

Lemma 17 $\bot \notin SRed_F(\mathcal{N})$ for every $\mathcal{N} \subseteq \mathbf{AF}$.

Proof By Lemma 2, condition (1) of the definition of $SRed_F$ cannot apply. Nor can condition (2).   □

Lemma 18 *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 $FRed^{\cap \mathscr{G}, \sqsupset}$ 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].

To strengthen the redundancy criterion, we define a *tiebreaker order* $\sqsupset$ such that $C \leftarrow A \sqsupset D \leftarrow B$ if and only if $C = D$ and $A \supset B$. In this way, $C \leftarrow B$ is redundant w.r.t. $C \leftarrow A$ if $A \supset B$, even though the base clause is the same. The only requirement on $\sqsupset$ is that it 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* $\mathscr{G}_{\mathfrak{J}}$ indexed by a propositional model $\mathfrak{J}$. Here, "grounding" will mean enabled projection. For A-formulas $\mathscr{C}$, we set $\mathscr{G}_{\mathfrak{J}}(\mathscr{C}) = \{\mathscr{C}\}_{\mathfrak{J}}$. For inferences ι, we set $\mathscr{G}_{\mathfrak{J}}(\iota) = \{\iota\}_{\mathfrak{J}}$.

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

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 their technical report) but not to establish that $FRed^{\cap\mathscr{G},\sqsupset}$ is a redundancy criterion, so we can proceed. It is a routine exercise to check that $ARed$ coincides with $FRed^{\cap\mathscr{G},\sqsupset} = (FRed_{\mathrm{I}}^{\cap\mathscr{G}}, FRed_{\mathrm{F}}^{\cap\mathscr{G},\sqsupset})$, which is defined as follows:

1. $\iota \in FRed_{\mathrm{I}}^{\cap\mathscr{G}}(\mathscr{N})$ if and only if for every propositional interpretation $\mathfrak{J}$, we have $\mathscr{G}_{\mathfrak{J}}(\iota) \subseteq FRed_{\mathrm{I}}(\mathscr{G}_{\mathfrak{J}}(\mathscr{N}))$;

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

We also need to check that the consequence relation $\models$ used in $SRed$ coincides with the consequence relation $\models_{\mathscr{G}}^{\cap}$, which is defined as $\mathscr{M} \models_{\mathscr{G}}^{\cap} \{\mathscr{C}\}$ if and only if for every $\mathfrak{J}$ and $D \in \mathscr{G}_{\mathfrak{J}}(\{\mathscr{C}\})$, we have $\mathscr{G}_{\mathfrak{J}}(\mathscr{M}) \models \{D\}$. After expanding $\mathscr{G}_{\mathfrak{J}}$, this is exactly the definition we used for lifting $\models$ 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 $\mathscr{M} \subseteq \mathscr{N}$ and $\iota \in SRed_{\mathrm{I}}(M)$, then $\iota \in SRed_{\mathrm{I}}(N)$. This holds because if $\bot \in \mathscr{M}$, then $\bot \in \mathscr{N}$.

(R3) Given an UNSAT inference ι, we must show that if $\mathscr{M} \subseteq SRed_{\mathrm{F}}(N)$ and $\iota \in SRed_{\mathrm{I}}(\mathscr{N})$, then $\iota \in SRed_{\mathrm{I}}(\mathscr{N} \setminus \mathscr{M})$. This amounts to proving that if $\bot \in \mathscr{N}$, then $\bot \in \mathscr{N} \setminus \mathscr{M}$, which follows from Lemma 17.

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

$SRed$ is highly versatile. It can justify the deletion of A-formulas that are propositionally tautological, such as $C \leftarrow \{v, \neg v\}$. It lifts the base redundancy criterion gracefully: If $D \in FRed_{\mathrm{F}}(\{C_i\}_{i=1}^n)$, then $D \leftarrow A_1 \cup \cdots \cup A_n \in SRed_{\mathrm{F}}(\{C_i \leftarrow A_i\}_{i=1}^n)$. 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 $FRed_{\mathrm{F}}$ supports subsumption (e.g., following the lines of Waldmann et al. [29]), this also extends to A-formulas: $D \leftarrow B \in SRed_{\mathrm{F}}(\{C \leftarrow A\})$ 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 19 (Simplification) *For every* SPLIT, COLLECT, *or* TRIM *inference, the conclusions collectively make the premises redundant according to $SRed_{\mathrm{F}}$.*

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

CASE COLLECT: We must show $C \leftarrow A \in SRed_{\mathrm{F}}(\{\bot \leftarrow A_i\}_{i=1}^n)$. By condition (1) of the definition of $SRed_{\mathrm{F}}$, it suffices to show $C \in FRed_{\mathrm{F}}((\{\bot \leftarrow A_i\}_{i=1}^n)_{\mathfrak{J}})$ for every $\mathfrak{J} \supseteq A$. If $A_i \subseteq \mathfrak{J}$ 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 (D6).

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

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*$_\mathrm{F}$ because $\mathsf{p}(x) \neq \mathsf{p}(y)$. The simplest solution is to take **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 $\mathscr{G}_\mathfrak{I}$ generates entire α-equivalence classes (e.g., $\mathscr{G}_\mathfrak{I}(\{\mathsf{p}(x)\}) = \{\mathsf{p}(x), \mathsf{p}(y), \mathsf{p}(z), \dots\}$) or groundings (e.g., $\mathscr{G}_\mathfrak{I}(\{\mathsf{p}(x)\}) = \{\mathsf{p}(a), \mathsf{p}(\mathsf{f}(a)), \dots\}$). 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 20 *Let $\mathscr{N} \subseteq \mathbf{AF}$ be an A-formula set, and let $\mathfrak{I}$ be a propositional interpretation. If $\mathscr{N}$ is saturated w.r.t. SInf and SRed$_\mathrm{I}$, then $\mathscr{N}_\mathfrak{I}$ is saturated w.r.t. FInf and FRed$_\mathrm{I}$.*

Proof Assuming $\iota \in \mathit{FInf}(\mathscr{N}_\mathfrak{I})$, we must show $\iota \in \mathit{FRed}_\mathrm{I}(\mathscr{N}_\mathfrak{I})$. 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 $\iota \in (\mathit{SInf}(\mathscr{N}))_\mathfrak{I}$. This means that there exists a BASE inference $\iota_0 \in \mathit{SInf}(\mathscr{N})$. By saturation of $\mathscr{N}$, we have $\iota_0 \in \mathit{SRed}_\mathrm{I}(\mathscr{N})$. By definition of *SRed*$_\mathrm{I}$, $\{\iota_0\}_\mathfrak{I} = \{\iota\} \subseteq \mathit{FRed}_\mathrm{I}(\mathscr{N}_\mathfrak{I})$, as required. $\square$

Theorem 21 (Static completeness) *Assume $(\mathit{FInf}, \mathit{FRed})$ is statically complete. Then $(\mathit{SInf}, \mathit{SRed})$ is statically complete.*

Proof Suppose $\mathscr{N} \subseteq \mathbf{AF}$, $\mathscr{N} \models \{\bot\}$, and $\mathscr{N}$ is saturated w.r.t. *SInf* and *SRed*$_\mathrm{I}$. We will show $\bot \in \mathscr{N}$.

First, we show $\bot \in \mathscr{N}_\mathfrak{I}$ for every $\mathfrak{I}$. From $\mathscr{N} \models \{\bot\}$, by the definition of $\models$ on A-formulas, it follows that $N_\mathfrak{I} \models \{\bot\}$. Moreover, by Lemma 20, $N_\mathfrak{I}$ is saturated w.r.t. *FInf* and *FRed*$_\mathrm{I}$. By static completeness of $(\mathit{FInf}, \mathit{FRed})$, we get $\bot \in \mathscr{N}_\mathfrak{I}$.

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

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

Corollary 22 (Dynamic completeness) *Assume $(\mathit{FInf}, \mathit{FRed})$ is statically complete. Then $(\mathit{SInf}, \mathit{SRed})$ is dynamically complete.*

Proof This immediately follows from Theorem 21 by Lemma 6 in the technical report of Waldmann et al. [30]. $\square$

3.4 Local Saturation

The above completeness result, about $\rhd_{SRed_F}$-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 $\rhd_{SRed_F}$-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 $\perp \leftarrow A$ such that $A \subseteq \mathcal{J}$ is derivable for some model $\mathcal{J} \models \mathcal{N}_\perp$, the prover will never be able to apply the UNSAT rule to derive $\perp$. It should then be allowed to deliver a verdict of "consistent." We will call such model-specific saturations *local* and standard saturations *global*.

Definition 23 A set $\mathcal{N} \subseteq \mathbf{AF}$ is *locally saturated* w.r.t. *SInf* and *SRed*$_I$ if either $\perp \in \mathcal{N}$ or there exists a propositional model $\mathcal{J} \models \mathcal{N}_\perp$ such that $\mathcal{N}_\mathcal{J}$ is saturated w.r.t. *FInf* and *FRed*$_I$.

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

Theorem 24 (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$_I$ and such that $\mathcal{N} \models \{\perp\}$, we have $\perp \in \mathcal{N}$.*

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

Example 25 Consider the following A-clause set expressed using AVATAR conventions:

$$\{\perp \leftarrow \{\neg[\mathsf{p}(x)], \neg[\mathsf{q}(y)]\}, \qquad \mathsf{p}(x) \leftarrow \{[\mathsf{p}(x)]\}, \qquad \mathsf{q}(y) \leftarrow \{[\mathsf{q}(y)]\}, \qquad \neg\mathsf{q}(\mathsf{a})\}$$

It is not globally saturated for resolution, because the conclusion $\perp \leftarrow \{[\mathsf{q}(y)]\}$ of resolving the last two A-clauses is missing, but it is locally saturated with $\mathcal{J} \supseteq \{[\mathsf{p}(x)], \neg[\mathsf{q}(y)]\}$ as the witness in Definition 23.

We also need a notion of local fairness that works in tandem with local saturation.

Definition 26 A sequence $(\mathcal{N}_i)_i$ of sets of A-formulas is *locally fair* w.r.t. *SInf* and *SRed*$_I$ if either $\perp \in \mathcal{N}_i$ for some i or there exists a propositional model $\mathcal{J} \models (\mathcal{N}_\infty)_\perp$ such that $FInf((\mathcal{N}_\infty)_\mathcal{J}) \subseteq \bigcup_i FRed_I((\mathcal{N}_i)_\mathcal{J})$.

Lemma 27 *Let $(\mathcal{N}_i)_i$ be a $\rhd_{SRed_F}$-derivation that is locally fair w.r.t. SInf and SRed$_I$. Then the limit inferior $\mathcal{N}_\infty$ is locally saturated w.r.t. SInf and SRed$_I$.*

Proof The proof is by case analysis on the condition by which $(\mathcal{N}_i)_i$ is locally fair. If $\perp \in \mathcal{N}_i$, then $\perp \in \mathcal{N}_\infty$ by Lemma 17, and $\mathcal{N}_\infty$ is therefore locally saturated. In the remaining case, we have $N_i \subseteq N_\infty \cup SRed_F(N_\infty)$ by Lemma 4 in the technical report of Waldmann et al. [30], and therefore $\bigcup_i FRed_I((\mathcal{N}_i)_\mathcal{J}) \subseteq \bigcup_i FRed_I((\mathcal{N}_\infty)_\mathcal{J} \cup FRed_F((\mathcal{N}_\infty)_\mathcal{J})) = \bigcup_i FRed_I((\mathcal{N}_\infty)_\mathcal{J})$ because we clearly have $(SRed_F(N_\infty))_\mathcal{J} \subseteq FRed_F((\mathcal{N}_\infty)_\mathcal{J}) \cup (N_\infty)_\mathcal{J}$.   □

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

Theorem 28 (Strong dynamic completeness) *Assume* $(FInf, FRed)$ *is statically complete. Given a* $\rhd_{SRed_F}$*-derivation* $(\mathcal{N}_i)_i$ *that is locally fair w.r.t. SInf and $SRed_I$ 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 $(\bigcup_i \mathcal{N}_i) \setminus SRed_F(\bigcup_i \mathcal{N}_i) \models \{\bot\}$. By their Lemma 2, $(\bigcup_i \mathcal{N}_i) \setminus SRed_F(\bigcup_i \mathcal{N}_i) \subseteq \mathcal{N}_\infty$. Hence $\mathcal{N}_\infty \supseteq (\bigcup_i \mathcal{N}_i) \setminus SRed_F(\bigcup_i \mathcal{N}_i) \models \{\bot\}$. By Lemma 27, $\mathcal{N}_\infty$ is locally saturated, so by Theorem 24, $\bot \in \mathcal{N}_\infty$. 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 analysis 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 22, we get $\bot \in (\mathcal{N}_i)_{\mathcal{J}}$ for some i. By Lemma 2 and the definition of $\rhd_{FRed_F}$, 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 $\rhd_{SRed_F}$-derivations, and their fairness criteria will imply local fairness. Consequently, by Theorem 28, they will all be complete.

4 Model-Guided Provers

The transition system $\rhd_{SRed_F}$ 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 $\rhd_{SRed_F}$-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 interpretation 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 $\mathcal{M} \subseteq SRed_F(\mathcal{N} \uplus \mathcal{M}')$
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 $\mathcal{N}_\bot \not\models \{\bot\}$

The DERIVE rule can add new A-formulas ($\mathcal{M}'$) and delete redundant A-formulas ($\mathcal{M}$). In practice, DERIVE will perform only sound or consistency-preserving inferences, but we impose no such restriction. If soundness of a prover is desired, it can be derived easily from

the soundness of the individual inferences. Similarly, $\mathcal{M}$ and $\mathcal{M}'$ will usually be enabled in $\mathcal{J}$, but we do not require this.

The interpretation $\mathcal{J}$ should be a model of $\mathcal{N}_\perp$ most of the time; when it is not, SWITCH can be used to switch interpretation or STRONGUNSAT to finish the refutation. Although the condition $\mathcal{J}_i \models (\mathcal{N}_i)_\perp$ might be violated for some i, to make progress we must periodically check it and apply SWITCH as needed. Much of the work that is performed while the condition is violated will likely be wasted. To avoid this waste, Vampire invokes the SAT solver whenever it selects a clause as part of the given clause procedure.

Transitions can be combined to form $\Longrightarrow_{\mathsf{MG}}$-derivations (pronounced "arrow-MG-derivations").

Lemma 29 *If* $(\mathcal{J}, \mathcal{N}) \Longrightarrow_{\mathsf{MG}} (\mathcal{J}', \mathcal{N}')$, *then* $\mathcal{N} \rhd_{SRed_\mathsf{F}} \mathcal{N}'$.

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 $\rhd_{SRed_\mathsf{F}}$. $\qquad\square$

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.

Example 30 Let us revisit Example 10. Initially, the propositional interpretation is $\mathcal{J}_0 = \{\neg v_0, \neg v_1\}$. After the split, we have the A-clauses $\neg p(a)$, $\neg q(z,z)$, $p(x) \leftarrow \{v_0\}$, $q(y,b) \leftarrow \{v_1\}$, and $\perp \leftarrow \{\neg v_0, \neg v_1\}$. The natural option is to switch interpretation. We take $\mathcal{J}_1 = \{v_0, \neg v_1\}$. We then derive $\perp \leftarrow \{v_0\}$. Since $\mathcal{J}_1 \not\models \perp \leftarrow \{v_0\}$, we switch to $\mathcal{J}_2 = \{\neg v_0, v_1\}$, where we derive $\perp \leftarrow \{v_1\}$. Finally, we detect that the propositional clauses are unsatisfiable and generate $\perp$. This corresponds to the transitions below, where arrows are annotated by transition names and light gray boxes identify enabled A-clauses:

$$(\mathcal{J}_0, \{\,\boxed{\neg p(a)}\,, \boxed{\neg q(z,z)}\,, \boxed{p(x) \vee q(y,b)}\,\})$$
$$\Longrightarrow_{\text{DERIVE}} (\mathcal{J}_0, \{\,\boxed{\neg p(a)}\,, \boxed{\neg q(z,z)}\,, \perp \leftarrow \{\neg v_0, \neg v_1\}\,, p(x) \leftarrow \{v_0\}, q(y,b) \leftarrow \{v_1\}\})$$
$$\Longrightarrow_{\text{SWITCH}} (\mathcal{J}_1, \{\,\boxed{\neg p(a)}\,, \boxed{\neg q(z,z)}\,, \perp \leftarrow \{\neg v_0, \neg v_1\}\,, \boxed{p(x) \leftarrow \{v_0\}}\,, q(y,b) \leftarrow \{v_1\}\})$$
$$\Longrightarrow_{\text{DERIVE}} (\mathcal{J}_1, \{\,\boxed{\neg p(a)}\,, \boxed{\neg q(z,z)}\,, \perp \leftarrow \{\neg v_0, \neg v_1\}\,, \boxed{p(x) \leftarrow \{v_0\}}\,, q(y,b) \leftarrow \{v_1\},$$
$$\boxed{\perp \leftarrow \{v_0\}}\,\})$$
$$\Longrightarrow_{\text{SWITCH}} (\mathcal{J}_2, \{\,\boxed{\neg p(a)}\,, \boxed{\neg q(z,z)}\,, \perp \leftarrow \{\neg v_0, \neg v_1\}\,, p(x) \leftarrow \{v_0\}, \boxed{q(y,b) \leftarrow \{v_1\}}\,,$$
$$\perp \leftarrow \{v_0\}\})$$
$$\Longrightarrow_{\text{DERIVE}} (\mathcal{J}_2, \{\,\boxed{\neg p(a)}\,, \boxed{\neg q(z,z)}\,, \perp \leftarrow \{\neg v_0, \neg v_1\}\,, p(x) \leftarrow \{v_0\}, \boxed{q(y,b) \leftarrow \{v_1\}}\,,$$
$$\perp \leftarrow \{v_0\}, \boxed{\perp \leftarrow \{v_1\}}\,\})$$
$$\Longrightarrow_{\substack{\text{STRONG-}\\\text{UNSAT}}} (\mathcal{J}_2, \{\,\boxed{\neg p(a)}\,, \boxed{\neg q(z,z)}\,, \perp \leftarrow \{\neg v_0, \neg v_1\}\,, p(x) \leftarrow \{v_0\}, \boxed{q(y,b) \leftarrow \{v_1\}}\,,$$
$$\perp \leftarrow \{v_0\}, \boxed{\perp \leftarrow \{v_1\}}\,, \boxed{\perp}\,\})$$

4.2 Fairness

We need a fairness criterion for MG that implies local fairness of the underlying $\rhd_{SRed_\mathsf{F}}$-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 31 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 $\mathcal{J} = \mathcal{J}'_\infty = \mathcal{J}'^\infty$.
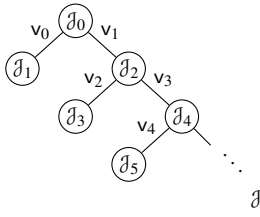
Fig. 1: A split tree with a single infinite branch
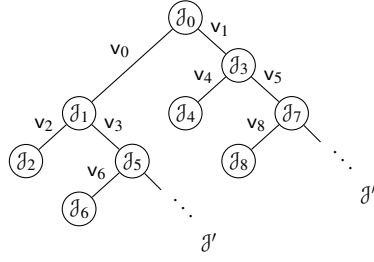
Fig. 2: A split tree with two infinite branches

Intuitively, a limit point is a propositional interpretation that is the limit of a family of interpretations that we revisit infinitely often. We will see that there always exists a limit point. To achieve fairness, we will focus on saturating a limit point.

Example 32 Let $(\mathcal{J}_i)_i$ be the sequence such that $\mathcal{J}_{2i} \cap \mathbf{V} = \{v_1, v_3, \ldots, v_{2i-1}\}$ (i.e., $v_1, v_3, \ldots, v_{2i-1}$ are true and the other variables are false) and $\mathcal{J}_{2i+1} = (\mathcal{J}_{2i} \setminus \{\neg v_{2i}\}) \cup \{v_{2i}\}$. Although it is not in the sequence, the interpretation $\mathcal{J} \cap \mathbf{V} = \{v_1, v_3, \ldots\}$ is a limit point. The split tree of $(\mathcal{J}_i)_i$ 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.

The above example hints at why the proof of MG's dynamic completeness is nontrivial: Some derivations might involve infinitely many split branches, making it difficult for the prover to focus on any single one and saturate it.

Example 33 A sequence may have multiple limit points. Let $(\mathcal{J}_i)_i$ be the sequence such that $\mathcal{J}_0 \cap \mathbf{V} = \emptyset$, $\mathcal{J}_{4i+1} \cap \mathbf{V} = \{v_0\} \cup \{v_{4j+3} \mid j < i\}$, $\mathcal{J}_{4i+2} \cap \mathbf{V} = \{v_0, v_{4i+2}\} \cup \{v_{4j+3} \mid j < i\}$, $\mathcal{J}_{4i+3} \cap \mathbf{V} = \{v_{4j+1} \mid j \le i\}$, and $\mathcal{J}_{4i+4} \cap \mathbf{V} = \{v_{4j+1} \mid j \le i\} \cup \{v_{4i+4}\}$. 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 34 *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 35 *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., such that $\mathcal{J}'_j \models v_k$ if and only if $\mathcal{J} \models v_k$ for every $k \leq j$. Moreover, 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 $\mathcal{J}'_0, \ldots, \mathcal{J}'_j$. Among the infinitely many elements $\mathcal{J}_i$ that agree with $\mathcal{J}'_0, \ldots, \mathcal{J}'_j$ on $v_1, \ldots v_j$, there must be infinitely many with $\mathcal{J}_i \models v_{j+1}$ or infinitely many with $\mathcal{J}_i \models \neg v_{j+1}$. In the first case, set $\mathcal{J}'_{j+1} = \mathcal{J}_i$ for one such index i, and analogously in the second case. $\square$

Lemma 35 tells us that every sequence has a limit point. No matter how erratically the prover switches branches, it will systematically explore at least one branch in a limit point. It then suffices to perform the base *FInf*-inferences fairly in that branch:

Definition 36 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 $FInf((\mathcal{N}_\infty)_{\mathcal{J}}) \subseteq \bigcup_i FRed_{\mathrm{I}}((\mathcal{N}_i)_{\mathcal{J}})$.

Until $\bot$ is derived, it is impossible in a fair $\Longrightarrow_{\mathsf{MG}}$-derivation to delay SWITCH forever (by the first half of (2)) or to starve off DERIVE by performing only SWITCH transitions (by the second half of (2)). 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 expect that a realistic prover would ensure the inclusion $FInf((\mathcal{N}_\infty)_{\mathcal{J}}) \subseteq \bigcup_i FRed_{\mathrm{I}}((\mathcal{N}_i)_{\mathcal{J}})$ for *all* limit points $\mathcal{J}$. However, a prover like Vampire, based on the given-clause procedure with an age-based heuristic, might saturate only one of the limit points, as we will see in Sect. 6.2.

Fairness of $\Longrightarrow_{\mathsf{MG}}$-derivations is deliberately defined in terms of $FRed_{\mathrm{I}}$ instead of $SRed_{\mathrm{I}}$. This results in a more suitable 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. $(SInf, SRed)$. For example, the inference $(t \approx s, \mathsf{p}(t), \mathsf{p}(s))$ is locally redundant in $\mathcal{J} \supseteq \{v_0\}$ if the A-clause $\mathsf{p}(s) \leftarrow \{v_0\}$ has already been derived, but it is not redundant w.r.t. $(SInf, SRed)$.

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. Then for every $\mathcal{D} \in (\mathcal{N}_\infty)_\bot$, there exists an index i such that $\mathcal{J}_j \models \{\mathcal{D}\}$ for every $j \geq i$.*

Proof Let $\mathcal{D} = \bot \leftarrow A \in (\mathcal{N}_\infty)_\bot$. Then $\bot \leftarrow A \in \mathcal{N}_k$ for some k. For every $j \geq k$, we then have $\bot \leftarrow B \in \mathcal{N}_j$ for some $B \subseteq A$, since every $\Longrightarrow_{\mathsf{MG}}$-derivation is a $\rhd_{SRed_\mathsf{F}}$-derivation and $\bot \leftarrow A$ can only become redundant due to such a $\bot \leftarrow B$. Since $\{\bot \leftarrow B\} \models \{\bot \leftarrow A\}$, we get $(\mathcal{N}_j)_\bot \models \{\bot \leftarrow A\}$ for every $j \geq k$. 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 changes only 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 every $j \geq i$. $\square$

Lemma 38 *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 37, there exists an index i such that $\mathcal{J}_j \models \{\mathcal{C}\}$ for every $j \geq i$. 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 every $j \geq i'$ and hence $\mathcal{J} \models \{\mathcal{C}\}$. $\square$

In the spirit of refinement, we have that fairness of an $\Longrightarrow_{\mathsf{MG}}$-derivation implies local fairness of the underlying $\rhd_{SRed_\mathsf{F}}$-derivation:

Theorem 39 (Fairness) *Let $(\mathcal{J}_i, \mathcal{N}_i)_i$ be a fair $\Longrightarrow_{\mathsf{MG}}$-derivation. Then $(\mathcal{N}_i)_i$ is a $\rhd_{SRed_F}$-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 $FInf((\mathcal{N}_\infty)_{\mathcal{J}}) \subseteq \bigcup_i FRed_I((\mathcal{N}_i)_{\mathcal{J}})$. We take this limit point as the witness for $\mathcal{J}$ in Definition 26. It remains to show that $\mathcal{J} \models (\mathcal{N}_\infty)_\bot$. This follows from Lemma 38. $\qquad\square$

Corollary 40 (Dynamic completeness) *Assume $(FInf, FRed)$ is statically complete. Given a fair $\Longrightarrow_{\mathsf{MG}}$-derivation $(\mathcal{J}_i, \mathcal{N}_i)_i$ such that $\mathcal{N}_0 \models \{\bot\}$, we have $\bot \in \mathcal{N}_i$ for some i.*

Proof By Theorem 28. $\qquad\square$

A well-behaved propositional solver, as in labeled splitting, enumerates potential models in a systematic way and always gives rise to a single limit point $\mathcal{J}_\infty$, which can be taken for $\mathcal{J}$ in the definition of fairness (Definition 36). 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 [27, Sect. 4]. However, such a strategy is not sufficient if the prover exploits local redundancy, as we will see in 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 SAT solvers [20] could produce this kind of behavior. Then it is more challenging to ensure fairness, as we will see in Sect. 6.

Example 41 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 30, but not $\bot \leftarrow \{\mathsf{v}_1\}$. By static completeness of the splitting calculus, we conclude that the A-clause set is consistent.

Example 42 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, and without selection [2, Sect. 3], a resolution prover would diverge attempting to generate infinitely many clauses of the form $\mathsf{p}(x) \vee \mathsf{q}(\mathsf{f}^i(\mathsf{a}))$.

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 $\mathsf{p}(x)$. 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 43 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 44 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 $p(f^i(a))$ and $q(f^i(a))$, respectively. If we split every emerging clause $p(f^i(a)) \vee q(f^i(a))$ and the SAT solver always makes v_{2i} true before v_{2i+1}, we end up with the situation of Example 32 and Fig. 1. For the limit point $\mathfrak{J}$, all *FInf*-inferences are performed. Thus, the derivation is fair.

Example 45 We build a clause set from two copies of Example 44, where each clause C from each copy $i \in \{1, 2\}$ is extended to $\neg r_i \vee C$. We add the clause $r_1 \vee r_2$ and split it as our first move. From there, each branch imitates Example 44. A SAT solver might jump back and forth between them, as in Example 33 and Fig. 2. Even if the 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 ($\mathfrak{J}'$ or $\mathfrak{J}''$).

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 and unlock them when coming back to an interpretation where they are no longer locally redundant. In AVATAR, conditionally deleted A-clauses are stored in the locked set; in labeled splitting, they are stored in the split stack. We will 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 $(\mathfrak{J}, \mathcal{N}, \mathcal{L})$, where $\mathfrak{J}$ is a propositional interpretation, $\mathcal{N} \subseteq \mathbf{AF}$ is a set of A-formulas, and $\mathcal{L} \subseteq \mathscr{P}_{\text{fin}}(\mathbf{A}) \times \mathbf{AF}$ is a set of pairs of finite assertion sets and A-formulas. Intuitively, $(B, C \leftarrow A) \in \mathcal{L}$ means that $C \leftarrow A$ is "locally redundant" in all interpretations $\mathfrak{J} \supseteq B$. The function $\|\ \|$ erases the locks: $\|\mathcal{L}\| = \{\mathscr{C} \mid (B, \mathscr{C}) \in \mathcal{L} \text{ for some } B\}$. Initial states have the form $(\mathfrak{J}, N, \emptyset)$, where $N \subseteq \mathbf{F}$. The *locking prover* is defined by two transition rules:

Lift $\quad (\mathfrak{J}, \mathcal{N}, \mathcal{L}) \Longrightarrow_{\mathsf{L}} (\mathfrak{J}', \mathcal{N}' \cup \|\mathscr{U}\|, \mathcal{L} \setminus \mathscr{U})$
$\qquad$ if $(\mathfrak{J}, \mathcal{N}) \Longrightarrow_{\mathsf{MG}} (\mathfrak{J}', \mathcal{N}')$ and $\mathscr{U} = \{(B, C \leftarrow A) \in \mathcal{L} \mid B \nsubseteq \mathfrak{J}' \text{ and } A \subseteq \mathfrak{J}'\}$

Lock $\quad (\mathfrak{J}, \mathcal{N} \uplus \{C \leftarrow A\}, \mathcal{L}) \Longrightarrow_{\mathsf{L}} (\mathfrak{J}, \mathcal{N}, \mathcal{L} \cup \{(B, C \leftarrow A)\})$
$\qquad$ if $B \subseteq \mathfrak{J}$ and $C \in FRed_{\mathsf{F}}(\mathcal{N}_{\mathfrak{J}'})$ for every $\mathfrak{J}' \supseteq A \cup B$

The Lift rule performs an $\Longrightarrow_{\mathsf{MG}}$-transition and unlocks any A-formulas that are no longer locally reundant. The Lock rule can be used to lock A-formulas that are locally redundant.

Lemma 46 *Let $(\mathfrak{J}_i, \mathcal{N}_i, \mathcal{L}_i)_i$ be an $\Longrightarrow_{\mathsf{L}}$-derivation. Then $(\mathfrak{J}_i, \mathcal{N}_i \cup \|\mathcal{L}_i\|)_i$ is an $\Longrightarrow_{\mathsf{MG}}$-derivation.*

Proof Every Lift transition clearly corresponds to an MG transition. Every Lock transition corresponds to a Derive transition with $\mathscr{M} = \mathscr{M}' = \emptyset$. $\qquad \square$

Example 47 Let $\mathfrak{J}_0 = \{\neg v_0\}$ and $\mathfrak{J}_1 = \{v_0\}$. The following derivation based on first-order resolution illustrates the locking and unlocking of an A-clause:

$$(\mathfrak{J}_0, \{\,\boxed{\neg p(a)}\,, p(x) \leftarrow \{\neg v_0\}, p(a)\,\}, \emptyset)$$

$\Longrightarrow_{\text{LOCK}}\ (\mathcal{J}_0, \{\boxed{\neg\mathsf{p}(\mathsf{a})}, \mathsf{p}(x)\leftarrow\{\neg\mathsf{v}_0\}\}, \{(\{\neg\mathsf{v}_0\}, \mathsf{p}(\mathsf{a}))\})$

$\Longrightarrow_{\text{LIFT}}\ (\mathcal{J}_0, \{\boxed{\neg\mathsf{p}(\mathsf{a})}, \mathsf{p}(x)\leftarrow\{\neg\mathsf{v}_0\}, \bot\leftarrow\{\neg\mathsf{v}_0\}\}, \{(\{\neg\mathsf{v}_0\}, \mathsf{p}(\mathsf{a}))\})$

$\Longrightarrow_{\text{LIFT}}\ (\mathcal{J}_1, \{\boxed{\neg\mathsf{p}(\mathsf{a})}, \mathsf{p}(x)\leftarrow\{\neg\mathsf{v}_0\}, \bot\leftarrow\{\neg\mathsf{v}_0\}, \boxed{\mathsf{p}(\mathsf{a})}\}, \emptyset)$

$\Longrightarrow_{\text{LIFT}}\ (\mathcal{J}_1, \{\boxed{\neg\mathsf{p}(\mathsf{a})}, \mathsf{p}(x)\leftarrow\{\neg\mathsf{v}_0\}, \bot\leftarrow\{\neg\mathsf{v}_0\}, \boxed{\mathsf{p}(\mathsf{a})}, \bot\}, \emptyset)$

Gray boxes indicate enabled unlocked clauses. We first put a lock on $\mathsf{p}(\mathsf{a})$, because it is "locally subsumed" by $\mathsf{p}(x)\leftarrow\{\neg\mathsf{v}_0\}$ in $\mathcal{J}_0$. Once we switch to $\mathcal{J}_1$, the lock is released, and we can use $\mathsf{p}(\mathsf{a})$ to conclude the refutation.

There are three things to note. First, if we had simply thrown away the clause $\mathsf{p}(\mathsf{a})$ instead of locking it, we would have lost refutability. Second, it would have been advantageous not to lock $\mathsf{p}(\mathsf{a})$ at all and to use it immediately to derive $\bot$; however, it is not difficult to come up with examples where locking actually helps, which is why AVATAR includes this mechanism. Third, although the derivation shows only "local subsumption," it could easily be changed to perform "local simplification"—e.g., demodulation from an equation $s\approx t\leftarrow A$.

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 $\mathsf{p}(\mathsf{c})$ is locally redundant in any interpretation $\mathcal{J}$ that contains $\neg\mathsf{v}_k$. If the sequence of interpretations is given by $\mathcal{J}_i = \{\mathsf{v}_0, \ldots, \mathsf{v}_{i-1}, \neg\mathsf{v}_i, \neg\mathsf{v}_{i+1}, \ldots\}$, 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 48 Consider the inconsistent initial clause set

$$\{\mathsf{t}(\mathsf{a}), \quad \neg\mathsf{t}(x)\vee\mathsf{t}(\mathsf{f}(x)), \quad \neg\mathsf{t}(x)\vee\neg\mathsf{s}(x)\vee\neg\mathsf{r}(x,y)\vee\mathsf{q}(y),$$
$$\neg\mathsf{p}(\mathsf{c}), \quad \neg\mathsf{q}(\mathsf{c}), \quad \mathsf{r}(x,y), \quad \neg\mathsf{r}(x,y)\vee\neg\mathsf{r}(x,z)\vee\mathsf{q}(x)\vee\mathsf{p}(x)\}$$

and ordered resolution with selection as the calculus. Assume that the selection function always chooses the maximal negative literals w.r.t. the precedence $\mathsf{p}\prec\mathsf{q}\prec\mathsf{r}\prec\mathsf{s}\prec\mathsf{t}$. Let $fml(\mathsf{v}_i) = \neg\mathsf{r}(\mathsf{f}^i(\mathsf{a}),x)\vee\mathsf{q}(x)$ and $fml(\mathsf{w}_i) = \neg\mathsf{s}(\mathsf{f}^i(\mathsf{a}))$, and let v_i and w_i be false in the initial model for all i. Following an age-based selection heuristic and maximal splitting, the second clause the prover derives is $\neg\mathsf{s}(\mathsf{a})\vee\neg\mathsf{r}(\mathsf{a},y)\vee\mathsf{q}(y)$, which it splits into $\neg\mathsf{s}(\mathsf{a})\leftarrow\{\mathsf{w}_0\}$ and $\neg\mathsf{r}(\mathsf{a},y)\vee\mathsf{q}(y)\leftarrow\{\mathsf{v}_0\}$. The s predicate's role is purely to ensure that this clause is split and that the assertion v_0 is introduced. The prover later also derives $\mathsf{q}(x)\vee\mathsf{p}(x)$ and $\mathsf{q}(y)\leftarrow\{\mathsf{v}_0\}$ and switches to a model in which v_i is true if and only if $i = 0$. The first of the two clauses is clearly locally redundant, so LOCK applies, and $(\{\mathsf{v}_0\}, \mathsf{q}(x)\vee\mathsf{p}(x))$ is added to $\mathscr{L}$.

Next, $\mathsf{q}(y)\leftarrow\{\mathsf{v}_1\}$ is derived, before $\mathsf{q}(y)\leftarrow\{\mathsf{v}_0\}$ is selected for inferences. Eventually, that latter clause can be used together with $\neg\mathsf{q}(\mathsf{c})$ to derive $\bot\leftarrow\{\mathsf{v}_0\}$. The prover then switches to a new model in which v_i is true if and only if $i = 1$. The clause $\mathsf{q}(x)\vee\mathsf{p}(x)$ can immediately be relocked. This process can be repeated indefinitely. The clause $\mathsf{q}(x)\vee\mathsf{p}(x)$, which is necessary for a refutation (together with $\neg\mathsf{p}(\mathsf{c})$ and $\neg\mathsf{q}(\mathsf{c})$), is 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 48, locking is not applied exhaustively: The A-clause $\neg\mathsf{r}(\mathsf{f}^i(\mathsf{a}),y)\vee\mathsf{q}(y)\leftarrow\{\mathsf{v}_i\}$ is not locked, even though $\mathsf{q}(y)\leftarrow\{\mathsf{v}_j\}$ has already been derived.

This situation is unrealistic and would not happen in Vampire. We could hope that is enough for completeness to forbid such anomalous scenarios. However, this is not the case, as we can see from a more complicated example:

Example 49 The calculus is ordered resolution with selection using the precedence $p \prec q_1 \prec q_2 \prec r_1 \prec r_2 \prec s \prec t_1 \prec t_2 \prec u$, selecting nothing if the clause is of the form $\neg u(\ldots) \vee u(\ldots)$ and otherwise selecting the maximal negative literals.

The initial clauses are as follows. First, we have a splittable clause $q_1(x) \vee q_2(y)$. Then we have clauses $u(a, y)$ and $\neg u(x, y) \vee u(f(x), y)$. We will use the predicate symbol u to delay the selection of a clause in an age-based selection heuristic, by adding a literal $\neg u(f^j(x), y)$ to the clause. Moreover, we have the clause $s(x, y)$. We can prevent splitting by adding the literal $\neg s(x, y)$ to a clause. Finally, we add the following clauses:

$$\neg u(f(y), x) \vee r_1(x) \vee \neg q_1(x) \qquad \neg u(f(y), x) \vee \neg r_1(x) \qquad \neg t_1(x) \vee \neg s(x, y) \vee \neg p(x) \vee r_1(y)$$
$$\neg u(f(y), x) \vee r_2(x) \vee \neg q_2(x) \qquad \neg u(f(y), x) \vee \neg r_2(x) \qquad \neg t_2(x) \vee \neg s(x, y) \vee \neg p(x) \vee r_2(y)$$
$$t_1(a) \qquad t_2(f(a)) \qquad \neg t_1(x) \vee t_1(f(f(x))) \qquad \neg t_2(x) \vee t_2(f(f(x)))$$

The initial clause set is clearly inconsistent. Yet we will sketch an infinite derivation that corresponds to an age-based selection heuristic and that does not derive $\bot$. First, we split $q_1(x) \vee q_2(y)$ into $q_1(x) \leftarrow \{x_1\}$ and $q_2(x) \leftarrow \{x_2\}$, where the assertion denotations are as follows:

$$fml(w_{2i}) = \neg s(f^{2i}(a), x) \vee r_1(x) \qquad\qquad fml(v_i) = \neg p(f^i(a))$$
$$fml(w_{2i+1}) = \neg s(f^{2i+1}(a), x) \vee r_2(x) \qquad\qquad fml(x_i) = q_i(x)$$

The derivation uses the following sequence of interpretations $\mathcal{J}_i$:

- $\mathcal{J}_i \models v_j$ if and only if $j < i$;
- $\mathcal{J}_i \models w_j$ if and only if $j \in \{i, i+1\}$;
- $\mathcal{J}_i \models x_1$ if and only if i is even;
- $\mathcal{J}_i \models x_2$ if and only if i is odd.

The derivation thus alternates between two families of interpretations, with even and odd indices, giving rise to two limit points.

After the clause $q_1(x) \vee q_2(y)$ is split, the prover is in the model $\mathcal{J}_0$ and derives the clauses $\neg u(y, x) \vee r_1(x) \vee \neg q_1(x)$, $\neg u(y, x) \vee r_2(x) \vee \neg q_2(x)$, $\neg s(a, y) \vee r_1(y) \vee \neg p(a)$. The last clause is split into $\neg s(a, y) \vee r_1(y) \leftarrow \{w_0\}$, $\neg p(a) \leftarrow \{v_0\}$, and $\bot \leftarrow \{\neg v_0, \neg w_0\}$. Then an analogous split happens with r_2 instead of r_1. After a few more inferences, we derive $r_1(x) \vee \neg q_1(x)$ and then $r_1(y) \leftarrow \{w_0\}$, which makes $r_1(x) \vee \neg q_1(x)$ locally redundant (and analogously for r_2 in place of r_1). Once $r_1(y) \leftarrow \{w_0\}$ is picked as the given clause, the prover derives $\bot \leftarrow \{w_0\}$ and switches to the next model $\mathcal{J}_1$.

In the first family, $\mathcal{J}_{2i}$, the clause $\neg q_1(x) \vee r_1(x)$ is always locally redundant due to $r_1(x) \leftarrow \{w_{2i}\}$, and is locked with the assertion w_{2i}. Similarly, $\neg q_2(x) \vee r_2(x)$ is locally redundant in the second family, $\mathcal{J}_{2i+1}$, with the assertion w_{2i+1}. In both cases we can already lock each of the clauses while the prover is still in the previous model ($\mathcal{J}_{2i-1}$ and $\mathcal{J}_{2i}$, respectively).

The clause $\neg q_2(x) \vee r_2(x)$ is thus only ever unlocked in interpretations $\mathcal{J}_{2i}$. In those interpretations, $q_2(x) \leftarrow \{x_2\}$ is disabled and hence no inferences can be performed with $\neg q_2(x) \vee r_2(x)$. The same holds *mutatis mutandis* for $\neg q_1(x) \vee r_1(x)$, which is unlocked only when no inferences can be performed with it. As a result, the derivation never performs inferences with $q_1(x) \leftarrow \{x_1\}$ or $q_2(x) \leftarrow \{x_2\}$. Removing these A-clauses makes the set satisfiable; thus, by soundness, the derivation cannot contain $\bot$.

Given the right sequence of propositional interpretations returned by the SAT solver, the derivation in Example 49 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 SAT solvers in such a concrete way. This derivation is also strongly fair—every inference that is possible infinitely often, perhaps intermittently, is eventually made redundant. Thus strong fairness is not a sufficient criterion for completeness either.

5.3 Fairness

Our solution to the issues encountered above is as follows. Let $(\mathcal{J}_i, \mathcal{N}_i, \mathcal{L}_i)_i$ be an $\Longrightarrow_{\mathsf{L}}$-derivation. Given a subsequence $(\mathcal{J}'_j)_j$ of $(\mathcal{J}_i)_i$, let $(\mathcal{N}'_j)_j$ be the corresponding subsequence of $(\mathcal{N}_i)_i$. To achieve fairness, we now consider $\mathcal{N}'_\infty$, the A-formulas persistent in the 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 50 An $\Longrightarrow_{\mathsf{L}}$-derivation $(\mathcal{J}_i, \mathcal{N}_i, \mathcal{L}_i)_i$ is *fair* if (A) $\mathcal{L}_0 = \emptyset$ and (B) 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 $FInf((\mathcal{N}'_\infty)_{\mathcal{J}} \cup (\limsup_{j\to\infty}\|\mathcal{L}'_j\|)_{\mathcal{J}} \setminus \|\mathcal{L}'^\infty\|_{\mathcal{J}}) \subseteq \bigcup_i FRed_{\mathsf{I}}((\mathcal{N}_i \cup \|\mathcal{L}_i\|)_{\mathcal{J}})$, 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. In particular, if we know that each A-formula is locked at most finitely often, then $\limsup_{j\to\infty}\|\mathcal{L}'_j\| = \|\mathcal{L}'^\infty\|$ and the inclusion in the definition above simplifies to $FInf((\mathcal{N}'_\infty)_{\mathcal{J}}) \subseteq \bigcup_i FRed_{\mathsf{I}}((\mathcal{N}_i \cup \|\mathcal{L}_i\|)_{\mathcal{J}})$.

Theorem 51 (Fairness) *Let $(\mathcal{J}_i, \mathcal{N}_i, \mathcal{L}_i)_i$ be a fair $\Longrightarrow_{\mathsf{L}}$-derivation. Then $(\mathcal{J}_i, \mathcal{N}_i \cup \|\mathcal{L}_i\|)_i$ is a fair $\Longrightarrow_{\mathsf{MG}}$-derivation.*

Proof We already showed that $(\mathcal{J}_i, \mathcal{N}_i, \mathcal{L}_i)_i$ is an $\Longrightarrow_{\mathsf{MG}}$-derivation in Lemma 46. It remains to show fairness. If the $\Longrightarrow_{\mathsf{L}}$-derivation is fair by case (1) of Definition 50, we apply case (1) of Definition 36 to show that the $\Longrightarrow_{\mathsf{MG}}$-derivation is fair. Otherwise, from case (2) of Definition 50, we retrieve a limit point $\mathcal{J}$. We will show, for that limit point, case (2) of Definition 36:

$$FInf((\liminf_{i\to\infty}\mathcal{N}_i \cup \|\mathcal{L}_i\|)_{\mathcal{J}}) \subseteq \bigcup_i FRed_{\mathsf{I}}((\mathcal{N}_i \cup \|\mathcal{L}_i\|)_{\mathcal{J}})$$

Assume (a) $\iota \in FInf((\liminf_{j\to\infty}\mathcal{N}_j \cup \|\mathcal{L}_j\|)_{\mathcal{J}})$. By the definition of fairness of $\Longrightarrow_{\mathsf{L}}$-derivations, if all of ι's premises belong to $(\mathcal{N}'_\infty)_{\mathcal{J}} \cup ((\limsup_{j\to\infty}\|\mathcal{L}'_j\|)_{\mathcal{J}} \setminus (\|\mathcal{L}'^\infty\|)_{\mathcal{J}})$, then ι is redundant. Otherwise, we have that (b) one of ι's premises, C, is not in that set; that is, $C \notin (\mathcal{N}'_\infty)_{\mathcal{J}}$ and either $C \notin (\limsup_{j\to\infty}\|\mathcal{L}'_j\|)_{\mathcal{J}}$ or $C \in (\|\mathcal{L}'^\infty\|)_{\mathcal{J}}$.

By (a) we have that $C \leftarrow A \in \liminf_{j\to\infty}\mathcal{N}_j \cup \|\mathcal{L}_j\| \subseteq \liminf_{j\to\infty}\mathcal{N}'_j \cup \|\mathcal{L}'_j\|$ for some $A \subseteq \mathcal{J}$. Since $C \notin (\mathcal{N}'_\infty)_{\mathcal{J}}$ by (b), $C \leftarrow A$ cannot be persistent in $(\mathcal{N}'_j)_j$ and hence must occur infinitely often in the sequence $(\|\mathcal{L}'_j\|)_j$. Thus $C \in (\limsup_{j\to\infty}\|\mathcal{L}'_j\|)_{\mathcal{J}}$ and therefore $C \in (\|\mathcal{L}'^\infty\|)_{\mathcal{J}}$ by (b).

Hence $(B, C \leftarrow A') \in \mathcal{L}'^\infty$ for some $A' \subseteq \mathcal{J}$ and B. If $(B, C \leftarrow A') \in \mathcal{L}'_j$ for some j, then necessarily $B \subseteq \mathcal{J}'_j$ due to the side conditions of the $\Longrightarrow_{\mathsf{L}}$-transitions. Since this is true for infinitely many indices j, we also have $B \subseteq \mathcal{J}'^\infty = \mathcal{J}$, and thus $C \in FRed((\mathcal{N}_i)_{\mathcal{J}})$ for some i by

the side condition of the Lock transition. Therefore, by reducedness of *FRed*, the inference ι is redundant.   □

Corollary 52 (Dynamic completeness) *Assume* (*FInf*, *FRed*) *is statically complete. Given a fair* $\Longrightarrow_{\mathsf{L}}$*-derivation* $(\mathcal{J}_i, \mathcal{N}_i, \mathcal{L}_i)_i$ *such that* $\mathcal{N}_0 \models \{\bot\}$*, we have* $\bot \in \mathcal{N}_i$ *for some i.*

Proof By Theorems 28 and 39.   □

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, 22]. At least two reimplementations exist, in Ebner's `super` tactic for Lean [12] and in the Drodi prover by Oscar Contreras. Here we attempt to capture AVATAR's essence.

We will define an abstract AVATAR-based prover that extends the locking prover L with a given clause procedure [17, Sect. 2.3]. 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*. (We will often omit the adjective "timestamped.") Given a subset $\mathcal{N} \subseteq \mathbf{TAF}$, we define $\wr\mathcal{N}\wr = \{\mathscr{C} \mid (\mathscr{C}, t) \in \mathcal{N} \text{ for some } t\}$ and overload existing notations to erase timestamps as necessary. Accordingly, $\lfloor\mathcal{N}\rfloor = \lfloor\wr\mathcal{N}\wr\rfloor$, $\mathcal{N}_\bot = \wr\mathcal{N}\wr_\bot$, and $\mathcal{N}_{\mathcal{J}} = \wr\mathcal{N}\wr_{\mathcal{J}}$. Note that we use a new set of calligraphic letters (e.g., $\mathscr{C}, \mathcal{N}$) to range over timestamped A-formulas and timestamped A-formula sets. We say that $\mathcal{N}$ is enabled in $\mathcal{J}$ if and only if $\wr\mathcal{N}\wr$ is enabled in $\mathcal{J}$. We also define $\wr(\mathscr{C}_1, \ldots, \mathscr{C}_n, \mathcal{D})\wr = (\wr\mathscr{C}_1\wr, \ldots, \wr\mathscr{C}_n\wr, \wr\mathcal{D}\wr)$ for **TAF**-inferences ι.

Using the saturation framework [29, Sect. 3], we lift a calculus (*SInf*, *SRed*) on **AF** to a calculus (*TSInf*, *TSRed*) on **TAF** with the tiebreaker order $<$ on timestamps. The tiebreaker is used to strengthen redundancy, so that if the same A-formula appears but with two different timestamps, the more recent version is considered redundant. In other words, if an A-formula appears with two timestamps, the later version is redundant. Note that *TSRed* is in general not reduced. 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.

Lemma 53 *Let* $\mathcal{N} \subseteq \mathbf{AF}$*,* $\mathscr{C} \in \mathbf{AF}$*, and* $t, k \in \mathbb{N}$*. Then:*

1. $\wr TSInf(\mathcal{N})\wr = SInf(\wr\mathcal{N}\wr)$*;*
2. $\wr TSRed_{\mathrm{I}}(\mathcal{N})\wr = SRed_{\mathrm{I}}(\wr\mathcal{N}\wr)$*;*
3. $\wr TSRed_{\mathrm{F}}(\mathcal{N})\wr \supseteq SRed_{\mathrm{F}}(\wr\mathcal{N}\wr)$*; and*
4. $(\mathscr{C}, t + k) \in TSRed_{\mathrm{F}}(\{(\mathscr{C}, t)\})$ *if* $k > 0$*.*

Proof This follows directly from the definition in Waldmann et al. [29].   □

A state is a tuple $(\mathcal{J}, \mathcal{A}, \mathcal{P}, \mathcal{Q}, \mathcal{L})$ consisting of a propositional interpretation $\mathcal{J}$, a set of enabled nonpropositional *active* A-formulas $\mathcal{A} \subseteq \mathbf{TAF}$, a set of enabled nonpropositional *passive* A-formulas $\mathcal{P} \subseteq \mathbf{TAF}$, a set of A-formulas $\mathcal{Q} \subseteq \mathbf{TAF}$ that are either disabled in $\mathcal{J}$ or propositional clauses, and a set of locked A-formulas $\mathcal{L} \subseteq \mathscr{P}_{\mathrm{fin}}(\mathbf{A}) \times \mathbf{TAF}$ such that

(1) $\mathcal{A}_\perp = \mathcal{P}_\perp = \emptyset$; $\qquad\qquad$ (2) $\mathcal{A} \cup \mathcal{P}$ is enabled in $\mathcal{J}$; $\qquad\qquad$ (3) $\mathcal{Q}_\mathcal{J} \subseteq \{\perp\}$.

Whenever we write a tuple $(\mathcal{J}, \mathcal{A}, \mathcal{P}, \mathcal{Q}, \mathcal{L})$, we assume that it satisfies all these invariants. For every $\mathcal{L} \subseteq \mathbf{A} \times \mathbf{TAF}$, we define $\lfloor\mathcal{L}\rfloor = \{(B, \lfloor\mathcal{C}\rfloor) \mid (B, \mathcal{C}) \in \mathcal{L}\} \subseteq \mathbf{A} \times \mathbf{AF}$.

The input formulas are first put in the passive set $\mathcal{P}$. Once an A-formula is selected for inferences and all inferences with it and the active A-formulas have been made redundant, it is moved to the active set $\mathcal{A}$. Inferences such as SPLIT produce disabled and propositional clauses, which are put into $\mathcal{Q}$. When switching to a new model, the prover moves the newly enabled A-formulas from $\mathcal{Q}$ to $\mathcal{P}$ and the newly disabled A-formulas from $\mathcal{A}$ and $\mathcal{P}$ 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; for example, $\mathcal{P}$ is a separate set because fairness will be stated in terms of $\mathcal{A}$ and $\mathcal{P}$. In a practical implementation, this division would likely be different. The set $\mathcal{Q}$ would typically be distributed over two data structures: The propositional clauses in $\mathcal{Q}_\perp$ are directly passed to the SAT solver and need not be stored by the prover itself. The remaining A-formulas $\mathcal{Q} \setminus \mathcal{Q}_\perp$ are those that need to be moved back into $\mathcal{P}$ when the prover switches to an interpretation that enables them. 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. This is what Vampire does. Alternatively, they could 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.

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

INFER $\qquad$ $(\mathcal{J}, \mathcal{A}, \mathcal{P} \uplus \{\mathcal{C}\}, \mathcal{Q}, \mathcal{L}) \Longrightarrow_{\mathsf{AV}} (\mathcal{J}, \mathcal{A} \cup \{\mathcal{C}\}, \mathcal{P}', \mathcal{Q}', \mathcal{L})$
$\qquad\qquad$ if $\mathit{TSInf}(\mathcal{A}, \{\mathcal{C}\}) \subseteq \mathit{TSRed}_{\mathrm{I}}(\mathcal{A} \cup \{\mathcal{C}\} \cup \mathcal{P}' \cup \mathcal{Q}')$,
$\qquad\qquad\quad$ $\mathcal{P} \subseteq \mathcal{P}'$, and $\mathcal{Q} \subseteq \mathcal{Q}'$

PROCESS $\qquad$ $(\mathcal{J}, \mathcal{A}, \mathcal{P}, \mathcal{Q}, \mathcal{L}) \Longrightarrow_{\mathsf{AV}} (\mathcal{J}, \mathcal{A}', \mathcal{P}', \mathcal{Q}', \mathcal{L})$
$\qquad\qquad$ if $\mathcal{A} \supseteq \mathcal{A}'$ and
$\qquad\qquad\quad$ $(\mathcal{A} \setminus \mathcal{A}') \cup (\mathcal{P} \setminus \mathcal{P}') \cup (\mathcal{Q} \setminus \mathcal{Q}') \subseteq \mathit{TSRed}_{\mathrm{F}}(\mathcal{A}' \cup \mathcal{P}' \cup \mathcal{Q}')$

SWITCH $\qquad$ $(\mathcal{J}, \mathcal{A}, \mathcal{P}, \mathcal{Q}, \mathcal{L}) \Longrightarrow_{\mathsf{AV}} (\mathcal{J}', \mathcal{A}', \mathcal{P}' \cup \lfloor\!\lfloor\mathcal{U}\rfloor\!\rfloor, \mathcal{Q}', \mathcal{L} \setminus \mathcal{U})$
$\qquad\qquad$ if $\mathcal{J} \not\models \mathcal{Q}_\perp$, $\mathcal{J}' \models \mathcal{Q}_\perp$, $\mathcal{A}' = \{\mathcal{C} \in \mathcal{A} \mid \mathcal{C}$ is enabled in $\mathcal{J}'\}$,
$\qquad\qquad\quad$ $\mathcal{U} = \{(B, (C \leftarrow A, t)) \in \mathcal{L} \mid B \not\subseteq \mathcal{J}'$ and $A \subseteq \mathcal{J}'\}$, and
$\qquad\qquad\quad$ $\mathcal{A} \cup \mathcal{P} \cup \mathcal{Q} = \mathcal{A}' \cup \mathcal{P}' \cup \mathcal{Q}'$

STRONGUNSAT $\qquad$ $(\mathcal{J}, \mathcal{A}, \mathcal{P}, \mathcal{Q}, \mathcal{L}) \Longrightarrow_{\mathsf{AV}} (\mathcal{J}, \mathcal{A}, \mathcal{P}, \mathcal{Q} \cup \{(\perp, t)\}, \mathcal{L})$
$\qquad\qquad$ if $\mathcal{Q}_\perp \not\approx \{\perp\}$

LOCKA $\qquad$ $(\mathcal{J}, \mathcal{A} \uplus \{(C \leftarrow A, t)\}, \mathcal{P}, \mathcal{Q}, \mathcal{L}) \Longrightarrow_{\mathsf{AV}}$
$\qquad\qquad$ $(\mathcal{J}, \mathcal{A}, \mathcal{P}, \mathcal{Q}, \mathcal{L} \cup \{(B, (C \leftarrow A, t))\})$
$\qquad\qquad$ 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. In practice, a prover would ensure well-timestampedness by assigning timestamps monotonically, but this is not necessary for the fairness and completeness proofs.

Lemma 54 *Let* $(\mathcal{J}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{Q}_i, \mathcal{L}_i)_i$ *be an* $\Longrightarrow_{\mathsf{AV}}$-*derivation. Then* $(\mathcal{J}_i, \lfloor\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i\rfloor, \lfloor\mathcal{L}_i\rfloor)_i$ *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. $\qquad\qquad\square$

Lemma 55 *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* $TSInf(\mathcal{A}_i) \subseteq$
$TSRed_{\mathrm{I}}(\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \lfloor\!\lfloor \mathcal{L}_i \rfloor\!\rfloor)$ *for every i.*

Proof The invariant is preserved by each transition.   □

Example 56 Let us redo the $\Longrightarrow_{\mathsf{MG}}$-derivation of Example 30 using $\Longrightarrow_{\mathsf{AV}}$. For readability, we emphasize in gray the A-clauses that appear or move between state components and omit all timestamps. One possible derivation is

$$(\mathcal{J}_0, \emptyset, \{\neg\mathsf{p}(\mathsf{a}), \neg\mathsf{q}(z,z), \mathsf{p}(x) \vee \mathsf{q}(y,\mathsf{b})\}, \emptyset, \emptyset)$$

$\Longrightarrow_{\text{INFER}}$ $(\mathcal{J}_0, \{\,\boxed{\neg\mathsf{p}(\mathsf{a})}\,\}, \{\neg\mathsf{q}(z,z), \mathsf{p}(x) \vee \mathsf{q}(y,\mathsf{b})\}, \emptyset, \emptyset)$

$\Longrightarrow_{\text{INFER}}$ $(\mathcal{J}_0, \{\neg\mathsf{p}(\mathsf{a}), \boxed{\neg\mathsf{q}(z,z)}\,\}, \{\mathsf{p}(x) \vee \mathsf{q}(y,\mathsf{b})\}, \emptyset, \emptyset)$

$\Longrightarrow_{\text{PROCESS}}$ $(\mathcal{J}_0, \{\neg\mathsf{p}(\mathsf{a}), \neg\mathsf{q}(z,z)\}, \emptyset,$
$\qquad\qquad \{\,\boxed{\bot \leftarrow \{\neg\mathsf{v}_0, \neg\mathsf{v}_1\}}\,, \boxed{\mathsf{p}(x) \leftarrow \{\mathsf{v}_0\}}\,, \boxed{\mathsf{q}(y,\mathsf{b}) \leftarrow \{\mathsf{v}_1\}}\,\}, \emptyset)$

$\Longrightarrow_{\text{SWITCH}}$ $(\mathcal{J}_1, \{\neg\mathsf{p}(\mathsf{a}), \neg\mathsf{q}(z,z)\}, \{\,\boxed{\mathsf{p}(x) \leftarrow \{\mathsf{v}_0\}}\,\},$
$\qquad\qquad \{\bot \leftarrow \{\neg\mathsf{v}_0, \neg\mathsf{v}_1\}, \mathsf{q}(y,\mathsf{b}) \leftarrow \{\mathsf{v}_1\}\}, \emptyset)$

$\Longrightarrow_{\text{INFER}}$ $(\mathcal{J}_1, \{\neg\mathsf{p}(\mathsf{a}), \neg\mathsf{q}(z,z), \boxed{\mathsf{p}(x) \leftarrow \{\mathsf{v}_0\}}\,\}, \emptyset,$
$\qquad\qquad \{\bot \leftarrow \{\neg\mathsf{v}_0, \neg\mathsf{v}_1\}, \mathsf{q}(y,\mathsf{b}) \leftarrow \{\mathsf{v}_1\}, \boxed{\bot \leftarrow \{\mathsf{v}_0\}}\,\}, \emptyset)$

$\Longrightarrow_{\text{SWITCH}}$ $(\mathcal{J}_2, \{\neg\mathsf{p}(\mathsf{a}), \neg\mathsf{q}(z,z)\}, \{\,\boxed{\mathsf{q}(y,\mathsf{b}) \leftarrow \{\mathsf{v}_1\}}\,\},$
$\qquad\qquad \{\bot \leftarrow \{\neg\mathsf{v}_0, \neg\mathsf{v}_1\}, \bot \leftarrow \{\mathsf{v}_0\}, \boxed{\mathsf{p}(x) \leftarrow \{\mathsf{v}_0\}}\,\}, \emptyset)$

$\Longrightarrow_{\text{INFER}}$ $(\mathcal{J}_2, \{\neg\mathsf{p}(\mathsf{a}), \neg\mathsf{q}(z,z), \boxed{\mathsf{q}(y,\mathsf{b}) \leftarrow \{\mathsf{v}_1\}}\,\}, \emptyset,$
$\qquad\qquad \{\bot \leftarrow \{\neg\mathsf{v}_0, \neg\mathsf{v}_1\}, \bot \leftarrow \{\mathsf{v}_0\}, \mathsf{p}(x) \leftarrow \{\mathsf{v}_0\}, \boxed{\bot \leftarrow \{\mathsf{v}_1\}}\,\}, \emptyset)$

$\Longrightarrow_{\substack{\text{STRONG-}\\\text{UNSAT}}}$ $(\mathcal{J}_2, \{\neg\mathsf{p}(\mathsf{a}), \neg\mathsf{q}(z,z), \mathsf{q}(y,\mathsf{b}) \leftarrow \{\mathsf{v}_1\}\}, \emptyset,$
$\qquad\qquad \{\bot \leftarrow \{\neg\mathsf{v}_0, \neg\mathsf{v}_1\}, \bot \leftarrow \{\mathsf{v}_0\}, \mathsf{p}(x) \leftarrow \{\mathsf{v}_0\}, \bot \leftarrow \{\mathsf{v}_1\}, \boxed{\bot}\,\}, \emptyset)$

Example 57 Let us redo the $\Longrightarrow_{\mathsf{L}}$-derivation of Example 47 using $\Longrightarrow_{\mathsf{AV}}$, following the same conventions as in the previous example. One possible derivation is

$$(\mathcal{J}_0, \emptyset, \{\neg\mathsf{p}(\mathsf{a}), \mathsf{p}(x) \leftarrow \{\neg\mathsf{v}_0\}, \mathsf{p}(\mathsf{a})\}, \emptyset, \emptyset)$$

$\Longrightarrow_{\text{LOCKP}}$ $(\mathcal{J}_0, \emptyset, \{\neg\mathsf{p}(\mathsf{a}), \mathsf{p}(x) \leftarrow \{\neg\mathsf{v}_0\}\}, \emptyset, \{(\{\neg\mathsf{v}_0\}, \boxed{\mathsf{p}(\mathsf{a})}\,)\})$

$\Longrightarrow_{\text{INFER}}$ $(\mathcal{J}_0, \{\,\boxed{\neg\mathsf{p}(\mathsf{a})}\,\}, \{\mathsf{p}(x) \leftarrow \{\neg\mathsf{v}_0\}\}, \emptyset, \{(\{\neg\mathsf{v}_0\}, \mathsf{p}(\mathsf{a}))\})$

$\Longrightarrow_{\text{INFER}}$ $(\mathcal{J}_0, \{\neg\mathsf{p}(\mathsf{a}), \boxed{\mathsf{p}(x) \leftarrow \{\neg\mathsf{v}_0\}}\,\}, \emptyset, \{\,\boxed{\bot \leftarrow \{\neg\mathsf{v}_0\}}\,\}, \{(\{\neg\mathsf{v}_0\}, \mathsf{p}(\mathsf{a}))\})$

$\Longrightarrow_{\text{SWITCH}}$ $(\mathcal{J}_1, \{\neg\mathsf{p}(\mathsf{a})\}, \{\,\boxed{\mathsf{p}(\mathsf{a})}\,\}, \{\bot \leftarrow \{\neg\mathsf{v}_0\}, \boxed{\mathsf{p}(x) \leftarrow \{\neg\mathsf{v}_0\}}\,\}, \emptyset)$

$\Longrightarrow_{\text{INFER}}$ $(\mathcal{J}_1, \{\neg\mathsf{p}(\mathsf{a}), \boxed{\mathsf{p}(\mathsf{a})}\,\}, \emptyset, \{\bot \leftarrow \{\neg\mathsf{v}_0\}, \mathsf{p}(x) \leftarrow \{\neg\mathsf{v}_0\}, \boxed{\bot}\,\}, \emptyset)$

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 an even stronger fairness criterion.

A problematic scenario involves two premises $\mathcal{C}, \mathcal{D}$ of a binary inference ι and four transitions repeated forever, 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.

Example 58 More concretely, make two copies of the clause set

$$\{\neg p(x) \vee p(f(x)) \vee q(x), \qquad p(a), \qquad \neg q(x), \qquad \neg p(x) \vee q(f(x))\}$$

one with the assertion $\{x_1\}$, the other with $\{x_2\}$, in addition to the propositional clause $\bot \leftarrow \{\neg x_1, \neg x_2\}$. Suppose the prover starts with the model $\{x_1\}$ and processes the clauses in the order given above. It first chooses the A-clause $\neg p(x) \vee p(f(x)) \vee q(x) \leftarrow \{x_1\}$ for inferences, followed by $p(a) \leftarrow \{x_1\}$. Let $fml(v_i) = p(f^i(a))$ and $fml(w_i) = q(f^i(a))$. By resolution and splitting, it derives $p(f(a)) \leftarrow \{v_1\}$, $q(a) \leftarrow \{w_0\}$, and $\bot \leftarrow \{x_1, \neg v_1, \neg w_0\}$. It then switches to a model in which x_2 is true. There it selects $\neg p(x) \vee p(f(x)) \vee q(x) \leftarrow \{x_2\}$ and $p(a) \leftarrow \{x_2\}$ for inferences, deriving analogous A-clauses as in the x_1 branch.

Let the models alternate between the x_1 and x_2 branches, making as few variables true as possible. Because the models alternate between the two branches, $\neg p(x) \vee p(f(x)) \vee q(x) \leftarrow \{x_i\}$ will always be the oldest passive A-clause after switching to a new model. Assume that the prover chooses this clause for inferences based on age. If we are allowed to interleave age-based selection with heuristic selection, we can cause the prover to switch model after selecting at most two additional A-clauses for inferences: If an A-clause $q(f^j(a)) \leftarrow \{w_j\}$ is enabled, we heuristically select both that A-clause and $\neg q(x) \leftarrow \{x_i\}$. Otherwise, an A-clause of the form $p(f^j(a)) \leftarrow \{v_j\}$ is enabled. Assume that j is maximal among such clauses, and that thus v_{j+1} is false in the model. We heuristically select that clause for inferences, deriving $\bot \leftarrow \{\neg v_{j+1}, \neg w_j, v_j, x_i\}$ by splitting $p(f^{j+1}(a)) \vee q(f^j(a)) \leftarrow \{v_j, x_i\}$.

With this strategy, the prover will never select $\neg p(x) \vee q(f(x)) \leftarrow \{x_i\}$ for inferences, since there is always an older clause to choose first. Consequently, it will never derive $\bot$.

In Example 58, the prover did not derive $\bot$ because it never performed an inference between $p(a) \leftarrow \{x_1\}$ and $\neg p(x) \vee q(f(x)) \leftarrow \{x_1\}$ (and analogously for x_2), even though both A-clauses are enabled infinitely often. Forbidding this situation does not guarantee completeness either. As Example 49 showed, there exist strongly fair $\Longrightarrow_L$-derivations that do not derive $\bot$ from an inconsistent initial set.

We believe that this counterexample cannot arise with Vampire, because Vampire alternates between age-based and weight-based selection using a fixed ratio (the "age–weight ratio" or "pick–given ratio"). In contrast, our example requires a highly unrestricted heuristic selection, where we choose young, large A-clauses such as $p(f^n(a)) \leftarrow \{v_n\}$ even though smaller, older ones are enabled.

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

Example 59 Take the following consistent A-clause set:

$$\{\neg q(x) \leftarrow \{x\}, \qquad p(a) \leftarrow \{x\}, \qquad \neg p(x) \vee p(f(x)) \vee q(f(x)) \leftarrow \{x\},$$
$$\neg q(x) \leftarrow \{\neg x\}, \qquad p(a) \leftarrow \{\neg x\}, \qquad \neg p(x) \leftarrow \{\neg x\}\}$$

Assume ordered resolution as the base calculus with the precedence $q \prec p$. Thus the prover will not resolve $\neg q(x) \leftarrow \{x\}$ with $\neg p(x) \vee p(f(x)) \vee q(f(x)) \leftarrow \{x\}$. We will sketch a derivation with two limit points, $\mathcal{J} \models x$ and $\mathcal{J}' \not\models x$, where $\mathcal{J}'$ will not be locally saturated. Let $fml(v_i) = p(f^i(a))$ and $fml(w_i) = q(f^i(a))$. We define the sequence of models $(\mathcal{J}_i)_i$ such that:

$$\mathcal{J}_{2i} \models x \qquad \mathcal{J}_{2i} \models v_j \text{ iff } j \leq i \qquad \mathcal{J}_{2i} \not\models w_j$$
$$\mathcal{J}_{2i+1} \not\models x \qquad \mathcal{J}_{2i+1} \models v_j \qquad \mathcal{J}_{2i+1} \models w_j \text{ iff } j > i$$

The prover now starts in the model $\mathcal{J}_0$, and processes the formulas in the order we listed them at the beginning of the example. The first new formula it derives is $\mathsf{p}(\mathsf{f}(\mathsf{a})) \vee \mathsf{q}(\mathsf{f}(\mathsf{a})) \leftarrow \{x\}$, which it splits into $\mathsf{p}(\mathsf{f}(\mathsf{a})) \leftarrow \{\mathsf{v}_1\}$, $\mathsf{q}(\mathsf{f}(\mathsf{a})) \leftarrow \{\mathsf{w}_1\}$, and $\bot \leftarrow \{x, \neg \mathsf{v}_1, \neg \mathsf{w}_1\}$. The last propositional clause is not satisfied in $\mathcal{J}_0$, so the prover switches to a new interpretation.

After switching to the next model $\mathcal{J}_1 = \mathcal{J}_{2\cdot 0 + 1}$, the two formulas $\mathsf{p}(\mathsf{f}(\mathsf{a})) \leftarrow \{\mathsf{v}_1\}$ and $\mathsf{q}(\mathsf{f}(\mathsf{a})) \leftarrow \{\mathsf{w}_1\}$ remain in the active set. The prover then chooses the oldest enabled passive formula, $\neg \mathsf{q}(x) \leftarrow \{\neg x\}$, for inferences. Thus deriving the propositional clause $\bot \leftarrow \{\neg x, \mathsf{w}_1\}$, which is not satisfied in $\mathcal{J}_1$.

This process is then repeated infinitely often. In the model $\mathcal{J}_{2i}$, the prover derives the three new formulas $\mathsf{p}(\mathsf{f}^{i+1}(\mathsf{a})) \leftarrow \{\mathsf{v}_{i+1}\}$, $\mathsf{q}(\mathsf{f}^{i+1}(\mathsf{a})) \leftarrow \{\mathsf{w}_{i+1}\}$, and $\bot \leftarrow \{x, \neg \mathsf{v}_{i+1}, \neg \mathsf{w}_{i+1}\}$. The last propositional clause causes a switch to the next model $\mathcal{J}_{2i+1}$, where $\bot \leftarrow \{\neg x, \mathsf{w}_{i+1}\}$ is derived.

The subsequence $(\mathcal{J}_{2i+1})_i$ converges a limit point, call it $\mathcal{J}'$. The formulas enabled at $\mathcal{J}'$ are not saturated: $\mathsf{p}(\mathsf{a})$ and $\neg \mathsf{p}(x)$ are enabled, but $\bot$ is not.

6.3 Fairness

Definition 60 An $\Longrightarrow_{\mathsf{AV}}$-derivation $(\mathcal{J}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{Q}_i, \mathcal{L}_i)_i$ is *fair* if (A) $\mathcal{L}_0 = \emptyset$, (B) $\mathcal{A}_0 = \emptyset$, and (C) either (1) $\bot \in \mathcal{l} \bigcup_i \mathcal{Q}_i \mathcal{S}$ 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 $\mathcal{J}$ such that (3) $\liminf_{j \to \infty} TSInf(\mathcal{A}'_j, \mathcal{P}'_j) = \emptyset$ and (4) $(\limsup_{j \to \infty} \lVert \mathcal{L}'_j \rVert)_{\mathcal{J}} \setminus \lVert \mathcal{L}'^{\infty} \rVert_{\mathcal{J}} \subseteq \bigcup_i FRed_{\mathsf{F}}((\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \lVert \mathcal{L}_i \rVert)_{\mathcal{J}})$.

Condition (3) ensures 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 58. Condition (4) is similar to the condition on locks in Definition 50.

Theorem 61 (Fairness) *Let* $(\mathcal{J}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{Q}_i, \mathcal{L}_i)_i$ *be a fair* $\Longrightarrow_{\mathsf{AV}}$-derivation. Then the $\Longrightarrow_{\mathsf{L}}$-derivation $(\mathcal{J}_i, \mathcal{l} \mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \mathcal{S}, \mathcal{l} \mathcal{L}_i \mathcal{S})_i$ *is fair.*

Proof We trivially have $\mathcal{l} L_0 \mathcal{S} = \emptyset$. Furthermore, if $\bot \in \bigcup_i \mathcal{Q}_i$, we clearly have $\Longrightarrow_{\mathsf{L}}$-fairness. It remains to show subcase (B)(2) of Definition 50, using the corresponding subsequence as used for $\Longrightarrow_{\mathsf{AV}}$-fairness. So let $\clubsuit_{\mathsf{L}} = (\limsup_{j \to \infty} \lVert \mathcal{l} \mathcal{L}'_j \mathcal{S} \rVert)_{\mathcal{J}} \setminus \lVert \limsup_{j \to \infty} \mathcal{l} \mathcal{L}'_j \mathcal{S} \rVert_{\mathcal{J}}$ and $\clubsuit_{\mathsf{AV}} = (\limsup_{j \to \infty} \lVert \mathcal{L}'_j \rVert)_{\mathcal{J}} \setminus \lVert \mathcal{L}'^{\infty} \rVert_{\mathcal{J}}$ be the terms from the corresponding fairness conditions.

First we show $(\bigcup_j \mathcal{A}'_j \cup \mathcal{P}'_j \cup \lVert \mathcal{L}'_j \rVert)_{\mathcal{J}} \subseteq (\liminf_{j \to \infty} \mathcal{A}'_j \cup \mathcal{P}'_j)_{\mathcal{J}} \cup \bigcup_i FRed_{\mathsf{F}}((\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \lVert \mathcal{L}_i \rVert)_{\mathcal{J}})$, that is, every enabled formula in the subsequence is either persistent or redundant on the base level. So let $(C \leftarrow A, t) \in P'_j$. We prove the statement by induction on (A, t) w.r.t. the lexicographic order. If $C \in \clubsuit_{\mathsf{AV}}$ or $C \in \lVert \limsup_{j \to \infty} \mathcal{l} \mathcal{L}'_j \mathcal{S} \rVert_{\mathcal{J}}$, we are done. Otherwise $C \notin \lVert \mathcal{L}'^{\infty} \rVert_{\mathcal{J}}$ (because $\lVert \mathcal{L}'^{\infty} \rVert_{\mathcal{J}} \subseteq \lVert \limsup_{j \to \infty} \mathcal{l} \mathcal{L}'_j \mathcal{S} \rVert_{\mathcal{J}}$) and hence $C \notin (\limsup_{j \to \infty} \lVert \mathcal{L}'_j \rVert)_{\mathcal{J}}$ (because $C \notin \clubsuit_{\mathsf{AV}}$). So since $(C \leftarrow A, t)$ is not locked infinitely often, there exists an index after which it is never locked again, which means that it is either persistent in $(\mathcal{A}'_j \cup \mathcal{P}'_j)_j$ (and we are done) or deleted in PROCESS and thus $(C \leftarrow A, t) \in \bigcup_i TSRed_{\mathsf{F}}(\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i)$. By definition of $TSRed_{\mathsf{F}}$, either (a) $C \in \bigcup_i FRed_{\mathsf{F}}((\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i)_{\mathcal{J}})$, (b) $C \leftarrow B \in \bigcup_i \mathcal{l} \mathcal{A}_i \cup \mathcal{P}_i \mathcal{S}$ for some $B \subset A$, or (c) $(C \leftarrow A, s) \in \bigcup_i \mathcal{A}_i \cup \mathcal{P}_i$ for some $s < t$. In case (a), we are done. In cases (b) and (c), we apply the induction hypothesis.

Now let $R = \bigcup_i FRed_{\mathsf{I}}((\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \lVert \mathcal{L}_i \rVert)_{\mathcal{J}})$ and compute

$$FInf((\liminf_{j \to \infty} \mathcal{l} \mathcal{A}'_j \cup \mathcal{P}'_j \cup \mathcal{Q}'_j \mathcal{S})_{\mathcal{J}} \cup \clubsuit_{\mathsf{L}})$$
$$= FInf((\liminf_{j \to \infty} \mathcal{l} \mathcal{A}'_j \cup \mathcal{P}'_j \mathcal{S})_{\mathcal{J}} \cup \clubsuit_{\mathsf{L}}) \qquad \text{(since by fairness } \bot \notin (\mathcal{Q}'_j)_{\mathcal{J}} \text{ and}$$

$$\hspace{4cm}\text{hence } \mathcal{Q}'_j \text{ is disabled at } \liminf)$$

$$\subseteq FInf((\liminf_{j\to\infty} \mathcal{A}'_j \cup \mathcal{P}'_j)_{\mathcal{J}} \cup \textstyle\bigcup_i FRed_{\mathsf{F}}((\mathcal{A}_i \cup \mathcal{P}_i)_{\mathcal{J}})) \hspace{2cm} (\text{as shown above})$$

$$\subseteq FInf((\liminf_{j\to\infty} \mathcal{A}'_j \cup \mathcal{P}'_j)_{\mathcal{J}}) \cup R \hspace{2cm} (\text{by reducedness of } FRed)$$

$$= (TSInf(\liminf_{j\to\infty} \mathcal{A}'_j \cup \mathcal{P}'_j))_{\mathcal{J}} \cup R \hspace{2cm} (\text{by Lemmas 13 and 53})$$

$$= (\liminf_{j\to\infty} TSInf(\mathcal{A}'_j \cup \mathcal{P}'_j))_{\mathcal{J}} \cup R \hspace{2cm} (\text{by property of } \liminf)$$

$$= (\liminf_{j\to\infty} TSInf(\mathcal{A}'_j) \cup TSInf(\mathcal{A}'_j, \mathcal{P}'_j))_{\mathcal{J}} \cup R \hspace{1cm} (\text{by definition of } TSInf(\ ,\ ))$$

$$\subseteq (\textstyle\bigcup_i TSRed_{\mathrm{I}}(\mathcal{A}'_i \cup \mathcal{P}'_i \cup \mathcal{Q}'_i \cup \lfloor\!\lfloor \mathcal{L}'_i \rfloor\!\rfloor) \cup \liminf_{j\to\infty} TSInf(\mathcal{A}'_j, \mathcal{P}'_j))_{\mathcal{J}} \cup R \hspace{0.5cm} (\text{by Lemma 55})$$

$$\subseteq (\textstyle\bigcup_i TSRed_{\mathrm{I}}(\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \lfloor\!\lfloor \mathcal{L}_i \rfloor\!\rfloor))_{\mathcal{J}} \cup R \hspace{2cm} (\text{by } \Longrightarrow_{\mathsf{AV}}\text{-fairness})$$

$$= \textstyle\bigcup_i FRed_{\mathrm{I}}((\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \lfloor\!\lfloor \mathcal{L}_i \rfloor\!\rfloor)_{\mathcal{J}}) \hspace{2cm} (\text{by Lemmas 16 and 53}) \quad \square$$

Corollary 62 (Dynamic completeness) *Assume* $(FInf, FRed)$ *is statically complete. Given a fair* $\Longrightarrow_{\mathsf{AV}}$*-derivation* $(\mathcal{J}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{Q}_i, \mathcal{L}_i)_i$ *such that* $\mathcal{P}_0 \cup \mathcal{Q}_0 \models \{\bot\}$*, we have* $\bot \in \mathcal{Q}_i$ *for some* i*.*

Proof By Theorems 28, 39, and 51. $\hspace{6cm}\square$

Assuming the restriction on locking we already required for $\Longrightarrow_{\mathsf{L}}$-derivations, the $\Longrightarrow_{\mathsf{AV}}$ relation is concrete enough to allow us to show that typical clause selection strategies are fair and avoid the counterexamples from Sects. 5.2 and 6.2. 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 $\prec$. Intuitively, $\mathcal{C} \prec \mathcal{D}$ if the prover will select $\mathcal{C}$ before $\mathcal{D}$ whenever both are present. That is, the prover always chooses one of the $\prec$-minimal A-formulas. We use two selection orders: $\prec_{\mathbf{TAF}}$, on timestamped A-formulas, must be followed infinitely often; $\prec_{\mathbf{F}}$, on base formulas, must be followed otherwise.

Definition 63 Let X be a set. A *selection order* $\prec$ on X is an irreflexive and transitive relation such that $\{y \mid y \nsucc x\}$ is finite for every $x \in X$.

Example 64 Let $X \subseteq \mathbf{TAF}$ be such that X only contains finitely many A-formulas with the same timestamp. Define $\prec_{\mathrm{age}}$ on X so that $(\mathscr{C}, t) \prec_{\mathrm{age}} (\mathscr{C}', t')$ if and only if $t < t'$. Then $\prec_{\mathrm{age}}$ is a selection order corresponding to age-based selection.

Remark 65 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 $\omega + 1 = \{0 < 1 < 2 < \cdots < \omega\}$ is not a selection order since $\{y \mid y \nsucc \omega\}$ 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 $\mathbf{TAF}$ also generalize the mechanism, outlined by Bachmair and Ganzinger in a footnote [2, Sect. 4.3] and elaborated by Schlichtkrull et al. [27, Sect. 4], of using an $\mathbb{N}$-valued weight function that is strictly monotone in the timestamp.

Example 66 Let $\mathbf{F}$ be the set of first-order clauses in a fixed signature. Define the selection order $\prec_{\mathrm{nv}}$ on $\mathbf{F}$ by $C' \prec_{\mathrm{nv}} C$ if and only if $|C'| \le |C|$, where $|C|$ denotes the sum of the number of nonvariable positions in C. Then $\prec_{\mathrm{nv}}$ is a selection order because there exists at most 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 $\prec_1$ and $\prec_2$ corresponds to the nondeterministic alternation between them. The prover may choose either a $\prec_1$-minimal or a $\prec_2$-minimal A-formula, at its discretion.

Lemma 67 *Let $\prec_1$ and $\prec_2$ be selection orders on X. Then $\prec_1 \cap \prec_2$ 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 68 *Let $\prec$ be a selection order on an infinite set X. Then for all elements x and y, there exists an element 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 58. Given $\mathcal{N} \subseteq \mathbf{AF}$, let $\lceil \mathcal{N} \rceil = \bigcup \{A \mid C \leftarrow A \in \mathcal{N} \text{ for some } C\}$.

Definition 69 A function $F : \mathscr{P}(\mathbf{AF}) \to \mathscr{P}(\mathbf{AF})$ is called *strongly finitary* if (1) $\lfloor F(\mathcal{N}) \rfloor$ is finite for every $\mathcal{N} \subseteq \mathbf{AF}$ such that $\lfloor \mathcal{N} \rfloor$ is finite, and (2) there exists a function $B : \mathbf{F} \to \mathscr{P}_{\text{fin}}(\mathbf{A})$ such that $\lceil F(\mathcal{N}) \rceil \subseteq \lceil \mathcal{N} \rceil \cup B(\lfloor \mathcal{N} \rfloor)$ for every $\mathcal{N} \subseteq \mathbf{AF}$.

A set of $\mathbf{AF}$-inferences *Inf* is *strongly finitary* if the function $\mathcal{N} \mapsto concl(Inf(\mathcal{N}))$ is strongly finitary. An inference rule is *strongly finitary* if the set of inferences it defines is strongly finitary. We can extend a strongly finitary function F to sets of base formulas by taking $F_{\mathbf{F}}(N) = \lfloor F(N \times \mathscr{P}_{\text{fin}}(\mathbf{A})) \rfloor$ for every $N \subseteq \mathbf{F}$ and to sets of timestamped A-formulas by taking $F_{\mathbf{TAF}}(\mathcal{N}) = F(\lfloor \mathcal{N} \rfloor) \times \mathbb{N}$ for every $\mathcal{N} \subseteq \mathbf{TAF}$. The functions F and $F_{\mathbf{F}}$ are finitary, mapping finite sets to finite sets. Moreover, if F and G are strongly finitary, then so is $\mathcal{N} \mapsto F(\mathcal{N}) \cup G(\mathcal{N})$.

The function B in Definition 69 gives a bound on the new assertions. For the inference rules BASE, UNSAT, COLLECT, TRIM, and STRONGUNSAT, we can set $B(N) = \emptyset$ since the conclusions do not contain assertions which were not already in the premises. So if $FInf(N)$ is finite for every finite $N \subseteq \mathbf{F}$, then *SInf* is strongly finitary. The inference rules SPLIT and APPROX require a nonempty $B(N)$, containing the assertions chosen for the split A-formulas. Deterministic splitting rules (where the chosen assertions are fully determined by the base formula), such as AVATAR's, are thus also strongly finitary because then $B(N)$ is finite. The optional inference rule TAUTO is not strongly finitary.

For the completeness of the given clause procedure, Lemma 77, we will fix a strongly finitary function I restricting the inferences: The prover may perform an inference only if its conclusion is in $I(\mathscr{A}_i)$, where $\mathscr{A}_i$ is the active clause set after the INFER transition. This restriction will allow us to rule out the case where $\lfloor \bigcup_i \mathscr{A}_i \rfloor$ is finite and the prover switches models without making progress. Condition (1) in Definition 69 then says that the prover only infers finitely many $\mathbf{F}$-formulas—this will in turn ensure that splitting creates only finitely many new assertions. Condition (2) says that the inferred A-formulas contain only finitely many new assertions. Taken together, only finitely many assertions are added in the case where $\lfloor \bigcup_i \mathscr{A}_i \rfloor$ is finite, which means that the prover can only switch models finitely often, a contradiction.

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 $p(x*0) \vee p(x)$ to $p(0) \vee p(x)$ transforms a nonsplittable clause into a splittable one. Even for the standard orders on first-order clauses, there can be infinitely many clauses that are smaller than 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 simplifications were to produce infinitely many new splittable clauses, the prover might split clauses and switch propositional interpretations forever without making progress.

Example 70 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 every inference $(C_2, C_1, D) \in I$, simplification with I can still produce infinitely many base formulas. This is because in an AV prover, the same base formula may be rederived 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 $0 \prec 1 \prec 2 \prec \cdots \prec \infty$, and let $I = \{(n, \infty, n+1) \mid n \in \mathbb{N}\}$. If the prover then rederives ∞ infinitely often, it might simplify ∞ using I 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 in at most a finite number of ways.

Definition 71 Let $\prec$ be a transitive well-founded relation on $\mathbf{F}$, and let $\preceq$ be its reflexive closure. A function $S : \mathbf{AF} \to \mathscr{P}(\mathbf{AF})$ is a *strongly finitary simplification bound* for $\prec$ if $\mathscr{N} \mapsto \bigcup_{\mathscr{C} \in \mathscr{N}} S(\mathscr{C})$ is strongly finitary and $\lfloor \mathscr{C}' \rfloor \preceq \lfloor \mathscr{C} \rfloor$ for every $\mathscr{C}' \in S(\mathscr{C})$.

The prover may simplify an A-formula $\mathscr{C}$ to $\mathscr{C}'$ only if $\mathscr{C}' \in S(\mathscr{C})$. It may also delete $\mathscr{C}$. 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. Analogously to strongly finitary functions, we define the extension of strongly finitary simplification bounds to sets of formulas as $S_{\mathbf{F}}(N) = \lfloor S(N \times \mathscr{P}_{\mathrm{fin}}(\mathbf{A})) \rfloor$. 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 switches: 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 72 *Let S be a strongly finitary simplification bound. For every $\mathscr{C} \in \mathbf{AF}$, let $S^*(\mathscr{C}) = \bigcup_{i=0}^{\infty} S^i(\mathscr{C})$, where S^i denotes the ith iterate of S. Then S^* is also a strongly finitary simplification bound.*

Proof Clearly, $\lfloor \mathscr{C}' \rfloor \preceq \lfloor \mathscr{C} \rfloor$ for every $\mathscr{C}' \in S^*(\mathscr{C})$. Let $N \subseteq \mathbf{F}$ be finite. Next we show that $S^*_{\mathbf{F}}(N)$ is finite as well. 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, $S_{\mathbf{F}}(M_\infty) \cup N \subseteq M_\infty = S_{\mathbf{F}}^*(N) \supseteq S^*_{\mathbf{F}}(N)$.

To show that M_∞ is finite, consider the sequence $M_{i+1} \setminus M_i$. By construction $S_{\mathbf{F}}(M_i) \setminus M_i \subseteq (S_{\mathbf{F}}(M_i \setminus M_{i-1}) \cup S_{\mathbf{F}}(M_{i-1})) \setminus M_i \subseteq (S_{\mathbf{F}}(M_i \setminus M_{i-1}) \cup M_i) \setminus M_i = S_{\mathbf{F}}(M_i \setminus M_{i-1}) \setminus M_i$, and thus $M_{i+1} \setminus M_i = S_{\mathbf{F}}(M_i \setminus M_{i-1}) \setminus M_i$. Because S is a strongly finitary simplification bound, $M_{i+1} \setminus M_i$ is always finite, and decreasing in the multiset extension of $\prec$. It is even strictly decreasing as long as $M_{i+1} \setminus M_i \neq \emptyset$ because $M_{i+1} \setminus M_i \cap M_i \setminus M_{i-1} = \emptyset$ and therefore $M_{i+1} \setminus M_i \neq M_i \setminus M_{i+1}$. From the well-foundedness of $\prec$, 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.

Thus $\lfloor S^*(\mathscr{N}) \rfloor$ is finite whenever $\lfloor \mathscr{N} \rfloor$ is finite. It remains to exhibit a function $B' : \mathbf{F} \to \mathscr{P}_{\mathrm{fin}}(\mathbf{A})$ such that $\lceil S^*(\mathscr{N}) \rceil \subseteq B'(\lfloor \mathscr{N} \rfloor) \cup \lceil \mathscr{N} \rceil$. By assumption, we have such a function B

for S. Then set $B'(C) = B(S^*_\mathbf{F}(C))$ (which is finite for all C), and we have $\lceil S^{i+1}(\mathcal{N}) \rceil = \lceil S(S^i(\mathcal{N})) \rceil \subseteq B(\lfloor S^i(\mathcal{N}) \rfloor) \cup \lceil S^i(\mathcal{N}) \rceil \subseteq B'(\lfloor \mathcal{N} \rfloor) \cup \lceil S^i(\mathcal{N}) \rceil$ and therefore $\lceil S^i(\mathcal{N}) \rceil \subseteq \lceil \mathcal{N} \rceil \cup B'(\lfloor \mathcal{N} \rfloor)$ by induction and hence $\lceil S^*(\mathcal{N}) \rceil \subseteq \lceil \mathcal{N} \rceil \cup B'(\lfloor \mathcal{N} \rfloor)$. □

Example 73 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 $C \lor L \lor L$ to $C \lor L$), deletion of resolved literals (simplifying $C \lor u \not\approx u$ to C), and subsumption resolution (simplifying $C\sigma \lor D\sigma \lor L\sigma$ to $C\sigma \lor D\sigma$ given the side premise $C \lor \neg L$). Removing redundant clauses is possible with every S.

Example 74 If the Knuth–Bendix order [16] is used as the term order and all weights are positive, then $S(C \leftarrow A) = \{C' \leftarrow A' \mid C' \prec C$ and $A' \subseteq A\}$ is a strongly finitary simplification bound. This can be used to cover demodulation.

Example 75 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 74 is not strongly finitary, and proving that demodulation is a strongly finitary simplification bound is much more involved. In this case, the necessary strongly finitary simplification bound even depends on the derivation.

Example 76 If unit equations are only removed by demodulation, reflexivity deletion, or subsumption, the one-step demodulations possible at any point in the derivation are a strongly finitary simplification bound. By Lemma 72, this implies that many-step demodulation is also a strongly finitary simplification bound.

Assume that demodulation is performed in a postorder traversal (i.e., subterms first), always rewriting using the oldest available equation. 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 exist only finitely many 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, let $|l|$ be the number of nonvariable positions in l, and let $u = 1$ 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 left-to-right lexicographic order as we move along the derivation.

If the prover derives a new ordered equation $l' \approx r'$, 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 $r\sigma \succ t\sigma$.

Based on 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.

Lemma 77 *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. *$\prec_{\mathbf{TAF}}$ is a selection order on $\bigcup_i \mathcal{P}_i$, and $\prec_{\mathbf{F}}$ is a selection order on $\mathbf{F}$;*
2. *$\mathcal{A}_0 = \emptyset$, $\mathcal{L}_0 = \emptyset$, and $\mathcal{P}_0 \cup \mathcal{Q}_0$ is finite;*
3. *for every INFER transition, either $\mathcal{C}$ is $\prec_{\mathbf{TAF}}$-minimal in $\mathcal{P}$ or $\lfloor \mathcal{C} \rfloor$ is $\prec_{\mathbf{F}}$-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, $\mathcal{P}' \cup \mathcal{Q}' \subseteq S^*_{\mathbf{TAF}}(\mathcal{A} \cup \mathcal{P} \cup \mathcal{Q} \cup \lfloor\!\lfloor \mathcal{L} \rfloor\!\rfloor)$;*
6. *if $\mathcal{J}_i \not\models (\mathcal{Q}_i)_\perp$, 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 $\prec_{\mathbf{TAF}}$-minimal $\mathcal{C}$ at i;*
9. *$(\limsup_{j \to \infty} \lfloor\!\lfloor \mathcal{L}'_j \rfloor\!\rfloor)_{\mathcal{J}} \setminus \lfloor\!\lfloor \mathcal{L}'^\infty \rfloor\!\rfloor_{\mathcal{J}} \subseteq \bigcup_i FRed_{\mathbf{F}}((\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \lfloor\!\lfloor \mathcal{L}_i \rfloor\!\rfloor)_{\mathcal{J}})$ for every subsequence $(\mathcal{J}'_j)_j$ of $(\mathcal{J}_i)_i$ converging to a limit point $\mathcal{J}$.*

Proof If $\perp \in \bigcup_i \langle \mathcal{Q}_i \rangle$, the derivation is trivially fair. Otherwise, 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, thus satisfying condition (C)(2) of Definition 36. Conditions (A) and (B) are satisfied due to condition (2) of this lemma, and (C)(4) due to (9). 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 $\liminf_{j \to \infty} TSInf(\mathcal{A}'_j, \mathcal{P}'_j) = \emptyset$, as required for (C)(3).

CASE 1: The set of $\prec_{\mathbf{F}}$-minimal A-formulas selected in an INFER transition for some state j is unbounded in $\prec_{\mathbf{F}}$. 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 $\prec_{\mathbf{F}}$-minimal in $\mathcal{P}_j$, and $C \prec_{\mathbf{F}} \lfloor \mathcal{S} \rfloor$. These INFER transitions clearly form an infinite subsequence. By Lemma 35, we can further refine it 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 $\iota \in \liminf_{j \to \infty} TSInf(\mathcal{A}'_j, \mathcal{P}'_j)$. By Lemma 68, for every $\mathcal{C} \in prems(\iota)$ there exists an index j such that $\lfloor \mathcal{C} \rfloor \prec_{\mathbf{F}} \lfloor \mathcal{S}'_j \rfloor$. Therefore $prems(\iota) \subseteq \mathcal{A}'_j$ by the $\prec_{\mathbf{F}}$-minimality requirement on the INFER transition, a contradiction.

CASE 2: The set of $\prec_{\mathbf{TAF}}$-minimal A-formulas selected in an INFER transition for some state j is unbounded in $\prec_{\mathbf{TAF}}$. This case is analogous to case 1.

CASE 3: Neither case 1 nor case 2 apply. Then the set of $\prec_{\mathbf{F}}$-minimal formulas selected in an INFER transitions is bounded and hence finite since $\prec_{\mathbf{F}}$ is a selection order. Similarly, the set of $\prec_{\mathbf{TAF}}$-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 $\bigcup_i \lfloor \mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \lfloor\!\lfloor \mathcal{L}_i \rfloor\!\rfloor \rfloor$ is finite as well. Since both S^* and I are strongly finitary, only a finite number of new assertions are introduced, and $\bigcup_i \langle \mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \lfloor\!\lfloor \mathcal{L}_i \rfloor\!\rfloor \rangle$ 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 exists 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 $\prec_{\mathbf{TAF}}$-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 A-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 $\liminf_{i \to \infty} TSInf(\mathcal{A}_i, \mathcal{P}_i) = \emptyset$. $\square$

Recall the abstract 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.

We could refine AV further and use Lemma 77 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. A slight complication is that in Vampire's AVATAR, A-clauses $C \leftarrow \{[C]\}$ are generated on a per-need basis when switching model. This is not a serious issue because we can imagine that the A-clauses were there all along in the $\mathcal{Q}$ set.

Even the concrete criterion offered by Lemma 77 refers, in its condition 9, to limit superiors and limit points. Some architectures will satisfy it by their very design. For AVATAR, an easy way to meet the condition is to bound the number of times each A-formula can be locked. Once that number has been reached, the A-formula can no longer be locked. An alternative, suggested by a reviewer, is to disable all splitting after the prover has run for a specified time.

7 Application to Other Architectures

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

7.1 Splitting without Backtracking

Before the invention of AVATAR, Riazanov and Voronkov [25] 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 clause $C \vee D$ with $C \vee \mathsf{p}$ 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 consistency-preserving. They do not show refutational completeness, but this is obvious since the rule is a simplification.

Riazanov and Voronkov also extend 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 $O_{\mathbb{P}}$, since it is an instance of the same calculus.

The calculus $O_{\mathbb{P}}$ is closely related to an instance of our framework. Let **F** be the set of Σ-clauses, with the empty clause as $\bot$. Let $O = (FInf, FRed)$, where *FInf* is the set of ordered resolution inferences on **F** with some selection function and *FRed* is the standard redundancy criterion [2, Sect. 4.2.2], and similarly $O_{\mathbb{P}} = (FPInf, FPRed)$. We use the notion of entailment from Example 1 for the base relations $\models$ and $\not\approx$ for both calculi. We take $\mathbf{V} = \mathbb{P}$ for defining **AF**. The properties (D1)–(D6) and (R1)–(R7) are verified for $\models$ and *FRed*, respectively. This gives us a splitting calculus $LA = (SInf, SRed)$, 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:

$$\frac{C \vee D \leftarrow A}{C \leftarrow A \cup \{a\} \quad D \leftarrow A \cup \{\neg a\}} \text{ BINSPLIT}$$

with the side conditions that $a \in asn(C)$ and $C \vee D$ is splittable into C, D. By Theorem 21, 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, except that it is based on superposition instead of ordered resolution. It performs branch-specific simplifications (a special case of subsumption demodulation [15]), which is supported by our locking mechanism. A SAT solver is used to detect propositional unsatisfiability (correponding to our UNSAT rule) and to eliminate assertions that are implied at the SAT solver's top level (corresponding to our TRIM rule).

The calculi $O_{\mathbb{P}}$ and LA are very similar but not identical. $O_{\mathbb{P}}$ 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

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

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 $FPRed_I$, 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 $SRed_I$, because the assertions are simply truncated by the projection operator $(\ )_{\partial}$ and 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. Correspondingly, on satisfiable problems, LA 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 $O_{\mathbb{P}}$ counterpart is subject to an inference between $\neg \mathsf{a} \vee \mathsf{b}$ and $\neg \mathsf{b}$.

As positive results, we will show that $O_\mathbb{P}$ and LA share the same notion of entailment and $O_\mathbb{P}$'s redundancy criterion is stronger than LA's, yet saturation w.r.t. LA guarantees saturation w.r.t. $O_\mathbb{P}$, 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: $\lfloor C \vee L_1 \vee \cdots \vee L_n \rfloor = \lfloor C \leftarrow \{\neg L_1, \ldots, \neg L_n\} \rfloor = C$. Moreover, let $\mathscr{G}$ denote the function that returns all ground instances of a clause, clause set, or inference according to Σ, which is assumed to contain at least one constant.

Lemma 78 *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 $\mathcal{K} \cup \mathcal{J}$. $\qquad\square$

Lemma 79 *Given an inference ι over $\Sigma_\mathbb{P}$-clauses, if $\alpha(\iota)$ is a BASE inference, then $\iota \in FPInf$.*

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 $\iota \in FPInf$. $\qquad\square$

Lemma 80 (1) *Given a $\Sigma_\mathbb{P}$-clause C, if $\alpha(C) \in SRed_\mathrm{F}(\alpha(N))$, then $C \in FPRed_\mathrm{F}(N)$.* (2) *Given an inference ι over $\Sigma_\mathbb{P}$-clauses, if $\alpha(\iota) \in SRed_\mathrm{I}(\alpha(N))$, then $\iota \in FPRed_\mathrm{I}(N)$.*

Proof For (2), let $\iota' \in \mathscr{G}(\iota)$ and let $C_n, \ldots, C_1$ be ι''s premises and C_0 be its conclusion. Let $\mathscr{E} = \{C \leftarrow A \in \alpha(\mathscr{G}(N)) \mid C \prec \lfloor C_1 \rfloor\}$ and $\mathscr{F} = \{C \in N \mid \mathscr{G}(C) \prec C_1\}$. By the definition of standard redundancy, assuming that $\{\alpha(C_n), \ldots, \alpha(C_2)\} \cup \mathscr{E} \models \{\alpha(C_0)\}$ we need to show that $\{C_n, \ldots, C_2\} \cup \mathscr{F} \models \{C_0\}$. By Lemma 78, this amounts to showing that $\{\alpha(C_n), \ldots, \alpha(C_2)\} \cup \alpha(\mathscr{F}) \models \{\alpha(C_0)\}$. By (D3), this follows from our assumption if $\mathscr{E} \subseteq \alpha(\mathscr{F})$. 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 $C_1 = C_0 = C$. $\qquad\square$

Lemma 81 (**Saturation**) *Let N be a set of $\Sigma_\mathbb{P}$-clauses. If N is saturated w.r.t. $O_\mathbb{P}$, then $\alpha(N)$ is saturated w.r.t. LA.*

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

By Lemma 79, there exists an inference $(C_n, \ldots, C_1, C_0) \in FPInf$. Since N is saturated, the inference is redundant—i.e., $\{C_n\theta, \ldots, C_2\theta\} \cup \mathscr{F} \models \{C_0\theta\}$, where $\mathscr{F} = \{C \mid C \in \mathscr{G}(N) \text{ and } C \prec C_1\theta\}$. If $\alpha(\mathscr{F}) \subseteq \mathscr{E}$, we can invoke Lemma 78 to conclude. However, in the general case, we have only that $\alpha(\mathscr{F} \setminus \mathscr{F}_{\mathrm{eq}}) \subseteq \mathscr{E}$, where $\mathscr{F}_{\mathrm{eq}} = \{C \in \mathscr{F} \mid \lfloor C\rfloor = \lfloor C_1\theta\rfloor\}$, and thus there might be models of $\mathscr{E}$ that are models of $\mathscr{F} \setminus \mathscr{F}_{\mathrm{eq}}$ but not of $\mathscr{F}_{\mathrm{eq}}$. Fortunately, we can show that $\{C_n\theta, \ldots, C_2\theta\} \cup (\mathscr{F} \setminus \mathscr{F}_{\mathrm{eq}}) \models \{C_0\theta\}$. We proceed by removing from $\mathscr{F}$ each clause $D \in \mathscr{F}_{\mathrm{eq}}$ in turn and by showing that the entailment is preserved by each step. Finally, we invoke Lemma 78. A slight complication is that $\mathscr{F}_{\mathrm{eq}}$ may be infinite. However, by compactness, only a finite subset $\mathscr{F}'_{\mathrm{eq}} \subseteq \mathscr{F}_{\mathrm{eq}}$ is needed to have the desired entailment.

Let $D \in N$ be a clause that generalizes the ground, $\succ$-largest clause in $\mathscr{F}'_{\mathrm{eq}}$. Then there exists an inference $(C_n, \ldots, C_2, D, D_0) \in FPInf$ such that $\lfloor C_0\theta\rfloor \in \lfloor\mathscr{G}(D_0)\rfloor$ 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 $\{C_n\theta, \ldots, C_2\theta\} \cup (\mathscr{F} \setminus \{D\theta\})$. Since N is saturated, $\{C_n\theta, \ldots, C_2\theta\} \cup \{C \in \mathscr{G}(N) \mid C \prec D\theta\} \models \{D_0\theta\}$. Since we are proceeding from largest to smallest clause, we have $\{C \in \mathscr{G}(N) \mid C \prec D\theta\} \subseteq \mathscr{F} \setminus \{D\theta\}$, even if some clauses have been removed from $\mathscr{F}$ 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 $\mathcal{J} \models \{C_n\theta, \ldots, C_2\theta\} \cup \mathscr{F}$, 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. $\qquad\square$

7.2 Labeled Splitting

Labeled splitting, as originally described by Fietzke and Weidenbach [13] 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 [20]. It works on pairs $(\Psi, \mathcal{N})$, 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 $\mathsf{LS} = (SInf, SRed)$ induced by the resolution calculus $\mathsf{R} = (FInf, FRed)$, where $\models$ and $\approx$ are as in Example 1 and $\mathbf{V} = \bigcup_{i \in \mathbb{N}}\{\mathsf{l}_i, \mathsf{r}_i, \mathsf{s}_i\}$. A-clauses are essentially the same as labeled clauses.

Splits are identified by unique *split levels*. Given a split on $C \vee D$ with level k, the propositional variables $\mathsf{l}_k \in asn(C)$ and $\mathsf{r}_k \in asn(D)$ represent the left and right branches,

respectively. In practice, the prover would dynamically extend *fml* to ensure that $fml(\mathsf{l}_k) = C$ and $fml(\mathsf{r}_k) = D$.

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, which Fietzke and Weidenbach avoid. We solve this as follows. Let $\top = {\sim}\bot$. We introduce the propositional variable $\mathsf{s}_k \in asn(\top)$ 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 cases are 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:

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

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. Use DERIVE (via LIFT) to delete a redundant enabled A-clause.

3. Use LOCK to temporarily remove a locally redundant enabled A-clause.

4. Use DERIVE (via LIFT) to simplify an enabled A-clause. If the original A-clause is made redundant, delete it; otherwise, use LOCK to temporarily remove it. If an enabled $\bot \leftarrow A$ is derived with $A \subseteq \bigcup_i \{\mathsf{l}_i, \mathsf{r}_i\}$, apply SWITCH or STRONGUNSAT.

5. Apply SOFTSPLIT (via LIFT and DERIVE) with split level k on an A-clause $\mathscr{C}$. Then use SWITCH to enable the left branch and apply LOCK on $\mathscr{C}$ 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 $fml(\mathsf{s}_i) = \top$.

We illustrate the strategy on an example.

Example 82 Let $\mathscr{N}_0$ be the clause set

$$\{\neg \mathsf{q}(x), \quad \mathsf{p}(x) \vee \mathsf{q}(y), \quad \mathsf{r}(x) \vee \mathsf{s}(y), \quad \neg \mathsf{p}(x) \vee \mathsf{q}(y)\}$$

It is unsatisfiable due to the first, second, and fourth clauses. Let $\mathscr{J}_0 = \neg \mathbf{V}$ be the initial model. The first disjunction is split into $\mathsf{p}(x) \leftarrow \{\mathsf{l}_0\}$, $\mathsf{q}(y) \leftarrow \{\mathsf{r}_0\}$ and $\bot \leftarrow \{\neg \mathsf{l}_0, \neg \mathsf{r}_0, \mathsf{s}_0\}$ by SOFTSPLIT. Then a SWITCH transition replaces $\mathscr{J}_0$ with $(\mathscr{J}_0 \setminus \{\neg \mathsf{l}_0, \neg \mathsf{s}_0\}) \cup \{\mathsf{l}_0, \mathsf{s}_0\}$. This

enables the A-clause $p(x) \leftarrow \{l_0\}$. Then LOCK removes $p(x) \lor q(y)$ from $\mathcal{N}_0$ for as long as s_0 is enabled. Splitting the other two disjunctions leads to the state $(\mathcal{J}', \mathcal{N}', \mathcal{L}')$, where

$$\mathcal{J}' = (\mathcal{J}_0 \setminus \bigcup_{i=0}^{2} \{\neg l_i, \neg s_i\}) \cup \bigcup_{i=0}^{2} \{l_i, s_i\}$$

$$\mathcal{N}' = \bigcup_{i=0}^{2} \{\bot \leftarrow \{\neg l_i, \neg r_i, s_i\}\} \cup$$
$$\{\neg q(x), \quad p(x) \leftarrow \{l_0\}, \quad r(x) \leftarrow \{l_1\}, \quad \neg p(x) \leftarrow \{l_2\}$$
$$q(y) \leftarrow \{r_0\}, \quad s(y) \leftarrow \{r_1\}, \quad q(y) \leftarrow \{r_2\}\}$$

$$\mathcal{L}' = \{(\{s_0\}, p(x) \lor q(x)), \quad (\{s_1\}, r(x) \lor s(y)), \quad (\{s_2\}, \neg p(x) \lor q(y))\}$$

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

The first backtracking step happens after a BASE inference produces $\bot \leftarrow \{l_0, l_2\}$ from $p(x) \leftarrow \{l_0\}$ and $\neg p(x) \leftarrow \{l_2\}$. The SWITCH disables s_1, because this split was not useful in closing the branch, and it moves from branch l_2 to r_2. The new model disables $\neg p(x) \leftarrow \{l_2\}$, enables $q(y) \leftarrow \{r_2\}$, and unlocks $r(x) \lor s(y)$.

The second backtracking step happens after $\bot \leftarrow \{r_2\}$ is derived from $\neg q(x)$ and $q(y) \leftarrow \{r_2\}$. Since both branches of the split s_2 have now been closed, the SWITCH rule is invoked, producing the model $(\mathcal{J}_0 \setminus \{\neg r_0, \neg s_0\}) \cup \{r_0, s_0\}$. This unlocks $\neg p(x) \lor q(y)$, and now only $q(y) \leftarrow \{r_0\}$ is enabled in addition to the unlocked input clauses. The r_0 branch is immediately closed by the generation of $\bot \leftarrow \{r_0\}$ from $q(y) \leftarrow \{r_0\}$ and $\neg q(x)$.

Now, the resulting A-clause set contains the propositional clauses $\bot \leftarrow \{l_0, l_2\}$, $\bot \leftarrow \{r_2\}$, $\bot \leftarrow \{r_0\}$ derived by inferences as well as $\bot \leftarrow \{s_0, \neg l_0, \neg r_0\}$ and $\bot \leftarrow \{s_2, \neg l_2, \neg r_2\}$ produced by SPLIT. Clearly, it entails $\bot \leftarrow \{s_0, s_2\}$. Since $fml(s_0) = fml(s_2) = \sim \bot$, at this point STRONGUNSAT derives $\bot$.

By following the strategy presented above, LS closely simulates the original calculus, in the sense that 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, inconsequential difference lies in the backtracking: Labeled splitting can move to a branch where $\bot$ is enabled, whereas our SWITCH rule requires that all propositional clauses are satisfied.

What about fairness? The above strategy helps achieve fairness by ensuring that there exists at most one limit point. It also uses locks in a well-behaved way. This means we can considerably simplify the notion of fairness for $\Longrightarrow_L$-derivations 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 demand that every persistent inference in it be made redundant, exactly as we do for $\Longrightarrow_L$-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 83 Alternating age-based and unrestricted heuristic selection is incomplete for labeled splitting just as it is for AVATAR (Example 58). To see why, start with the clause set

$$\{p(a,y), \quad \neg p(x,y) \vee p(s(x),y), \quad q(a), \quad r(x) \vee q(y) \vee \neg p(x,y), \quad s(x) \vee \neg q(x),$$
$$\neg s(x)\}$$

and always select the negative literal if there is one. The prover begins by deriving $p(s(a),y)$ and $r(a) \vee q(y)$ using the age-based heuristic. Then it heuristically selects $r(a) \vee q(y)$ and splits it. In the left branch, where $q(y)$ is enabled, $q(a)$ is locally redundant and locked. Before age-based selection allows the prover to derive $\bot$ from the clauses $s(x) \vee \neg q(x), q(y)$, and $\neg s(x)$, it will also have derived $p(s(s(a)),y)$ and $r(s(a)) \vee q(y)$. When the prover switches back to the right branch, it can heuristically select the newly derived disjunction and split it.

This process can be repeated to give rise to infinitely many splittable clauses of the form $r(s^i(a)) \vee q(y)$. In this way, no inferences are ever performed in the rightmost branch, only splits. The clause $q(a)$, 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

SMT solvers based on DPLL(T) [20] 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 overall solver is a decision procedure for the union of the theories. Some SMT solvers, including cvc5 [3], veriT [8], and Z3 [18], also support quantified formulas via instantiation at the expense of decidability.

Complete instantiation strategies have been developed for various fragments of first-order logic [14, 23, 24]. In particular, enumerative quantifier instantiation [23] 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 **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 [2, Sect. 4.2.2]. It can be used to split disjunctions destructively and to simplify formulas. We define the "theories with quantifiers" calculus TQ = (*FInf*, *FRed*). For the consequence relations $\models$ and $\approx$, we use entailment in the supported theories including quantifiers.

Some theories such as linear integer arithmetic are not compact and thus cannot directly be used for the consequence relation. Instead, we define $M \models_{LIA} N$ to be true if and only if there exist finite sets $M' \subseteq M$ and $N' \subseteq N$ such that $\bigwedge M' \rightarrow \bigvee N'$ is valid modulo linear integer arithmetic. For finite sets, this relation coincides with noncompact entailment: If M is finite, then $M \models_{LIA} \bot$ if and only if M is inconsistent modulo linear integer arithmetic. Both completeness and soundness of a concrete prover are statements about the finite set of input formulas, so using a compactified version of the consequence relation is purely an implementation detail and poses no restriction.

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 (ground) 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 84 (Dynamic completeness) *Assume TQ is statically complete for subunit sets. Let $(\mathfrak{J}_i, \mathscr{N}_i)_i$ be a fair $\Longrightarrow_{\mathsf{MG}}$-derivation based on TQ. If $\mathscr{N}_0 \models \{\bot\}$ and $\mathscr{N}_\infty$ contains only subunits, then $\bot \in \mathscr{N}_j$ for some j.*

Proof The proof is analogous to that of Theorem 28. Because we only have conditional static completeness of (*FInf*, *FRed*), we need the assumption that $\mathscr{N}_\infty$ 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 $SRed_{\mathrm{I}}$. Given $\mathsf{a} \approx \mathsf{b} \leftarrow \{\mathsf{v}_0\}$, $\mathsf{b} \approx c \leftarrow \{\mathsf{v}_1\}$, $\mathsf{a} \approx \mathsf{d} \leftarrow \{\mathsf{v}_2\}$, $c \approx \mathsf{d} \leftarrow \{\mathsf{v}_3\}$, and $\mathsf{a} \not\approx c \leftarrow \{\mathsf{v}_4\}$, an SMT solver 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 85 Consider the initial clause set

$$\{\forall x \,(x \leq 0 \vee \mathsf{a} > x), \qquad \mathsf{a} \approx 0, \qquad \mathsf{a} + 3 < 2\}$$

Eagerly applying quantifier instantiation, we get the instances

$$i \leq 0 \leftarrow \{[i \leq 0]\}, \quad \mathsf{a} > i \leftarrow \{[\mathsf{a} > i]\}, \quad \bot \leftarrow \{\neg[i \leq 0], \neg[\mathsf{a} > 1]\}$$

for every $i \in \mathbb{N}$. The solver then starts in a model where each $[\mathsf{a} > i]$ is 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 $\bot \leftarrow \{[\mathsf{a} > 1]\}$.

Iterating this process, we see that all conflicts are of the form $[\mathsf{a} > i]$ for some i. However, at the limit point—where $[\mathsf{a} > i]$ is false for every i—none of these conflicts is enabled. The only conflict which exists at the limit point is between $\mathsf{a} \approx 0$ and $\mathsf{a} + 3 < 2$, and the solver never finds it because it detects 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 analysis as in the proof of Lemma 77 works to establish fairness for this strategy.

First consider the case where quantifier instantiation is invoked infinitely often. Then there exists an infinite subsequence $(\mathfrak{J}'_j, \mathscr{N}'_j)_j$ of states such that (1) $(\mathfrak{J}'_j)_j$ converges to a limit point, and (2) no $\mathscr{N}'_j$ has a theory conflict. To prove the $\Longrightarrow_{\mathsf{MG}}$-derivation fair, we

need to show that $\iota \in \mathit{FInf}((\mathscr{N}_\infty)_\jmath)$ implies $\iota \in \mathit{FRed}_I((\mathscr{N}_i)_\jmath)$ for every ι. If ι is a theory conflict or clausification inference, then its finitely many premises are in $\mathscr{N}_j'$ for some j, contradicting the strategy. Otherwise, ι is a quantifier instantiation. Here, it suffices to ensure that A-formulas that are enabled infinitely often at a quantifier instantiation step are also fully instantiated. (Just as with AV provers, it is possible that not all limit points are saturated.)

Otherwise, quantifier instantiation is only invoked finitely often—either because every encountered model had a theory conflict, or because there was nothing to instantiate. Here, it suffices to assume that clausification is a strongly finitary simplification bound (which means that a formula can only be clausified in a finite number of ways). Under this assumption only finitely many base formulas will be derived; this implies that only a finite number of models will be considered. The last model will then be saturated due to the strategy.

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

Example 86 Consider an SMT solver equipped with the theory of uninterpreted functions and linear arithmetic. Let

$$\{\forall x\,(\mathsf{f}(x) \approx 0 \vee \mathsf{g}(x) \approx 0), \qquad \mathsf{f}(1) \approx 1, \qquad \mathsf{g}(1) \approx 1\}$$

be the initial clause set. The SMT solver first considers the propositional model $\jmath_0 = \neg\mathbf{V}$. There is no theory conflict, so the solver uses quantifier instantiation and clausification to derive $\mathsf{f}(0) \approx 0 \leftarrow \{\mathsf{v}_0\}$, $\mathsf{g}(0) \approx 0 \leftarrow \{\mathsf{v}_1\}$, and $\perp \leftarrow \{\neg\mathsf{v}_0, \neg\mathsf{v}_1\}$. We have $\jmath_0 \not\models \perp \leftarrow \{\neg\mathsf{v}_0, \neg\mathsf{v}_1\}$, so the solver switches to the model $\jmath_1 = (\jmath_0 \setminus \{\neg\mathsf{v}_0\}) \cup \{\mathsf{v}_0\}$. There is still no theory conflict, so it instantiates a quantifier again, producing the A-clauses $\mathsf{f}(1) \approx 0 \leftarrow \{\mathsf{v}_2\}$, $\mathsf{g}(1) \approx 0 \leftarrow \{\mathsf{v}_3\}$, and $\perp \leftarrow \{\neg\mathsf{v}_2, \neg\mathsf{v}_3\}$. The solver now switches to $\jmath_2 = (\jmath_1 \setminus \{\neg\mathsf{v}_2\}) \cup \{\mathsf{v}_2\}$. It derives a theory conflict $\perp \leftarrow \{\mathsf{v}_2\}$ and switches to $\jmath_3 = (\jmath_2 \setminus \{\neg\mathsf{v}_3\}) \cup \{\mathsf{v}_3\}$. For this model, there is also a conflict, $\perp \leftarrow \{\mathsf{v}_3\}$, and the solver terminates by applying UNSAT.

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 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. It seemed preferable 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 and prover heuristics 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 locking preserves completeness. As for the relationship 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, 10, 19].

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: A. Robinson, A. Voronkov (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: G. Gopalakrishnan, S. Qadeer (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: C. Benzmüller, G. Sutcliffe, R. Rojas (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: L. de Moura (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: R.A. Schmidt (ed.) CADE-22, *LNCS*, vol. 5663, pp. 35–50. Springer (2009)
7. Bonacina, M.P., Plaisted, D.A.: SGGS theorem proving: An exposition. In: S. Schulz, L. de Moura, B. Konev (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: R.A. Schmidt (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. Déharbe, D., Fontaine, P., Ranise, S., Ringeissen, C.: Decision procedures for the formal analysis of software. In: K. Barkaoui, A. Cavalcanti, A. Cerone (eds.) ICTAC 2006, *LNCS*, vol. 4281, pp. 366–370. Springer (2006)
11. Ebner, G., Blanchette, J., Tourret, S.: A unifying splitting framework. In: A. Platzer, G. Sutcliffe (eds.) CADE-28, *LNCS*, vol. 12699, pp. 344–360. Springer (2021)
12. 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)
13. Fietzke, A., Weidenbach, C.: Labelled splitting. Ann. Math. Artif. Intell. **55**(1–2), 3–34 (2009)
14. Ge, Y., de Moura, L.: Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In: A. Bouajjani, O. Maler (eds.) CAV 2009, *LNCS*, vol. 5643, pp. 306–320. Springer (2009)
15. Gleiss, B., Kovács, L., Rath, J.: Subsumption demodulation in first-order theorem proving. In: N. Peltier, V. Sofronie-Stokkermans (eds.) IJCAR 2020, Part I, *LNCS*, vol. 12166, pp. 297–315. Springer (2020)
16. Knuth, D.E., Bendix, P.B.: Simple word problems in universal algebras. In: J. Leech (ed.) Computational Problems in Abstract Algebra, pp. 263–297. Pergamon Press (1970)
17. McCune, W., Wos, L.: Otter—the CADE-13 competition incarnations. J. Autom. Reason. **18**(2), 211–220 (1997)

18. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: C.R. Ramakrishnan, J. Rehof (eds.) TACAS 2008, *LNCS*, vol. 4963, pp. 337–340. Springer (2008)

19. de Moura, L., Jovanović, D.: A model-constructing satisfiability calculus. In: R. Giacobazzi, J. Berdine, I. Mastroeni (eds.) VMCAI 2013, *LNCS*, vol. 7737, pp. 1–12. Springer (2013)

20. 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)

21. Nipkow, T., Klein, G.: Concrete Semantics: With Isabelle/HOL. Springer (2014)

22. Reger, G., Suda, M., Voronkov, A.: Playing with AVATAR. In: A.P. Felty, A. Middeldorp (eds.) CADE-25, *LNCS*, vol. 9195, pp. 399–415. Springer (2015)

23. Reynolds, A., Barbosa, H., Fontaine, P.: Revisiting enumerative instantiation. In: D. Beyer, M. Huisman (eds.) TACAS 2018, *LNCS*, vol. 10806, pp. 112–131. Springer (2018)

24. Reynolds, A., Tinelli, C., Goel, A., Krstić, S.: Finite model finding in SMT. In: N. Sharygina, H. Veith (eds.) CAV 2013, *LNCS*, vol. 8044, pp. 640–655. Springer (2013)

25. Riazanov, A., Voronkov, A.: Splitting without backtracking. In: B. Nebel (ed.) IJCAI 2001, pp. 611–617. Morgan Kaufmann (2001)

26. Robinson, J.A.: A machine-oriented logic based on the resolution principle. J. ACM **12**(1), 23–41 (1965)

27. Schlichtkrull, A., Blanchette, J.C., Traytel, D.: A verified prover based on ordered resolution. In: A. Mahboubi, M.O. Myreen (eds.) CPP 2019, pp. 152–165. ACM (2019)

28. Voronkov, A.: AVATAR: The architecture for first-order theorem provers. In: A. Biere, R. Bloem (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: N. Peltier, V. Sofronie-Stokkermans (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: A. Robinson, A. Voronkov (eds.) Handbook of Automated Reasoning, vol. II, pp. 1965–2013. Elsevier and MIT Press (2001)

32. Weidenbach, C., Gaede, B., Rock, G.: SPASS & FLOTTER version 0.42. In: M.A. McRobbie, J.K. Slaney (eds.) CADE-13, *Lecture Notes in Computer Science*, vol. 1104, pp. 141–145. Springer (1996)