A Unifying Splitting Framework

Gabriel Ebner, Jasmin Blanchette, Sophie Tourret

A Unifying Splitting Framework

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

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

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

1 Introduction

One of the great strengths of saturation calculi such as superposition [1] is that they avoid case distinctions. Derived clauses hold unconditionally, and the prover can stop as soon as it derives the empty clause, without having to backtrack. The drawback is that these calculi often generate long, unwieldy clauses that slow down the prover. A remedy is to partition the search space by splitting a multiple-literal clause $C_1 \vee \cdots \vee C_n$ into variable-disjoint subclauses C_i. Splitting approaches include splitting with backtracking [24], splitting without backtracking [20], labeled splitting [10], and AVATAR [22].

The SAT-based AVATAR architecture is of particular interest because it is so successful. Voronkov reported that an AVATAR-enabled Vampire could solve 421 TPTP [21] problems that had never been solved before by any system [22, Sect. 9], a mind-boggling number. AVATAR works well in combination with the superposition calculus because it combines superposition's strong equality reasoning with the SAT solver's strong clausal reasoning. It is also appealing theoretically, because it gracefully generalizes traditional saturation provers and yet degenerates to a SAT solver if the problem is propositional.

Example 1. To illustrate the approach, we follow the key steps of an AVATAR-enabled resolution prover on the initial clause set containing $\neg\mathsf{p}(\mathsf{a})$, $\neg\mathsf{q}(z, z)$, and $\mathsf{p}(x) \vee \mathsf{q}(y, \mathsf{b})$. The disjunction can be split into $\mathsf{p}(x) \leftarrow \{[\mathsf{p}(x)]\}$ and $\mathsf{q}(y, \mathsf{b}) \leftarrow \{[\mathsf{q}(y, \mathsf{b})]\}$, where $C \leftarrow \{[C]\}$ indicates that the clause C is enabled only in models in which the associated propositional variable $[C]$ is true. A SAT solver is then

run to choose a model $\mathcal{J}$ of $[\mathsf{p}(x)] \vee [\mathsf{q}(y, \mathsf{b})]$. Suppose $\mathcal{J}$ makes $[\mathsf{p}(x)]$ true and $[\mathsf{q}(y, \mathsf{b})]$ false. Then resolving $\mathsf{p}(x) \leftarrow \{[\mathsf{p}(x)]\}$ with $\neg\mathsf{p}(\mathsf{a})$ produces $\bot \leftarrow \{[\mathsf{p}(x)]\}$, which closes the branch. Next, the SAT solver makes the right disjunct true, and resolving $\mathsf{q}(y, \mathsf{b}) \leftarrow \{[\mathsf{q}(y, \mathsf{b})]\}$ with $\neg\mathsf{q}(z, z)$ yields $\bot \leftarrow \{[\mathsf{q}(y, \mathsf{b})]\}$. The SAT solver then reports "unsatisfiable," concluding the refutation.

What about refutational completeness? Far from being a purely theoretical concern, establishing completeness—or finding counterexamples—could yield insights 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 [23], based on Bachmair and Ganzinger [2]. It covers a wide array of techniques, but "the main missing piece of the framework is a generic treatment of clause splitting" [23, 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.

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

From a base calculus, we derive a *splitting calculus* (Sect. 3). This extends the base calculus with splitting and inherits the base's completeness. It works on A-clauses or A-formulas $C \leftarrow A$, where A is a set of propositional literals.

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

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

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

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

A technical report [8] is available with the proofs, several counterexamples, and further details. A formalization using Isabelle/HOL [16] is underway.

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. [23]. A-formulas generalize Voronkov's A-clauses [22].

Formulas. A set $\mathbf{F}$ of *formulas* is a set that contains a distinguished element $\bot$ denoting falsehood. A *consequence relation* $\models \subseteq (\mathcal{P}(\mathbf{F}))^2$ has the following properties for all $M, N, P, Q \subseteq \mathbf{F}$ and $C, D \in \mathbf{F}$: (D1) $\{\bot\} \models \emptyset$; (D2) $\{C\} \models \{C\}$; (D3) if $M \subseteq N$ and $P \subseteq Q$, then $M \models P$ implies $N \models Q$; (D4) if $M \models P$ and $N \models Q \cup \{C\}$ for every $C \in M$ and $N \cup \{D\} \models Q$ for every $D \in P$, then $N \models Q$. The intended meaning of $M \models N$ is $\bigwedge M \longrightarrow \bigvee N$. From $\models$, we can easily derive a relation understood as $\bigwedge M \longrightarrow \bigwedge N$, as required by 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 have $M \models N$ if and only if $\{C \in \mathbf{F} \mid C \in M\} \cup \{C \in \mathbf{F} \mid \sim C \in N\} \models \{C \in \mathbf{F} \mid \sim C \in M\} \cup \{C \in \mathbf{F} \mid C \in N\}$.

Following the saturation framework [23, p. 318], we distinguish between the consequence relation $\models$ used for stating refutational completeness and a possibly stronger consequence relation $\mathrel{\approx\!\!\models}$ for soundness. We require that $\mathrel{\approx\!\!\models}$ is compact.

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

Calculi and Derivations. A refutational calculus $(\mathit{Inf}, \mathit{Red})$ combines a set of inferences Inf and a redundancy criterion Red. We refer to Waldmann et al. [23] for the precise definitions. Recall in particular that $\mathit{Inf}(N)$ is the set of inferences from N, $\mathit{Inf}(N, M) = \mathit{Inf}(N \cup M) \setminus \mathit{Inf}(N \setminus M)$, N is *saturated* w.r.t. Inf and Red_I if $\mathit{Inf}(N) \subseteq \mathit{Red}_\mathrm{I}(N)$, and $(\mathit{Inf}, \mathit{Red})$ is *statically* (*refutationally*) *complete* (w.r.t. $\models$) if $\bot \in N$ for every $N \models \{\bot\}$ saturated w.r.t. Inf and Red_I.

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 $(N_i)_i$ over $\mathcal{P}(\mathbf{F})$ is *weakly fair* w.r.t. Inf and Red_I if $\mathit{Inf}(N_\infty) \subseteq \bigcup_i \mathit{Red}_\mathrm{I}(N_i)$ and *strongly fair* if $(\mathit{Inf}(N_i))^\infty \subseteq \bigcup_i \mathit{Red}_\mathrm{I}(N_i)$. Given a relation $\rhd$, a $\rhd$-*derivation* is an infinite sequence such that $x_i \rhd x_{i+1}$ for every i. Finite runs can be extended to derivations via stuttering.

Let $\rhd_{\mathit{Red}_\mathrm{F}} \subseteq (\mathcal{P}(\mathbf{F}))^2$ be the relation such that $M \rhd_{\mathit{Red}_\mathrm{F}} N$ if and only if $M \setminus N \subseteq \mathit{Red}_\mathrm{F}(N)$. The calculus $(\mathit{Inf}, \mathit{Red})$ is *dynamically* (*refutationally*) *complete* (w.r.t. $\models$) if for every $\rhd_{\mathit{Red}_\mathrm{F}}$-derivation $(N_i)_i$ that is weakly fair w.r.t. Inf and Red_I and such that $N_0 \models \{\bot\}$, we have $\bot \in N_i$ for some i.

A-Formulas. We fix throughout a countable set $\mathbf{V}$ of *propositional variables* $\mathsf{v}_0, \mathsf{v}_1, \ldots$. For each $\mathsf{v} \in \mathbf{V}$, let $\neg\mathsf{v} \in \neg\mathbf{V}$ denote its negation, with $\neg\neg\mathsf{v} = \mathsf{v}$. We assume that a formula $\mathit{fml}(\mathsf{v}) \in \mathbf{F}$ is associated with each $\mathsf{v} \in \mathbf{V}$. Intuitively, v

approximates $\mathit{fml}(\mathsf{v})$ at the propositional level. This definition is extended so that $\mathit{fml}(\neg\mathsf{v}) = {\sim}\mathit{fml}(\mathsf{v})$. An *assertion* $a \in \mathbf{A} = \mathbf{V} \cup \neg\mathbf{V}$ is either a propositional variable v or its negation $\neg\mathsf{v}$. Given a formula $C \in \mathbf{F}_{\sim}$, let $\mathit{asn}(C)$ denote the set of assertions $a \in \mathbf{A}$ such that $\{\mathit{fml}(a)\} \mathrel{\rotatebox{180}{$\not\approx$}} \{C\}$ and $\{C\} \mathrel{\rotatebox{180}{$\not\approx$}} \{\mathit{fml}(a)\}$.

A *propositional interpretation* $\mathcal{J} \subseteq \mathbf{A}$ is a set such that for every $\mathsf{v} \in \mathbf{V}$, exactly one of $\mathsf{v} \in \mathcal{J}$ and $\neg\mathsf{v} \in \mathcal{J}$ holds. We reserve the letter $\mathcal{J}$ for interpretations, and define $\mathit{fml}(\mathcal{J}) = \{\mathit{fml}(a) \mid a \in \mathcal{J}\}$.

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

We say that $C \leftarrow A \in \mathbf{AF}$ is *enabled* in $\mathcal{J}$ if $A \subseteq \mathcal{J}$. A set of A-formulas is *enabled* in $\mathcal{J}$ if all of its members are enabled in $\mathcal{J}$. The *enabled projection* $\mathcal{N}_\mathcal{J} \subseteq \lfloor \mathcal{N} \rfloor$ consists of the projections $\lfloor \mathcal{C} \rfloor$ of all A-formulas $\mathcal{C}$ enabled in $\mathcal{J}$. Analogously, the *enabled projection* $\mathit{Inf}_\mathcal{J} \subseteq \lfloor \mathit{Inf} \rfloor$ of a set Inf of $\mathbf{AF}$-inferences consists of the projections $\lfloor \iota \rfloor$ of all inferences $\iota \in \mathit{Inf}$ whose premises are all enabled in $\mathcal{J}$.

A propositional interpretation $\mathcal{J}$ is a *propositional model* of $\mathcal{N}_\bot$, written $\mathcal{J} \models \mathcal{N}_\bot$, if $\bot \notin (\mathcal{N}_\bot)_\mathcal{J}$. Moreover, we write $\mathcal{J} \mathrel{\rotatebox{180}{$\not\approx$}} \mathcal{N}_\bot$ if $\bot \notin (\mathcal{N}_\bot)_\mathcal{J}$ or $\mathit{fml}(\mathcal{J}) \mathrel{\rotatebox{180}{$\not\approx$}} \{\bot\}$. A set $\mathcal{N}_\bot$ is *propositionally satisfiable* if there exists an interpretation $\mathcal{J}$ such that $\mathcal{J} \models \mathcal{N}_\bot$. In contrast to consequence relations, propositional modelhood $\models$ interprets the set $\mathcal{N}_\bot$ conjunctively: $\mathcal{J} \models \mathcal{N}_\bot$ is understood as $\mathcal{J} \models \bigwedge \mathcal{N}_\bot$.

Finally, we lift $\models$ and $\mathrel{\rotatebox{180}{$\not\approx$}}$ from $\mathcal{P}(\mathbf{F})$ to $\mathcal{P}(\mathbf{AF})$: $\mathcal{M} \models \mathcal{N}$ if and only if $\mathcal{M}_\mathcal{J} \models \lfloor \mathcal{N} \rfloor$ for every $\mathcal{J}$ in which $\mathcal{N}$ is enabled, and $\mathcal{M} \mathrel{\rotatebox{180}{$\not\approx$}} \mathcal{N}$ if and only if $\mathit{fml}(\mathcal{J}) \cup \mathcal{M}_\mathcal{J} \mathrel{\rotatebox{180}{$\not\approx$}} \lfloor \mathcal{N} \rfloor$ for every $\mathcal{J}$ in which $\mathcal{N}$ is enabled.

Example 3. In the original AVATAR [22], 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 $\mathit{fml}(\mathsf{v}) = C$ for some C such that $[C] = \mathsf{v}$, for every v.

3 Splitting Calculi

Let $\mathbf{F}$ be a set of base formulas equipped with $\bot$, $\models$, and $\mathrel{\rotatebox{180}{$\not\approx$}}$. The relation $\mathrel{\rotatebox{180}{$\not\approx$}}$ is assumed to be nontrivial: (D5) $\emptyset \mathrel{\rotatebox{180}{$\not\approx$}} \emptyset$. Let $\mathbf{A}$ be a set of assertions over $\mathbf{V}$ and $\mathbf{AF}$ be the set of A-formulas over $\mathbf{F}$ and $\mathbf{A}$. Let $(\mathit{FInf}, \mathit{FRed})$ be a base calculus for $\mathbf{F}$, where FRed is a redundancy criterion that additionally satisfies (1) an inference is FRed_I-redundant if one of its premises is FRed_F-redundant; (2) $\bot \notin \mathit{FRed}_\mathrm{F}(N)$ for every $N \subseteq \mathbf{F}$; and (3) $C \in \mathit{FRed}_\mathrm{F}(\{\bot\})$ for every $C \neq \bot$. These requirements can easily be met by a well-designed redundancy criterion [1, Sect. 4.3].

Below, we will define the *splitting calculus* induced by the base calculus. We will see that it not only is statically and dynamically complete w.r.t. $\models$, but also meets weaker, "local completeness" criteria that capture model switching.

The Inference Rules. We start with the mandatory inference rules.

Definition 4. The *splitting inference system SInf* consists of all instances of

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

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

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

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

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

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

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

The three rules identified by double bars are simplifications; they replace their premises with their conclusions in the current A-formula set. The premises' removal is justified by $\mathit{SRed}_{\mathrm{F}}$, defined below. Also note that Base preserves the soundness of *FInf* w.r.t. $\not\approx$ and that the other rules are sound w.r.t. $\not\models$.

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

Collect and Trim do some garbage collection. StrongUnsat is a variant of Unsat that uses $\not\approx$ instead of $\not\models$. It might correspond to invoking an SMT solver [3] ($\not\approx$) with a time limit, falling back on a SAT solver ($\not\models$). Approx can be used to make any derived A-formula visible to $\not\approx$. Tauto allows communication in the other direction, from the SAT solver to the calculus.

Example 5. Suppose the base calculus is first-order resolution [2] and the initial clauses are $\neg\mathsf{p}(\mathsf{a})$, $\neg\mathsf{q}(z, z)$, and $\mathsf{p}(x) \vee \mathsf{q}(y, \mathsf{b})$, as in Example 1. Split replaces the last clause by $\bot \leftarrow \{\neg\mathsf{v}_0, \neg\mathsf{v}_1\}$, $\mathsf{p}(x) \leftarrow \{\mathsf{v}_0\}$, and $\mathsf{q}(y, \mathsf{b}) \leftarrow \{\mathsf{v}_1\}$. Two Base inferences then generate $\bot \leftarrow \{\mathsf{v}_0\}$ and $\bot \leftarrow \{\mathsf{v}_1\}$. Finally, Unsat generates $\bot$.

The Redundancy Criterion. Next, we lift the base redundancy criterion.

Definition 6. The *splitting redundancy criterion* $SRed = (SRed_{\mathrm{I}}, SRed_{\mathrm{F}})$ is specified as follows. An A-formula $C \leftarrow A \in \mathbf{AF}$ is redundant w.r.t. $\mathcal{N}$, written $C \leftarrow A \in SRed_{\mathrm{F}}(\mathcal{N})$, if (1) $C \in FRed_{\mathrm{F}}(\mathcal{N}_{\mathcal{J}})$ for every propositional interpretation $\mathcal{J} \supseteq A$ or (2) there exists an A-formula $C \leftarrow B \in \mathcal{N}$ with $B \subset A$. An inference $\iota \in SInf$ is redundant w.r.t. $\mathcal{N}$, written $\iota \in SRed_{\mathrm{I}}(\mathcal{N})$, if (1) ι is a BASE inference and $\{\iota\}_{\mathcal{J}} \subseteq FRed_{\mathrm{I}}(\mathcal{N}_{\mathcal{J}})$ for every $\mathcal{J}$ or (2) ι is an UNSAT inference and $\bot \in \mathcal{N}$.

SRed qualifies as a redundancy criterion. It can justify the deletion of A-formulas that are propositionally tautological. 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, 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 $B \supset A$.

Local Saturation. It is not difficult to show that if $(FInf, FRed)$ is statically complete, then $(SInf, SRed)$ is statically and hence dynamically complete. However, this result fails to capture a key aspect of most splitting architectures. Since $\rhd_{SRed_{\mathrm{F}}}$-derivations have no notion of current split branch or model $\mathcal{J}$, they must also perform disabled inferences. To respect enabledness, we need a weaker notion of saturation. If an A-formula set is consistent, it should suffice to saturate w.r.t. a single propositional model. In other words, if no A-formula $\bot \leftarrow A \subseteq \mathcal{J}$ is derivable for some model $\mathcal{J} \models \mathcal{N}_{\bot}$, the prover should be allowed to give a verdict of "consistent." We will call such model-specific saturations *local*.

Definition 7. A set $\mathcal{N} \subseteq \mathbf{AF}$ is *locally saturated* w.r.t. *SInf* and $SRed_{\mathrm{I}}$ if either $\bot \in \mathcal{N}$ or there exists $\mathcal{J} \models \mathcal{N}_{\bot}$ such that $\mathcal{N}_{\mathcal{J}}$ is saturated w.r.t. *FInf* and $FRed_{\mathrm{I}}$.

Theorem 8 (Strong static completeness). *Assume $(FInf, FRed)$ is statically complete. Given a set $\mathcal{N} \subseteq \mathbf{AF}$ that is locally saturated w.r.t. SInf and $SRed_{\mathrm{I}}$ and such that $\mathcal{N} \models \{\bot\}$, we have $\bot \in \mathcal{N}$.*

Example 9. Consider the A-clause set $\{\bot \leftarrow \{\neg[\mathsf{p}(x)], \neg[\mathsf{q}(y)]\}, \mathsf{p}(x) \leftarrow \{[\mathsf{p}(x)]\}, \mathsf{q}(y) \leftarrow \{[\mathsf{q}(y)]\}, \neg\mathsf{q}(\mathsf{a})\}$ expressed using AVATAR conventions. It is not saturated for resolution, because the conclusion $\bot \leftarrow \{[\mathsf{q}(y)]\}$ of resolving the last two A-clauses is missing, but it is locally saturated with $\mathcal{J} \supseteq \{[\mathsf{p}(x)], \neg[\mathsf{q}(y)]\}$.

Definition 10. A sequence $(\mathcal{N}_i)_i$ of sets of A-formulas is *locally fair* w.r.t. *SInf* and $SRed_{\mathrm{I}}$ if either $\bot \in \mathcal{N}_i$ for some i or there exists $\mathcal{J} \models (\mathcal{N}_{\infty})_{\bot}$ such that $FInf((\mathcal{N}_{\infty})_{\mathcal{J}}) \subseteq \bigcup_i FRed_{\mathrm{I}}((\mathcal{N}_i)_{\mathcal{J}})$.

Theorem 11 (Strong dynamic completeness). *Assume $(FInf, FRed)$ is statically complete. Given an $\rhd_{SRed_{\mathrm{F}}}$-derivation $(\mathcal{N}_i)_i$ that is locally fair w.r.t. SInf and $SRed_{\mathrm{I}}$ and such that $\mathcal{N}_0 \models \{\bot\}$, we have $\bot \in \mathcal{N}_i$ for some i.*

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 will correspond to $\rhd_{SRed_{\mathrm{F}}}$-derivations, and their fairness criteria will imply local fairness. Consequently, by Theorem 11, they will all be complete.

4 Model-Guided Provers

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

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

Derive	$(\mathcal{J}, \mathcal{N} \uplus \mathcal{M}) \Longrightarrow_{\mathsf{MG}} (\mathcal{J}, \mathcal{N} \uplus \mathcal{M}')$	if $\mathcal{M} \subseteq \mathit{SRed}_{\mathrm{F}}(\mathcal{N} \uplus \mathcal{M}')$
Switch	$(\mathcal{J}, \mathcal{N}) \Longrightarrow_{\mathsf{MG}} (\mathcal{J}', \mathcal{N})$	if $\mathcal{J}' \models \mathcal{N}_\bot$
StrongUnsat	$(\mathcal{J}, \mathcal{N}) \Longrightarrow_{\mathsf{MG}} (\mathcal{J}, \mathcal{N} \cup \{\bot\})$	if $\mathcal{N}_\bot \mathrel{\mid\!\approx} \{\bot\}$

From an $\Longrightarrow_{\mathsf{MG}}$-derivation, we obtain an $\rhd_{SRed_{\mathrm{F}}}$-derivation by simply erasing the $\mathcal{J}$ components. The Derive rule can add new A-formulas and delete redundant A-formulas. $\mathcal{J}$ should be a model of $\mathcal{N}_\bot$ most of the time; when it is not, Switch can be used to switch model or StrongUnsat to finish the refutation.

Example 12. Let us revisit Example 5. Initially, let $\mathcal{J}_0 = \{\neg \mathsf{v}_0, \neg \mathsf{v}_1\}$. After the split, we have $\neg \mathsf{p}(\mathsf{a})$, $\neg \mathsf{q}(z, z)$, $\mathsf{p}(x) \leftarrow \{\mathsf{v}_0\}$, $\mathsf{q}(y, \mathsf{b}) \leftarrow \{\mathsf{v}_1\}$, and $\bot \leftarrow \{\neg \mathsf{v}_0, \neg \mathsf{v}_1\}$. The natural option is to switch model. We take $\mathcal{J}_1 = \{\mathsf{v}_0, \neg \mathsf{v}_1\}$. We then derive $\bot \leftarrow \{\mathsf{v}_0\}$. Since $\mathcal{J}_1 \not\models \bot \leftarrow \{\mathsf{v}_0\}$, we switch to $\mathcal{J}_2 = \{\neg \mathsf{v}_0, \mathsf{v}_1\}$, where we derive $\bot \leftarrow \{\mathsf{v}_1\}$. Finally, we detect that the propositional clauses are unsatisfiable.

We need a fairness criterion for MG that implies local fairness of the underlying $\rhd_{SRed_{\mathrm{F}}}$-derivation. The latter requires a witness $\mathcal{J}$ but gives us no hint as to where to look for one. Our solution involves a topological concept: $\mathcal{J}$ is a *limit point* in $(\mathcal{J}_i)_i$ if there exists a subsequence $(\mathcal{J}'_i)_i$ of $(\mathcal{J}_i)_i$ such that $\mathcal{J} = \mathcal{J}'_\infty = \mathcal{J}'^\infty$.

Example 13. Let $(\mathcal{J}_i)_i$ be the sequence such that $\mathcal{J}_{2i} \cap \mathbf{V} = \{\mathsf{v}_1, \mathsf{v}_3, \ldots, \mathsf{v}_{2i-1}\}$ (i.e., $\mathsf{v}_1, \mathsf{v}_3, \ldots, \mathsf{v}_{2i-1}$ are true and the other variables are false) and $\mathcal{J}_{2i+1} = (\mathcal{J}_{2i} \setminus \{\neg \mathsf{v}_{2i}\}) \cup \{\mathsf{v}_{2i}\}$. Although it is not in the sequence, the interpretation $\mathcal{J} \cap \mathbf{V} = \{\mathsf{v}_1, \mathsf{v}_3, \ldots\}$ is a limit point. The associated split tree is shown in Fig. 1. The direct path from the root to a node $\mathcal{J}_i$ specifies the assertions that are true in $\mathcal{J}_i$.

Example 14. Let $(\mathcal{J}_i)_i$ be such that $\mathcal{J}_0 \cap \mathbf{V} = \emptyset$, $\mathcal{J}_{4i+1} \cap \mathbf{V} = \{\mathsf{v}_0\} \cup \{\mathsf{v}_{4j+3} \mid j < i\}$, $\mathcal{J}_{4i+2} \cap \mathbf{V} = \{\mathsf{v}_0, \mathsf{v}_{4i+2}\} \cup \{\mathsf{v}_{4j+3} \mid j < i\}$, $\mathcal{J}_{4i+3} \cap \mathbf{V} = \{\mathsf{v}_{4j+1} \mid j \leq i\}$, and $\mathcal{J}_{4i+4} \cap \mathbf{V} = \{\mathsf{v}_{4j+1} \mid j \leq i\} \cup \{\mathsf{v}_{4i+4}\}$. This sequence has two limit points: $\mathcal{J}' = \liminf_{i \to \infty} \mathcal{J}_{4i+1}$ and $\mathcal{J}'' = \liminf_{i \to \infty} \mathcal{J}_{4i+3}$. The split tree is depicted in Fig. 2.

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

Definition 15. An $\Longrightarrow_{\mathsf{MG}}$-derivation $(\mathcal{J}_i, \mathcal{N}_i)_i$ is *fair* if either (1) $\bot \in \mathcal{N}_i$ for some i or (2) $\mathcal{J}_i \models (\mathcal{N}_i)_\bot$ for infinitely many indices i and there exists a limit point $\mathcal{J}$ of $(\mathcal{J}_i)_i$ such that $\mathit{FInf}((\mathcal{N}_\infty)_\mathcal{J}) \subseteq \bigcup_i \mathit{FRed}_{\mathrm{I}}((\mathcal{N}_i)_\mathcal{J})$.

Fairness of an $\Longrightarrow_{\mathsf{MG}}$-derivation implies local fairness of the underlying $\rhd_{SRed_{\mathrm{F}}}$-derivation. A well-behaved propositional solver, as in labeled splitting, always gives rise to a single limit point $\mathcal{J}_\infty$, which can be taken for $\mathcal{J}$ in Definition 15.

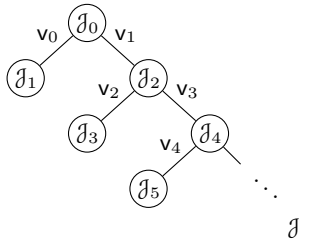

Fig. 1: A split tree with a single infinite branch

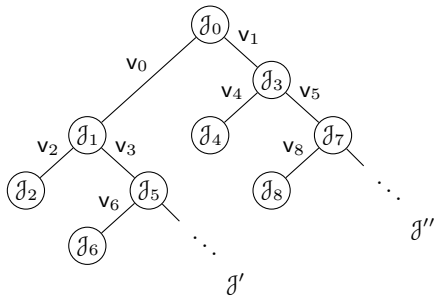

Fig. 2: A split tree with two infinite branches

By contrast, an unconstrained solver, as supported by AVATAR, can produce multiple limit points. Then it is more challenging to ensure fairness.

Example 16. 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. Now suppose that v_{2i} and v_{2i+1} are associated with $\mathsf{p}(\mathsf{f}^i(\mathsf{a}))$ and $\mathsf{q}(\mathsf{f}^i(\mathsf{a}))$, respectively. If we split every emerging $\mathsf{p}(\mathsf{f}^i(\mathsf{a}))\vee\mathsf{q}(\mathsf{f}^i(\mathsf{a}))$ and the SAT solver always makes v_{2i} true first, we end up with the situation of Example 13 and Fig. 1. For the limit point $\mathcal{J}$, all *FInf*-inferences are performed. Thus, the derivation is fair.

Example 17. We build a clause set from two copies of Example 16, where each clause C from each copy $i \in \{1, 2\}$ is extended to $\neg\mathsf{r}_i \vee C$. We add the clause $\mathsf{r}_1 \vee \mathsf{r}_2$ and split it as our first move. From there, each branch imitates Example 16. A SAT solver might jump back and forth, as in Example 14 and Fig. 2. Even if A-clauses get disabled and re-enabled infinitely often, we must perform all nonredundant inferences in at least one of the two limit points ($\mathcal{J}'$ or $\mathcal{J}''$).

5 Locking Provers

Next, we refine the model-guided prover into a locking prover that temporarily locks away A-formulas that are redundant locally w.r.t. some $\mathcal{J}$ but not globally. The states are triples $(\mathcal{J}, \mathcal{N}, \mathcal{L})$, with $\mathcal{L} \subseteq \mathbf{A} \times \mathbf{AF}$. Intuitively, $(B,\, C \leftarrow A) \in \mathcal{L}$ means that $C \leftarrow A$ is "locally redundant" in interpretations $\mathcal{J} \supseteq B$. The function $\llbracket\ \rrbracket$ erases the locks: $\llbracket\mathcal{L}\rrbracket = \{\mathcal{C} \mid (B, \mathcal{C}) \in \mathcal{L} \text{ for some } B\}$. Initial states have the form $(\mathcal{J}, N, \emptyset)$, where $N \subseteq \mathbf{F}$. The *locking prover* is defined by these two rules:

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

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

We note that $\Longrightarrow_{\mathsf{L}}$-derivations refine $\Longrightarrow_{\mathsf{MG}}$-derivations, with states $(\mathcal{J}, \mathcal{N}, \mathcal{L})$ mapped to $(\mathcal{J}, \mathcal{N} \cup \llbracket\mathcal{L}\rrbracket)$.

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 $\mathcal{J}$ that contains $\neg \mathsf{v}_k$. For the models $\mathcal{J}_i = \{\mathsf{v}_1, \ldots, \mathsf{v}_i, \neg \mathsf{v}_{i+1}, \ldots\}$, the clause $\mathsf{p}(\mathsf{c})$ would always be locally redundant and ignored. Yet $\mathsf{p}(\mathsf{c})$ might not be locally redundant at the unique limit point $\mathcal{J} = \mathbf{V}$. We could rule out this counterexample by requiring that derivations are strongly fair—that is, every inference possible infinitely often must eventually be made redundant. However, we have found a counterexample showing that strong fairness does not ensure completeness [8, Example 46]. It would seem that this counterexample could arise with Vampire if the underlying SAT solver produces this specific sequence of interpretations.

Our solution is as follows. Let $(\mathcal{J}_i, \mathcal{N}_i, \mathcal{L}_i)_i$ be an $\Longrightarrow_{\mathsf{L}}$-derivation, let $(\mathcal{J}'_j)_j$ be a subsequence of $(\mathcal{J}_i)_i$, and let $(\mathcal{N}'_j)_j$ be the corresponding subsequence of $(\mathcal{N}_i)_i$. To achieve fairness, we now consider $\mathcal{N}'_\infty$, the A-formulas persistent in the unlocked subsequence $(\mathcal{N}'_j)_j$. By contrast, fairness of $\Longrightarrow_{\mathsf{MG}}$-derivations used $\mathcal{N}_\infty$.

Definition 18. An $\Longrightarrow_{\mathsf{L}}$-derivation $(\mathcal{J}_i, \mathcal{N}_i, \mathcal{L}_i)_i$ is *fair* if either (1) $\bot \in \bigcup_i \mathcal{N}_i$ or (2) $\mathcal{J}_i \models (\mathcal{N}_i)_\bot$ for infinitely many indices i and there exists a subsequence $(\mathcal{J}'_j)_j$ converging to a limit point $\mathcal{J}$ such that $\mathit{FInf}((\mathcal{N}'_\infty)_\mathcal{J} \cup ((\limsup_{j\to\infty} \llbracket \mathcal{L}'_j \rrbracket)_\mathcal{J} \setminus \llbracket \mathcal{L}'^\infty \rrbracket)_\mathcal{J}) \subseteq \bigcup_i \mathit{FRed}_\mathrm{I}((\mathcal{N}_i \cup \llbracket \mathcal{L}_i \rrbracket)_\mathcal{J})$, where $(\mathcal{N}'_j)_j$ and $(\mathcal{L}'_j)_j$ correspond to $(\mathcal{J}'_j)_j$.

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

6 AVATAR-Based Provers

AVATAR was unveiled in 2014 by Voronkov [22]. Since then, he and his colleagues studied many options and extensions [3, 17]. A second implementation, in Lean's `super` tactic, is due to Ebner [9]. Here we attempt to capture AVATAR's essence.

The abstract AVATAR-based prover we define in this section extends the locking prover L with a given clause procedure [13]. 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.

Let $\mathbf{TAF} = \mathbf{AF} \times \mathbb{N}$ be the set of *timestamped A-formulas*. Given $\mathcal{N} \subseteq \mathbf{TAF}$, we define $\wr\mathcal{N}\int = \{\mathcal{C} \mid (\mathcal{C}, t) \in \mathcal{N} \text{ for some } t\}$, and we overload existing notations to erase timestamps. Thus, $\lfloor \mathcal{N} \rfloor = \lfloor \wr\mathcal{N}\int \rfloor$, $\mathcal{N}_\bot = \wr\mathcal{N}\int_\bot$, and so on. Note that we use a new set of calligraphic letters (e.g., $\mathcal{C}, \mathcal{N}$) to range over timestamped A-formulas and A-formulas sets. Using the saturation framework [23, Sect. 3], we lift $(\mathit{SInf}, \mathit{SRed})$ to a calculus $(\mathit{TSInf}, \mathit{TSRed})$ on $\mathbf{TAF}$ with the tiebreaker order $>$ on timestamps, so that $(\mathcal{C}, t + k) \in \mathit{TSRed}_\mathrm{F}(\{(\mathcal{C}, t)\})$ for any $k > 0$.

A state is a tuple $(\mathcal{J}, \mathcal{A}, \mathcal{P}, \mathcal{Q}, \mathcal{L}) \in \mathcal{P}(\mathbf{A}) \times \mathcal{P}(\mathbf{TAF})^3 \times \mathcal{P}(\mathbf{TAF} \times \mathcal{P}_{\text{fin}}(\mathbf{A}))$, where $\mathcal{A}$, $\mathcal{P}$, and $\mathcal{Q}$ are respectively the sets of *active*, *passive*, and other (disabled or propositional) timestamped A-formulas, and $\mathcal{L}$ is the set of locked timestamped A-formulas such that (1) $\mathcal{A}_\bot = \mathcal{P}_\bot = \emptyset$, (2) $\mathcal{A} \cup \mathcal{P}$ is enabled in $\mathcal{J}$, and (3) $\mathcal{Q}_\mathcal{J} \subseteq \{\bot\}$. The *AVATAR-based prover* AV is defined as follows:

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

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

Let $(\mathcal{J}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{Q}_i, \mathcal{L}_i)_i$ be an $\Longrightarrow_{\mathsf{AV}}$-derivation. It is easy to see that it refines the $\Longrightarrow_\mathsf{L}$-derivation $(\mathcal{J}_i, \lbag \mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \rbag, \lbag \mathcal{L}_i \rbag)_i$ and that the saturation invariant $\mathit{TSInf}(\mathcal{A}_i) \subseteq \mathit{TSRed}_\mathrm{I}(\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \llbracket \mathcal{L}_i \rrbracket)$ holds if $\mathcal{A}_0 = \emptyset$.

In contrast with nonsplitting provers, for AV, fairness w.r.t. formulas does not imply fairness w.r.t. inferences. A problematic scenario involves two premises $\mathcal{C}, \mathcal{D}$ of an inference ι and four transitions repeated forever, possibly with other steps interleaved: Infer makes $\mathcal{C}$ active; Switch disables it; Infer makes $\mathcal{D}$ active; Switch disables it. Even though $\mathcal{C}$ and $\mathcal{D}$ are selected in a strongly fair fashion, ι is never performed. We need an even stronger fairness criterion.

Definition 19. An $\Longrightarrow_{\mathsf{AV}}$-derivation $(\mathcal{J}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{Q}_i, \mathcal{L}_i)_i$ is *fair* if (1) $\bot \in \lbag \bigcup_i \mathcal{Q}_i \rbag$ or (2) $\mathcal{J}_i \models (\mathcal{Q}_i)_\bot$ for infinitely many indices i and there exists a subsequence $(\mathcal{J}'_j)$ converging to a limit point $\mathcal{J}'_\infty$ such that (3) $\liminf_{j\to\infty} \mathit{TSInf}(\mathcal{A}'_j, \mathcal{P}'_j) = \emptyset$ and (4) $(\limsup_{j\to\infty} \llbracket \mathcal{L}'_j \rrbracket)_\mathcal{J} \setminus \llbracket \mathcal{L}'^\infty \rrbracket_\mathcal{J} \subseteq \bigcup_i \mathit{FRed}_\mathrm{F}((\mathcal{A}_i \cup \mathcal{P}_i \cup \mathcal{Q}_i \cup \llbracket \mathcal{L}_i \rrbracket)_\mathcal{J})$.

Condition (3) ensures that all inferences involving passive A-formulas are redundant at the limit point. It would not suffice to require $\mathcal{P}'_\infty = \emptyset$ because A-formulas can move back and forth between $\mathcal{A}$, $\mathcal{P}$, and $\mathcal{Q}$, as we just saw. Condition (4) is similar to the condition on locks in Definition 18. If the $\Longrightarrow_{\mathsf{AV}}$-derivation is fair, the corresponding $\Longrightarrow_\mathsf{L}$-derivation is also fair.

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 always select $\mathcal{C}$

before $\mathcal{D}$ if both are present. We use two selection orders: $\lessdot_{\mathbf{TAF}}$, based on timestamps, must be followed infinitely often; $\lessdot_{\mathbf{F}}$ must be followed otherwise. For the first one, we can use $\lessdot_{\text{age}}$ defined so that $(\mathcal{C}, t) \lessdot_{\text{age}} (\mathcal{C}', t')$ if $t < t'$.

Definition 20. Let X be a set. A *selection order* $\lessdot$ on X is an irreflexive and transitive relation such that $\{y \mid y \not\gtrdot x\}$ is finite for all $x \in X$.

The intersection of two orders $\lessdot_1$ and $\lessdot_2$ corresponds to the nondeterministic alternation between them. The prover may choose either a $\lessdot_1$-minimal or a $\lessdot_2$-minimal A-formula, at its discretion.

To ensure completeness, 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. Given $\mathcal{N} \subseteq \mathbf{AF}$, let $\lceil \mathcal{N} \rceil = \{A \mid C \leftarrow A \in \mathcal{N} \text{ for some } C\}$.

Definition 21. A function $F : \mathcal{P}(\mathbf{AF}) \to \mathcal{P}(\mathbf{AF})$ is *strongly finitary* if $\lfloor F(\mathcal{N}) \rfloor$ and $\bigcup \lceil F(\mathcal{N}) \rceil \setminus \bigcup \lceil \mathcal{N} \rceil$ are finite for any $\mathcal{N} \subseteq \mathbf{AF}$ such that $\lfloor \mathcal{N} \rfloor$ is finite.

Intuitively, a strongly finitary function F returns finitely many base formulas and finitely many new assertions, although it may return infinitely many A-formulas. Clearly, $F(\mathcal{N})$ is finite for any finite $\mathcal{N} \subseteq \mathbf{AF}$. If $\mathit{FInf}(N)$ is finite for any finite $N \subseteq \mathbf{F}$, then performing *SInf*-inferences is strongly finitary. Deterministic Split rules, such as AVATAR's, are also strongly finitary. We can lift a strongly finitary F to any $\mathcal{N} \subseteq \mathbf{TAF}$ by taking $F_{\mathbf{TAF}}(\mathcal{N}) = F(\lfloor \mathcal{N} \rfloor) \times \mathbb{N}$. If F and G are strongly finitary, then so is $\mathcal{N} \mapsto F(\mathcal{N}) \cup G(\mathcal{N})$.

Simplification rules used by the prover must be restricted even more to ensure completeness, because they can lead to new splits and assertions. For example, simplifying $\mathsf{p}(x * 0) \vee \mathsf{p}(x)$ to $\mathsf{p}(0) \vee \mathsf{p}(x)$ transforms an unsplittable clause into a splittable one. If simplifications were to produce infinitely many such clauses, the prover might split and switch models forever without making progress.

Definition 22. Let $\prec$ be a well-founded relation on $\mathbf{F}$, and let $\preceq$ be its reflexive closure. A function $S : \mathbf{AF} \to \mathcal{P}(\mathbf{AF})$ is a *strongly finitary simplification bound* for $\prec$ if $\mathcal{N} \mapsto \bigcup_{\mathcal{C} \in \mathcal{N}} S(\mathcal{C})$ is strongly finitary and $\lfloor \mathcal{C}' \rfloor \preceq \lfloor \mathcal{C} \rfloor$ for all $\mathcal{C}' \in S(\mathcal{C})$.

The prover may simplify an A-formula $\mathcal{C}$ to $\mathcal{C}'$ only if $\mathcal{C}' \in S(\mathcal{C})$. It may also delete $\mathcal{C}$. Strongly finitary simplification bounds are closed under unions, allowing the combination of simplification techniques based on $\prec$. For superposition, a natural choice for $\prec$ is the clause order. The key property of strongly finitary simplification bounds is that if we saturate a finite set of A-formulas w.r.t. simplifications, the saturation is also finite.

Example 23. Let $\mathbf{F}$ be the set of first-order clauses and $S(C \leftarrow A) = \{C' \leftarrow A' \mid C' \text{ is a subclause of } C \text{ and } A' \subseteq A\}$. Then S is a strongly finitary simplification bound. This S covers many simplification techniques, including elimination of duplicate literals, deletion of resolved literals, and subsumption resolution.

Example 24. If the Knuth–Bendix order [12] is used and all weights are positive, then $S(C \leftarrow A) = \{C' \leftarrow A' \mid C' \prec C \text{ and } A' \subseteq A\}$ is a strongly finitary simplification bound. This can be used to cover demodulation.

Equipped with the above definitions, we introduce a fairness criterion that is more concrete and easier to apply than fairness of $\Longrightarrow_{\mathsf{AV}}$-derivations. We could refine AV further and use this criterion to show the completeness of an imperative procedure such as Voronkov's extended Otter loop [22, Fig. 3], thus showing that Vampire with AVATAR is complete if locking is sufficiently restricted.

Lemma 25. *Let I be a strongly finitary function, and let S be a strongly finitary simplification bound. Then a well-timestamped $\Longrightarrow_{\mathsf{AV}}$-derivation $(\mathcal{J}_i, \mathcal{A}_i, \mathcal{P}_i, \mathcal{Q}_i, \mathcal{L}_i)_i$ is fair if all of the following conditions hold:*

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

7 Application to Other Architectures

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

Splitting without Backtracking. Before AVATAR, Riazanov and Voronkov [20] had already experimented with splitting in Vampire in a lighter variant without backtracking. They based their work on ordered resolution O with selection [2]. Weidenbach [24, end of Sect. 4.5] independently outlined the same technique. The basic idea is to extend the signature Σ with a countable set $\mathbb{P}$ of nullary predicate symbols and to augment the base calculus with a binary splitting rule that replaces a Σ-clause $C \vee D$ with two $\Sigma_{\mathbb{P}}$-clauses $C \vee \mathsf{p}$ and $D \vee \neg\mathsf{p}$. Riazanov and Voronkov require that the precedence $\prec$ makes all $\mathbb{P}$-literals smaller than the Σ-literals. Binary splitting is then a simplification. They also extend the selection function of the base calculus to support $\mathbb{P}$-literals. Their *parallel* selection function imitates as much as possible the original selection function.

The calculus $\mathsf{O}_{\mathbb{P}}$ is closely related to an instance of our framework. Let $\mathbf{F}$ be the set of Σ-clauses, with the empty clause as $\bot$. Let $\mathsf{O} = (\mathit{FInf}, \mathit{FRed})$ be the base calculus. We take $\mathbf{V} = \mathbb{P}$. Let $\mathsf{LA} = (\mathit{SInf}, \mathit{SRed})$, whose name stands for *lightweight AVATAR*, be the induced splitting calculus. Lightweight AVATAR amounts to the splitting architecture Cruanes implemented in Zipperposition [7, Sect. 2.5]. Binary splitting can be realized in LA as a Split-like simplification

rule. The calculi $\mathsf{O}_\mathbb{P}$ and LA disagree slightly because $\mathsf{O}_\mathbb{P}$'s order $\prec$ can break ties using $\mathbb{P}$-literals and because LA can detect unsatisfiability early using the Unsat rule. Despite its slightly weaker order, LA is tighter than $\mathsf{O}_\mathbb{P}$ in the sense that saturation w.r.t. $\mathsf{O}_\mathbb{P}$ implies saturation w.r.t. LA but not vice versa.

Labeled Splitting. Labeled splitting, as originally described by Fietzke and Weidenbach [10] 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 [15]. It works on states $(\Psi, \mathcal{N})$, where Ψ is a stack storing the current state of the split tree and $\mathcal{N}$ is a set of *labeled clauses*—clauses annotated with finite sets of natural numbers.

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

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

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

where C and D share no variables and k is the next split level. Unlike AVATAR, labeled splitting keeps the premise and might split it again with another level.

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

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

Switch is powerful enough to support all of Fietzke and Weidenbach's backtracking rules, but to explore the tree in the same order as they do, we must choose the new model carefully. If a left branch is closed, the model must be updated so as to disable the splits that were not used to close this branch and to enable the right branch. If a right branch is closed, the split must be disabled,

and the model must switch to the right branch of the closest enabled split above it with an enabled left branch. If a right branch is closed but there is no split above with an enabled left branch, the entire tree has been visited. Then, a propositional clause $\bot \leftarrow A$ with $A \subseteq \bigcup_i \{\mathsf{s}_i\}$ is $\models$-entailed by the A-clause set, and StrongUnsat can finish the refutation by exploiting $\mathit{fml}(\mathsf{s}_i) = \top$.

The above strategy helps achieve fairness, because it ensures that there exists exactly one limit point. It also uses locks in a well-behaved way. This means we can considerably simplify the notion of fairness for $\Longrightarrow_{\mathsf{L}}$-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. For diverging derivations, Fietzke and Weidenbach construct a limit subsequence $(\Phi'_i, \mathcal{N}'_i)_i$ of the derivation $(\Phi_i, \mathcal{N}_i)_i$ and require that every persistent inference in it be made redundant, exactly as we do for $\Longrightarrow_{\mathsf{L}}$-derivations. The subsequence consists of all states that lie on the split tree's unique infinite branch. Locks are well behaved, with $\limsup_{j\to\infty} \llbracket \mathcal{L}'_j \rrbracket = \llbracket \mathcal{L}'^{\infty} \rrbracket$, 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.

SMT with Quantifiers. Satisfiability modulo theories (SMT) solvers based on DPLL(T) [15] combine a SAT solver with theory solvers. In the classical setup, the theories are decidable, and the SMT solver is a decision procedure for the union of the theories. Some SMT solvers also support quantified formulas via instantiation at the expense of decidability.

Complete instantiation strategies have been developed for various fragments of first-order logic [11,18,19]. In particular, enumerative quantifier instantiation [18] 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 considered above, 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.

Let $\mathbf{F}$ be the set of first-order Σ-formulas. We represent the SMT solver's underlying SAT solver by the Unsat rule and complement it with an inference system *FInf* that includes rules for clausification outside quantifiers, theory reasoning, and instantiation. The clausification rules derive C and D from a premise $C \wedge D$, among others; the theory rules derive $\bot$ from some Σ-formula set N such that $N \models \{\bot\}$, ignoring quantifiers; and the instantiation rules derive $\varphi(u)$ from premises $\forall x.\, \varphi(x)$, where u is a ground term. For *FRed*, we take an arbitrary instance of standard redundancy. Its only purpose is to split disjunctions destructively. We define the "theories with quantifiers" calculus $\mathsf{TQ} = (\mathit{FInf}, \mathit{FRed})$. For $\models$ and $\mathrel{|\!\approx}$, we use entailment in the supported theories including quantifiers.

We use the same approximation function as in AVATAR (Example 3). Let us call $C \leftarrow A$ 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 26 (Dynamic completeness). *Assume* TQ *is statically complete for subunit sets. Let* $(\mathcal{J}_i, \mathcal{N}_i)_i$ *be a fair* $\Longrightarrow_{\mathsf{MG}}$*-derivation based on* TQ*. If* $\mathcal{N}_0 \models \{\bot\}$ *and* $\mathcal{N}_\infty$ *contains only subunits, then* $\bot \in \mathcal{N}_j$ *for some* j*.*

Like AVATAR-based provers, SMT solvers will typically not perform all *SInf*-inferences, not even up to SRed_I. Given $\mathsf{a} \approx \mathsf{b} \leftarrow \{\mathsf{v}_0\}$, $\mathsf{b} \approx c \leftarrow \{\mathsf{v}_1\}$, $\mathsf{a} \approx \mathsf{d} \leftarrow \{\mathsf{v}_2\}$, $\mathsf{c} \approx \mathsf{d} \leftarrow \{\mathsf{v}_3\}$, and $\mathsf{a} \not\approx \mathsf{c} \leftarrow \{\mathsf{v}_4\}$, 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. For decidable theories, a practical fair strategy is to instantiate quantifiers only if no other rules are applicable.

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 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 [4–6, 14].

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

This research has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No. 713999, Matryoshka). The research has also received funding from the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) under the Vidi program (project No. 016.Vidi.189.037, Lean Forward).

References

1. Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. J. Log. Comput. 4(3), 217–247 (1994)
2. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. I, pp. 19–99. Elsevier (2001)
3. Bjø[r]ner, N., Reger, G., Suda, M., Voronkov, A.: AVATAR modulo theories. In: Benzmüller, C., Sutcliffe, G., Rojas, R. (eds.) GCAI 2016. EPiC Series in Computing, vol. 41, pp. 39–52. EasyChair (2016)
4. Bonacina, M.P., Graham-Lengrand, S., Shankar, N.: Satisfiability modulo theories and assignments. In: de Moura, L. (ed.) CADE-26. LNCS, vol. 10395, pp. 42–59. Springer (2017)
5. Bonacina, M.P., Lynch, C., de Moura, L.: On deciding satisfiability by DPLL($\Gamma + T$) and unsound theorem proving. In: Schmidt, R.A. (ed.) CADE-22. LNCS, vol. 5663, pp. 35–50. Springer (2009)
6. Bonacina, M.P., Plaisted, D.A.: SGGS theorem proving: An exposition. In: Schulz, S., de Moura, L., Konev, B. (eds.) PAAR-2014. EPiC Series in Computing, vol. 31, pp. 25–38. EasyChair (2014)
7. Cruanes, S.: Extending Superposition with Integer Arithmetic, Structural Induction, and Beyond. Ph.D. thesis, École polytechnique (2015)
8. Ebner, G., Blanchette, J., Tourret, S.: A unifying splitting framework (technical report). Technical report (2021), `https://matryoshka-project.github.io/pubs/splitting_report.pdf`
9. 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)
10. Fietzke, A., Weidenbach, C.: Labelled splitting. Ann. Math. Artif. Intell. 55(1–2), 3–34 (2009)
11. Ge, Y., de Moura, L.: Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 306–320. Springer (2009)
12. Knuth, D.E., Bendix, P.B.: Simple word problems in universal algebras. In: Leech, J. (ed.) Computational Problems in Abstract Algebra. pp. 263–297. Pergamon Press (1970)
13. McCune, W., Wos, L.: Otter—the CADE-13 competition incarnations. J. Autom. Reason. 18(2), 211–220 (1997)
14. de Moura, L., Jovanović, D.: A model-constructing satisfiability calculus. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) VMCAI 2013. LNCS, vol. 7737, pp. 1–12. Springer (2013)
15. 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)
16. Nipkow, T., Klein, G.: Concrete Semantics: With Isabelle/HOL. Springer (2014)
17. Reger, G., Suda, M., Voronkov, A.: Playing with AVATAR. In: Felty, A.P., Middeldorp, A. (eds.) CADE-25. LNCS, vol. 9195, pp. 399–415. Springer (2015)
18. Reynolds, A., Barbosa, H., Fontaine, P.: Revisiting enumerative instantiation. In: Beyer, D., Huisman, M. (eds.) TACAS 2018. LNCS, vol. 10806, pp. 112–131. Springer (2018)

19. Reynolds, A., Tinelli, C., Goel, A., Krstić, S.: Finite model finding in SMT. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 640–655. Springer (2013)
20. Riazanov, A., Voronkov, A.: Splitting without backtracking. In: Nebel, B. (ed.) IJCAI 2001. pp. 611–617. Morgan Kaufmann (2001)
21. Sutcliffe, G.: The TPTP problem library and associated infrastructure—from CNF to TH0, TPTP v6.4.0. J. Autom. Reason. 59(4), 483–502 (2017)
22. Voronkov, A.: AVATAR: The architecture for first-order theorem provers. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 696–710. Springer (2014)
23. Waldmann, U., Tourret, S., Robillard, S., Blanchette, J.: A comprehensive framework for saturation theorem proving. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) IJCAR 2020, Part I. LNCS, vol. 12166, pp. 316–334. Springer (2020)
24. Weidenbach, C.: Combining superposition, sorts and splitting. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. II, pp. 1965–2013. Elsevier and MIT Press (2001)