Journal of Automated Reasoning manuscript No.
(will be inserted by the editor)

A Combined Superposition and Model Evolution Calculus

Peter Baumgartner · **Uwe Waldmann**

October 11, 2010

Abstract We present a new calculus for first-order theorem proving with equality, $\mathcal{ME}$+Sup, which generalizes both the Superposition calculus and the Model Evolution calculus (with equality) by integrating their inference rules and redundancy criteria in a non-trivial way. The main motivation is to combine the advantageous features of these two rather complementary calculi in a single framework. In particular, Model Evolution, as a lifted version of the propositional DPLL procedure, contributes a non-ground splitting rule that effectively permits to split a clause into *non* variable disjoint subclauses. In the paper we present the calculus in detail. Our main result is its completeness under semantically justified redundancy criteria and simplification rules. We also show how under certain assumptions the model representation computed by a (finite and fair) derivation can be queried in an effective way.

1 Introduction

We present a new calculus for first-order theorem proving with equality, $\mathcal{ME}$+Sup, which generalizes both the Superposition calculus and the Model Evolution calculus (with equality), $\mathcal{ME}_{\mathrm{E}}$. It integrates the inference rules of Superposition and of Model Evolution in a non-trivial way while preserving the individual semantically-based redundancy criteria of the two calculi. The inference rules are controlled by a flexible labelling function on atoms. This permits non-trivial combinations where inference rule applicability is disjoint, but pure forms of both calculi can be (trivially) configured, too.

On a methodological level, this paper attempts to bridge the gap between instance-based methods (per $\mathcal{ME}_{\mathrm{E}}$) and Resolution methods (per Superposition). Both methods are rather successful, for instance in terms of performance of implemented systems at the annual CASC theorem proving competition. However, they currently stand rather separated. They provide decision procedures for different sub-classes of first-order logic, and their inference rules are incompatible, too. For instance, subsumption deletion can be used with instance-based methods in only a limited way.

Peter Baumgartner
NICTA and ANU, Canberra, Australia

Uwe Waldmann
MPI für Informatik, Saarbrücken, Germany

The main motivation for this work is to combine the advantages of both calculi in a single framework. Technically, $\mathcal{ME}$+Sup can be seen as an extension of the essential Model Evolution inference rules by Superposition inference rules. Alternatively, $\mathcal{ME}$+Sup can be seen to extend Superposition with a new splitting rule that permits, as a special case, to split a clause into *non* variable disjoint subclauses. It seems not too difficult to extend current Superposition theorem provers with the new splitting rule, in particular those that already provide infrastructure for a weaker form of splitting (such as SPASS [WSH$^+$07]). Finally, another motivation for this work is to simplify the presentation of $\mathcal{ME}_{\mathrm{E}}$ by aligning it with the better-known Superposition framework.

The following clause set is prototypical for the intended applications of $\mathcal{ME}$+Sup (function symbols are typeset in sans-serif and variables in *italics*).

(1) $x \leq z \vee \neg(x \leq y) \vee \neg(y \leq z)$
(2) $x \leq y \vee y \leq x$
(3) $\mathsf{select}(\mathsf{store}(a,i,e),i) \approx e$
(4) $\mathsf{select}(\mathsf{store}(a,i,e),j) \approx \mathsf{select}(a,j) \vee i \approx j$
(5) $\mathsf{select}(\mathsf{a0},i) \leq \mathsf{select}(\mathsf{a0},j) \vee i \approx j \vee \neg(i \leq j)$

The clauses (1) and (2) are properties of total orders, clauses (3) and (4) axiomatize arrays, and clause (5) says that the array a0 is sorted. The latter could be written, perhaps more intuitively, as $\mathsf{select}(\mathsf{a0},i) \leq \mathsf{select}(\mathsf{a0},j) \vee \neg(i < j)$ together with the axiom $(x \approx y) \vee x < y \vee \neg(x \leq y)$. We didn't do that just to keep the example small. The clause set (1)-(5) is satisfiable, but neither $\mathcal{ME}_{\mathrm{E}}$ nor Superposition equipped with standard redundancy criteria (with or without selection of negative literals) does terminate on these clauses: for $\mathcal{ME}_{\mathrm{E}}$, the size of the derived literals is not finitely bounded, and for Superposition the length of the derived clauses is not finitely bounded. On the other hand, $\mathcal{ME}$+Sup does terminate on (1)-(5). The following informal calculus preview will demonstrate this (and why).

1.1 Calculus Preview

The main technical novelty of the $\mathcal{ME}$+Sup calculus is the combination and interplay between its $\mathcal{ME}_{\mathrm{E}}$ and Superposition components. We are going to explain the main ideas how this works. The details are quite involved and are left to the main part of the paper.

$\mathcal{ME}$+Sup is parametrized by a term ordering as customarily used both in Superposition calculi and $\mathcal{ME}_{\mathrm{E}}$ to formulate restrictions on inference rule applicability and to justify simplification techniques. $\mathcal{ME}$+Sup is further parametrized in a non-standard way by a *labelling function* that partitions the set of ground atoms into two sets, the *split atoms* and the *superposition atoms*. Intuitively, the $\mathcal{ME}_{\mathrm{E}}$-part of $\mathcal{ME}$+Sup takes care of (explicitly) representing an interpretation for the split atoms, and the Superposition part of $\mathcal{ME}$+Sup takes care of (implicitly) representing an interpretation for the superposition atoms. A key point is that the former, *explicit* representation allows one to formulate certain "semantic" restrictions on the inference rules based on (simple) querying this representation. The example above will be instrumental to demonstrate this.

The clauses (1) and (2) alone are problematic as Superposition does not terminate on them. $\mathcal{ME}_{\mathrm{E}}$, like all instance-based methods, is a decision procedure for function-free clause sets and thus terminates on (1) and (2) alone. This suggests to treat clauses (1) and (2) mainly by the $\mathcal{ME}_{\mathrm{E}}$ component of $\mathcal{ME}$+Sup. Accordingly, we declare all $\leq$-atoms as split atoms and all $\approx$-atoms as superposition atoms. Intuitively, function-free non-equalities are prime candidates for split literals, whereas equalities, in particular between deep terms, are usually better handled using the superposition part of the calculus. To get an instructive example we assume a term ordering $\succeq$ that makes all $\leq$-atoms strictly greater than all $\approx$-atoms.

Let us now have a look at the $\mathcal{ME}$+Sup data structures and inference rules. The calculus maintains two main data structures, *contexts* and sets of *constrained clauses*. A context Λ is a set of possibly non-ground split literals that specifies a Herbrand interpretation on the split literals by utilizing the instantiation preorder on terms: a positive ground literal L is true in (the interpretation specified by) Λ if and only if (a) L is an instance of some literal $K \in \Lambda$, written as $K \gtrsim L$, and (b) there is no $K' \in \Lambda$ such that $K \gtrsim \overline{K'} \gtrsim L$. For example, if $\Lambda = \{x \leq y,\ \neg(z \leq \mathsf{b}),\ \mathsf{a} \leq \mathsf{b}\}$ then $\mathsf{a} \leq \mathsf{a}$, $\mathsf{a} \leq \mathsf{b}$ and $\mathsf{b} \leq \mathsf{a}$ are all true in Λ, whereas $\mathsf{b} \leq \mathsf{b}$ is false. For the latter notice that although $x \leq y \in \Lambda$ is a candidate to make $\mathsf{b} \leq \mathsf{b}$ true, there is a more specific complementary literal $\neg(z \leq \mathsf{b}) \in \Lambda$ that prevents that, i.e., $x \leq y \gtrsim z \leq \mathsf{b} \gtrsim \mathsf{b} \leq \mathsf{b}$. Finally, for technical reasons we assume that every context implicitly contains a pseudo-literal $\neg x$, where x is a *variable*, which (explicitly) assigns false to every split atom "by default".

We stress that this description is only approximative. It is sufficient for the example, but in general the atoms assigned true by a context are only a subset of the ones according to the construction above. But notice that the construction still gives a simple, sufficient condition for ground atoms being false, which is exploited to formulate inference rule restrictions.

A *(constrained) clause* is an expression of the form $C \cdot \Gamma$, where C is an ordinary clause and Γ is a *constraint*, that is, a multiset of literals. Together with a context Λ, any ground instance $C' \cdot \Gamma'$ of $C \cdot \Gamma$ evaluates to the ordinary (ground clause) C' if all literals in Γ' are split literals and are true in Λ. Otherwise $C' \cdot \Gamma'$ can be ignored. Operationally, the constraint part Γ of a constrained clause $C \cdot \Gamma$ contains the accumulated assumptions from a context that were used in deriving the clause.

The calculus maintains at any time a current context Λ, initially empty, and a current clause set Φ, initially the input clauses attached with empty constraints. The main inference rules grow Λ or Φ, but there are also simplification rules to modify Φ destructively. For the latter see Section 9. We describe the inference rules that grow Φ next.

$\mathcal{ME}$+Sup includes the usual Superposition calculus inference rules between clauses adapted to constrained clauses. In these rules the constraints are just passively accumulated. For the purpose of this preview it suffices to work with the following uniform, less restrictive formulation of the inference rules Sup-Pos and Sup-Neg in Section 5 (L is a literal):

$$\mathsf{Sup}\ \frac{l \approx r \vee C' \cdot \Gamma' \qquad L[u]_p \vee C \cdot \Gamma}{(L[r]_p \vee C \vee C' \cdot (\Gamma, \Gamma'))\sigma}$$

where (i) σ is an mgu of l and u, (ii) u is not a variable, (iii) $(l \approx r)\sigma$ is a superposition atom, (iv) $r\sigma \not\succeq l\sigma$, (v) $(l \approx r)\sigma$ is strictly maximal wrt. $\succeq$ in $(l \approx r \vee C')\sigma$, (vi) $L\sigma$ is maximal wrt. $\succeq$ in $(L \vee C)\sigma$, and (vii) if $L[u]_p$ is a positive superposition literal of the form $s[u]_p \approx t$ then $t\sigma \not\succeq s\sigma$.

In the example, the only applicable Sup-inferences are among clauses (3) and (4). The terms l and s mentioned in the inference rule can only be the left side of the equation in (3) and the left side of the left equation in (4). However, each such Sup-inference results in a tautology, a clause containing a trivial equation of the form $t \approx t$, and need not be carried out. In the terminology used below, such inferences are "universally redundant". Had we used $\mathcal{ME}_{\mathrm{E}}$ alone, condition (vii) would not apply and paramodulation into the right-hand sides of the mentioned equations in (3) and (4) would be possible, leading to non-termination.

There are additional inference rules for factorisation and for removal of trivial negative equations from clauses, which we do not spell out in this preview.

The next two inference rules combine a literal A ($\neg A$, respectively) that is taken from the current context and a constrained clause in a unit-resolution way.

$$\textsf{U-Sup}\ \frac{A \qquad \neg B \vee C \cdot \Gamma}{(C \cdot (\Gamma, B))\sigma} \qquad \textsf{U-Res}\ \frac{\neg A \qquad B \vee C \cdot \Gamma}{(C \cdot (\Gamma, \neg B))\sigma}$$

where (i) σ is an mgu of A and B, (ii) $A\sigma$ is a split atom, and (iii) $\neg B\sigma$ is maximal wrt. $\succeq$ in $(\neg B \vee C)\sigma$ ($B\sigma$ is maximal wrt. $\succeq$ in $(B \vee C)\sigma$, respectively).

The U-Sup and U-Res inference rules can be seen to identify in their right premise a clause instance with a (maximal) literal that is potentially or actually falsified in the current context Λ. That clause literal is then put aside into the constraint part for later processing by $\mathcal{ME}_{\mathrm{E}}$ rules. Notice that A cannot be treated as a (universally quantified) unit clause, one that forces all instances of A to be true. A future extension of the current context might contain an instance of $\neg A$, so that ground instances of A then become false again. Simply removing $\neg B$ from the right premise would therefore be unsound. This is why $B\sigma$ is stored as an assumption in the constraint in the resulting constrained clause. The same argumentation applies to U-Res.

All the above inference rules are governed by the additional restriction that the constraints of all the involved clauses must be satisfiable in the current context Λ. That is, if $C \cdot \Gamma$ is such a clause, then at least one ground instance Γ' of Γ must be true in Λ. This is justified, intuitively, because a constrained clause $C \cdot \Gamma$ with an unsatisfiable constraint does never evaluate to a ground instance C' of C.

It should be mentioned that the description of contexts and the U-Sup and U-Res rules above is simplified. Below we will define contexts over *rewrite literals*, that is, ordered equations of the form $l \to r$. An ordinary literal like $x \geq y$ will then be written as $(x \geq y) \to \mathrm{tt}$, where tt is a dedicated constant, standing for "true". The simplification is justified because in the example all split-literals are originally non-equations. In general, equations can be split-literals, too, and then a more general form of the U-Sup inference rule is needed. In fact, U-Sup comes in two versions below, corresponding to unit-superposition into positive and negative clauses too.

We can now start a derivation from clauses (1)-(5) and the initial context Λ that (implicitly) contains only $\neg x$:

(6)	$\neg(x \leq y) \vee \neg(y \leq z) \cdot \neg(x \leq z)$	(U-Res of $\neg x \in \Lambda$ and (1))
(7)	$y \leq x \cdot \neg(x \leq y)$	(U-Res of $\neg x \in \Lambda$ and (2))
(8)	$\Box \cdot (\neg(x \leq y), \neg(y \leq x))$	(U-Res of $\neg x \in \Lambda$ and (7))
(9)	$i \approx j \vee \neg(i \leq j) \cdot \neg(\mathsf{select}(\mathsf{a0}, i) \leq \mathsf{select}(\mathsf{a0}, j))$	(U-Res of $\neg x \in \Lambda$ and (5))

All applicable Sup-inferences are redundant at this point, and no other inference rules are applicable at all. For instance, U-Pos is not applicable to (9) and its literal $\neg(i \leq j)$, as there is no complementary literal in Λ to pair with. Intuitively, all $\leq$-literals are false in the current context $\Lambda = \{\}$, and so clause (9) is satisfied and needs not be worked on. Also, (9) cannot be used as the left premise in Sup-inferences because its clause literal $i \approx j$ is strictly smaller than its other clause literal $\neg(i \leq j)$.

The presence of the constrained empty clause (8) does not make a refutation, however, because the possibility of a model that falsifies the constraint of (8) has not been excluded. The following two inference rules deal with such constrained empty clauses, by analyzing if their constraint can be satisfied by modifying the context. Technically, they are formulated

on sequents of the form $\Lambda \vdash \Phi$, where Λ is the current context and Φ is the current clause set (read $\Lambda \vdash \Phi$ roughly as "assume the interpretation Λ and show that it satisfies Φ").

$$\textsf{Split}\ \frac{\Lambda \vdash \Phi}{\Lambda, \overline{K} \vdash \Phi \qquad \Lambda, K \vdash \Phi} \qquad \textsf{Close}\ \frac{\Lambda \vdash \Phi}{\Lambda \vdash \Phi, \square \cdot \emptyset}$$

The applicability conditions are as follows: Split is applicable if there is a constrained clause $\square \cdot \Gamma \in \Phi$ such that $K \in \Gamma$ is a split literal and Λ contains neither a variant of K nor a variant of $\overline{K}$.[1] Close is applicable if there is a constrained clause $\square \cdot \Gamma \in \Phi$ such that for every $K \in \Gamma$ the context Λ contains a variant of K.

The example derivation has arrived at the sequent $\Lambda \vdash \Phi = \{\} \vdash (1), \ldots, (9)$. The only applicable inference rule now is Split, on clause (8). It results in the left sequent with context $\Lambda_\mathrm{l} = \{x \leq y\}$ and the right sequent with context $\Lambda_\mathrm{r} = \{\neg(x \leq y)\}$.

The Split rule branches out in two contexts. Let us choose Λ_r as the new current context. Then, Close is applicable with clause (8), which indicates that $\Lambda_\mathrm{r} \vdash \Phi$ is not considered further. As expected, if all branches in a derivation are closed this way, a refutation has been found.[2]

With the right branch being closed, we need to consider the left branch, i.e., the current context is Λ_l now. The literal $x \leq y \in \Lambda_\mathrm{l}$ suggests to consider U-Sup inferences with $x \leq y$ as the left premise now. For example, the U-Sup inference with right premise (9) and its literal $\neg(i \leq j)$ would give $i \approx j \cdot (\neg(\mathsf{select}(\mathsf{a0}, i) \leq \mathsf{select}(\mathsf{a0}, j)), i \leq j)$. However, the constraint of this clause is unsatisfiable, as all instances of its negative literal are false in Λ_l. This is detected by applicability conditions that make this inference impossible (the last condition mentioned with Deduce in Section 6..

Because no (more) inferences are possible the derivation stops with the sequent $\Lambda_\mathrm{l} \vdash \{(1), \ldots, (9)\}$. It represents a certain model of the initial clause set. Section 7 explains the model construction in detail.

The main purpose of this calculus preview was to demonstrate how the advantageous features of the constituent calculi $\mathcal{ME}_\mathrm{E}$ and Superposition can be combined in a beneficial way. For $\mathcal{ME}_\mathrm{E}$, we exploited the finite-model building capabilities for function-free clause sets, which are difficult for Superposition. For Superposition, we exploited the stronger model-building capabilities when deep terms are involved, which is enabled by inference rules that are more restricted than their $\mathcal{ME}_\mathrm{E}$ counterpart. Altogether, this allowed us to argue that inference rules are *not* applicable, leading to termination on this example.

1.2 Related Work

$\mathcal{ME}$+Sup subsumes the Superposition calculus [BG98] and its redundancy concept and also the essentials of propositional DPLL, that is, split and unit propagation for ground literals. Model Evolution [BT03] and Model Evolution with Equality [BT05] are not completely covered, though, since universal literals and some optional inference rules are missing. The

[1] A *split literal* is a possibly negated atom that has at least one instance that is a split atom (recall that the distinction between split atoms and superposition atoms is made at the ground level).

[2] Regarding soundness: instead of applying Close one may imagine to temporarily replace all variables in all literals in the context by some (same) constant and matching the clause used by Close to complementary literals in the resulting context. With this construction it is not hard to argue for the soundness of the calculus, essentially because the branching introduces complementary literals.

model construction that we use has some similarity with the one used for Constraint Superposition [NR95], where one also starts with constructing a model for reduced instances and later extends this to the full clause set provided that this is constraint-free.

Model Evolution belongs to the family of *instance based methods* [Bau07, JW07, Kor09]. Some of them include Resolution type inference rules. For instance, Ordered Semantic Hyper-Linking [PZ00] uses A-Ordered Resolution for simplification purposes. A refinement of the Inst-Gen calculus [GK03, GK04] allows to reduce (by partial instantiation) certain classes of input clause sets to the monadic clause class, and Resolution can be coupled as a decision procedure for this class. Both papers [GK03] and [PZ00] leave more general combination schemes with Resolution as future work.

Indeed, the latest Inst-Gen prover integrates a Resolution as a sub-system. The purpose is to derive "small" clauses by Resolution which can then be used for (proper) subsumption deletion [Kor08]. Notice that this combination approach is incomparable to ours.

Another approach that compares better to ours, in the sense that it also targets separating Resolution and Instantiation-based inferences, has been described in [LM09]. The calculus described there, SIG-Res, allows one to divide an input clause set F into two parts $P \subseteq F$ and $R \subseteq F$. It works by saturating P under Inst-Gen inference rules, and applying (A-Ordered) Resolution inference rules to clauses from $P \cup R$, where at least one of the parent clauses is from R. The SIG-Res calculus differs conceptually from $\mathcal{ME}$+Sup. While $\mathcal{ME}$+Sup uses a labelling function on clause literals, SIG-Res thus uses a labelling function on clauses. More research is needed to assess the implications of this difference. For now it is perhaps fair to say that $\mathcal{ME}$+Sup is somewhat more advanced, as it features built-in equality inference rules and redundancy criteria, which are both missing in SIG-Res.

2 Formal Preliminaries

We consider signatures Σ comprised of a binary predicate symbol $\approx$ (equality), and a finite set of function symbols of given arity (constants are 0-ary function symbols). We also need a denumerable set of variables X disjoint from Σ. Terms (over Σ and X) are defined as usual. If t is a term we denote by $Var(t)$ the set of t's variables. A term t is *ground* iff $Var(t) = \emptyset$. A *substitution* is a mapping of variables to terms that is the identity almost everywhere. We write $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ for the substitution that maps the variable x_i to the term t_i, for $i = 1, \ldots, n$. The empty substitution is denoted by $\emptyset$. A substitution σ is applied to a term t, written as $t\sigma$, by simultaneously applying σ to the variables in t. A *renaming* is a substitution that is a bijection of X onto itself. A substitution γ is a *ground substitution for* t iff $t\gamma$ is ground.

We write $s \gtrsim t$, iff there is a substitution σ such that $s\sigma = t$.[3] We say that s is a *variant of* t, and write $s \sim t$, iff $s \gtrsim t$ and $t \gtrsim s$, or, equivalently, iff there is a renaming ρ such that $s\rho = t$. We write $s \gtrnsim t$ if $s \gtrsim t$ but $s \not\sim t$. The notation $s[t]_p$ means that the term t occurs in the term s at position p, as usual. The index p is left away when not important or clear from the context. Because equality is the only predicate symbol, an *atom* is always an equation $s \approx t$, which is identified with the multiset $\{s,t\}$. Consequently, equations are treated symmetrically, as $s \approx t$ and $t \approx s$ denote the same multiset. A literal is an atom (a *positive literal*) or the negation of an atom (a *negative literal*). Negative literals are generally written $s \not\approx t$ instead of $\neg(s \approx t)$. In the examples below we often write non-equational literals like $P(t_1, \ldots, t_n)$ or $\neg P(t_1, \ldots, t_n)$, which are meant to stand for the equational literals $P(t_1, \ldots, t_n) \approx \mathrm{tt}$ or

[3] Note that many authors would write $s \lesssim t$ in this case.

$P(t_1,\ldots,t_n) \not\approx \mathrm{tt}$, where tt is a fresh constant that is smaller than all other terms. We write $\overline{L}$ to denote the complement of a literal L, i.e. $\overline{A} = \neg A$ and $\overline{\neg A} = A$, for any atom A. A *clause* is a multiset of literals $\{L_1,\ldots,L_n\}$, generally written as a disjunction $L_1 \vee \cdots \vee L_n$. We write $L \vee C$ to denote the clause $\{L\} \cup C$. The empty clause is written as $\Box$. All the notions on substitutions above are extended from terms to atoms, literals and clauses in the obvious way.

Orderings. We suppose as given a reduction ordering $\succ$ that is total on ground Σ-terms.[4] Following usual techniques [BG98, NR95, e.g.], it is extended to an ordering on literals by taking a positive literal $s \approx t$ as the multiset $\{s,t\}$, a negative literal $s \not\approx t$ as the multiset $\{s,s,t,t\}$ and using the extension of $\succ$ to multisets of terms to compare literals. Similarly, clauses are compared by the multiset extension of the ordering on literals. All these (strict, partial) orderings will be denoted by the same symbol, $\succ$. The non-strict orderings $\succeq$ are defined as $s \succeq t$ iff $s \succ t$ or $s = t$. We say that a literal L is *maximal* (*strictly maximal*) in a clause $L \vee C$ iff there is no $K \in C$ with $K \succ L$ ($K \succeq L$).

Rewrite Systems. A *(rewrite) rule* is an expression of the form $l \to r$ where l and r are terms. A *rewrite system* is a set of rewrite rules. We say that a rewrite system R is *ordered by* $\succ$ iff $l \succ r$, for every rule $l \to r \in R$. In this paper we consider only (ground) rewrite systems that are ordered by $\succ$. A term t is *reducible by* $l \to r$ iff $t = t[l]_p$ for some position p, and t is *reducible wrt.* R if it is reducible by some rule in R. The notion *irreducible* means "not reducible". A rewrite system R is *left-reduced* (*fully reduced*) iff for every rule $l \to r \in R$, l is (l and r are) irreducible wrt. $R \setminus \{l \to r\}$. In other words, in a fully reduced rewrite system no rule is reducible by another rule, neither its left hand side *nor its right hand side*.

Interpretations. A *(Herbrand) interpretation* I is a set of ground atoms—exactly those that are true in the interpretation. Validity of ground literals, clauses, and clause sets in a Herbrand interpretation is defined as usual. We write $I \models F$ to denote the fact that I satisfies F, where F is a ground literal, ground clause, or set of ground clauses. As usual, this notation extends to (possibly non-ground) clauses by taking the set of all ground instances of the clause, and to clause sets by taking the set of all ground instances of all its clauses.

Because equality is the only predicate symbol, we can identify every interpretation with a binary relation on ground terms. If this relation is a congruence relation, we call it an *E-interpretation*. If I is an interpretation, we denote by $I^\star$ the smallest interpretation that includes I and is an E-interpretation.

We say that I *E-satisfies* F iff $I^\star \models F$. We say that F *E-entails* F', written $F \models F'$, iff every E-interpretation that satisfies F also satisfies F'.

The above notions are applied to ground rewrite systems instead of interpretations by taking the rules as equations. We write $R^\star \models F$ and mean $\{l \approx r \mid l \to r \in R\}^\star \models F$. It is well-known that any left-reduced (and hence any fully reduced) ordered rewrite system R is convergent,[5] see e.g. [BN98]) and that any ground equation $s \approx t$ is E-satisfied by R, i.e., $R^\star \models s \approx t$ if and only if s and t have the same (unique) normal form wrt. R.

[4] A *reduction ordering* is a strict partial ordering that is well-founded and is closed under context i.e., $s \succ s'$ implies $t[s] \succ t[s']$ for all terms t, and liftable, i.e., $s \succ t$ implies $s\delta \succ t\delta$ for every term s and t and substitution δ.

[5] A rewrite system is convergent iff it is confluent and terminating.

Labelling Function. Broadly speaking, $\mathcal{ME}$+Sup combines inference rules from the Superposition calculus and inference rules resembling those of Model Evolution, but for each atom only a subset of the full set of inference rules is usable. This is controlled by assuming a *labelling function* that partitions the set of positive ground atoms into two sets, the *split atoms* and the *superposition atoms*.[6] We say a (possibly non-ground) atom is a *split atom* (*superposition atom*) iff at least one ground instance is a split atom (superposition atom).

Thus, while a ground atom is either one or the other, the distinction is blurred for non-ground atoms. From a practical point of view, to avoid overlap between the $\mathcal{ME}$ and the superposition inference rules, it is desirable to keep the (non-ground) split atoms and superposition atoms as separate as possible.

The separation into split atoms and superposition atoms is quite flexible. No assumptions regarding disjointness of their underlying signatures or ordering assumptions between their elements are required. For instance, one may declare all ground atoms up to a certain term depth as split atoms. Even the set of non-ground split atoms is finite then, modulo renaming. As will become clear, the *contexts* evolved by the Model Evolution part of $\mathcal{ME}$+Sup are finite then, which might be interesting, e.g., to finitely represent (parts of) a counter-example for non-theorems.

3 Contexts

Contexts have been introduced in [BT03] as the main data structure in the Model Evolution calculus to represent interpretations; they have been adapted to the equality case in [BT05], but here we work with the original definition, which is simpler and more practical. More formally, when l and r are terms, a *rewrite literal* is a rule $l \to r$ or its negation $\neg(l \to r)$, the latter generally written as $l \not\to r$. By treating $\to$ as a predicate symbol, all operations defined on equational literals apply to rewrite literals as well. In particular, $\overline{l \to r} = l \not\to r$ and $\overline{l \not\to r} = l \to r$. If clear from the context, we use non-equational literals $P(t_1,\ldots,t_n)$ or $\neg P(t_1,\ldots,t_n)$ as shorthands for the rewrite literals $P(t_1,\ldots,t_n) \to \mathrm{tt}$ or $P(t_1,\ldots,t_n) \not\to \mathrm{tt}$. We also use the term "literal" to refer to equational literals as introduced earlier or to rewrite literals.

A *context* is a set of rewrite literals that also contains a pseudo-literal $\neg x$, for some variable x. In examples we omit writing $\neg x$ and instead implicitly assume it is present. Where L is a rewrite literal and Λ a context, we write $L \in_\sim \Lambda$ if L is a variant of a literal in Λ. A rewrite literal L is *contradictory with a context* Λ iff $\overline{L} \in_\sim \Lambda$. A context Λ is *contradictory* iff it contains a rewrite literal that is contradictory with Λ. For instance, if $\Lambda = \{\mathsf{f}(x) \to \mathsf{a}, \mathsf{f}(x) \not\to x\}$ then $\mathsf{f}(y) \not\to \mathsf{a}$ and $\mathsf{f}(y) \to y$ are contradictory with Λ, while $\mathsf{f}(\mathsf{a}) \to \mathsf{a}$, $\mathsf{a} \not\to \mathsf{f}(x)$ and $\mathsf{f}(x) \to y$ are not. From now on we assume that all contexts are non-contradictory. This is justified by the fact that the $\mathcal{ME}$+Sup calculus defined below can derive non-contradictory contexts only.

A context can be seen as a means to specify a set of literals. In the $\mathcal{ME}$ calculus without equality [BT03], it was straightforward to obtain a Herbrand interpretation from a context Λ by saying, roughly, that a literal $K \in \Lambda$ specifies a truth value for a ground instance unless Λ contains a literal that is more specific than $\overline{K}$ and that specifies the opposite truth value. This notion, called productivity, is needed here as well:

[6] Notice that with the symmetric treatment of equations, $l \approx r$ is a split atom if and only if $r \approx l$ is, and similarly for superposition atoms.

Definition 1 (Productivity [BT03]) Let L be a rewrite literal and Λ a context. A rewrite literal K *produces* L *in* Λ iff $K \gtrsim L$ and there is no $K' \in \Lambda$ such that $K \gtrsim \overline{K'} \gtrsim L$.

The context Λ *produces* L iff it contains a literal K that produces L in Λ, and Λ *produces* a multiset Γ of rewrite literals iff Λ produces each $L \in \Gamma$. □

For instance, the context Λ above produces $\mathsf{f}(\mathsf{b}) \to \mathsf{a}$, $\mathsf{f}(\mathsf{a}) \to \mathsf{a}$ and $\mathsf{f}(\mathsf{a}) \not\to \mathsf{a}$, but Λ produces neither $\mathsf{f}(\mathsf{a}) \to \mathsf{b}$ nor $\mathsf{a} \to \mathsf{f}(x)$.

In case of conflicts, when a context produces both a ground literal and its complement, the positive literal is preferred. The following definition achieves this:

$$\Pi_\Lambda := \{l \to r \mid \Lambda \text{ produces } l \to r \text{ and } l \to r \text{ is ground}\} \ .$$

For instance, if $\Lambda = \{\mathsf{f}(x) \to x\}$ and Σ consists of a constant a and the unary function symbol f then $\Pi_\Lambda = \{\mathsf{f}(\mathsf{a}) \to \mathsf{a}, \mathsf{f}(\mathsf{f}(\mathsf{a})) \to \mathsf{f}(\mathsf{a}), \ldots\}$.

In other words, Π_Λ specifies a Herbrand interpretation on rewrite literals. We already used Π_Λ in this way to discuss the example in Section 1.1. In general, however, a more refined construction is needed to obtain the final rewrite system, a subset of Π_Λ, that Λ stands for. The details are left to Section 7.

4 Constrained Clauses

Let $C = L_1 \vee \cdots \vee L_n$ be a clause, let $\Gamma = \{K_1, \ldots, K_m\}$ be a multiset of rewrite literals such that no K_i is of the form $x \to t$, where x is a variable and t is a term. The expression $C \cdot \Gamma$ is called a *constrained clause (with constraint Γ)*, and we generally write $C \cdot (K_1, \ldots, K_m)$ instead of $C \cdot \{K_1, \ldots, K_m\}$. The notation $C \cdot (\Gamma, K)$ means $C \cdot \Gamma \cup \{K\}$.

Informally, the constraint part Γ of a constrained clause $C \cdot \Gamma$ contains the accumulated assumptions from a context that were used in deriving the clause. The literals in the clause part C are going to be used for superposition inferences or are moved into the constraint part, dependent on their type (superposition literal/split literal).

Applying a substitution σ to $C \cdot \Gamma$, written as $(C \cdot \Gamma)\sigma$, means to apply σ to C and all literals in Γ. A constrained clause $C \cdot \Gamma$ is *ground* iff both C and Γ are ground. If γ is a ground substitution for $C \cdot \Gamma$, then the pair $(C \cdot \Gamma; \gamma)$ is called a *ground closure (of $C \cdot \Gamma$)*. For a set of constrained clauses Φ, Φ^{gr} is the set of all ground closures of all constrained clauses in Φ.

For the soundness proof of $\mathcal{ME}$+Sup we need the *clausal form* of a constrained clause $C \cdot \Gamma = L_1 \vee \cdots \vee L_m \cdot (l_1 \to r_1, \ldots, l_k \to r_k, l_{k+1} \not\to r_{k+1}, \ldots, l_n \not\to r_n)$, which is the ordinary clause $L_1 \vee \cdots \vee L_m \vee l_1 \not\approx r_1 \vee \cdots \vee l_k \not\approx r_k \vee l_{k+1} \approx r_{k+1} \vee \cdots \vee l_n \approx r_n$ and which we denote by $(C \cdot \Gamma)^{\mathrm{c}}$. From a completeness perspective, however, a different reading of constrained clauses is appropriate. The clause part C of a (ground) constrained clause $C \cdot \Gamma$ is evaluated in an E-interpretation I, whereas the literals in Γ are evaluated wrt. a context Λ in terms of productivity. The following definition makes this precise.

We say that a ground constraint Γ *consists of split rewrite literals* iff for every $l \to r \in \Gamma$ and for every $l \not\to r \in \Gamma$, it holds that $l \approx r$ is a split atom and $l \succ r$.

Definition 2 (Satisfaction of Constraints) Let Λ be a context, Γ a constraint and γ a ground substitution for Γ. We say that Λ *satisfies* (Γ, γ) and write $\Lambda \models (\Gamma, \gamma)$ iff

(i) $\Gamma\gamma$ consists of split rewrite literals,
(ii) Λ produces Γ and Λ produces $\Gamma\gamma$ by the same literals, that is, for each $L \in \Gamma$, there is a literal $K \in \Lambda$ that produces both L in Λ and $L\gamma$ in Λ. □

In the introduction we mentioned that ground constrained clauses whose constraint is not satisfied by a context can be ignored. Definition 2 makes that informal notion of satisfaction precise, in a stronger form. Intuitively, contexts need to define interpretations on split atoms only, via (a subset of) their produced ground (and ordered) split rewrite rules, which motivates condition (i). Condition (ii) says that each rewrite literal and its considered ground instance must be produced by the same context literal. For example, this is *not* the case when $\Lambda = \{\mathsf{P}(\mathsf{a})\}$ and $\Gamma = \{\mathsf{P}(x)\}$, where, e.g., Γ stems from the constrained empty clause $\Box \cdot \mathsf{P}(x)$. The literal $\mathsf{P}(\mathsf{a}) \in \Lambda$ makes $\mathsf{P}(\mathsf{a})$ true in the induced interpretation, and so the instance $\Box \cdot \mathsf{P}(\mathsf{a})$ is a candidate to indicate a contradiction. However, with condition (ii) it holds $\Lambda \not\models (\Gamma, \{x \mapsto a\})$, because Λ does not produce $\mathsf{P}(x)$ (at all), and so the constrained clause $\Box \cdot \mathsf{P}(x)$ can be ignored. Intuitively, the context literal $\mathsf{P}(\mathsf{a})$ is not responsible to falsify $\Box \cdot \mathsf{P}(x)$. On the other side, because input clauses must come with empty constraints, $\Box \cdot \mathsf{P}(x)$ can be obtained only from the clause $\neg\mathsf{P}(x) \cdot \emptyset$ (by U-Res). But then, U-Sup is applicable to $\mathsf{P}(\mathsf{a})$ and $\neg\mathsf{P}(x) \cdot \emptyset$, giving $\Box \cdot \mathsf{P}(\mathsf{a})$. Now we have $\Lambda \models (\mathsf{P}(\mathsf{a}), \emptyset)$, and hence $\Box \cdot \mathsf{P}(\mathsf{a})$ cannot be ignored. Indeed, Close is applicable now.

Definition 3 (Satisfaction of Constrained Clauses) Let Λ be a context, I an E-interpretation $(C \cdot \Gamma; \gamma)$ a ground closure. We say that the pair (Λ, I) *satisfies* $(C \cdot \Gamma; \gamma)$ and write $(\Lambda, I) \models (C \cdot \Gamma; \gamma)$ iff $\Lambda \not\models (\Gamma, \gamma)$ or $I \models C\gamma$. □

Definition 3 also applied to pairs (Λ, R), where R is a rewrite system, by implicitly taking $(\Lambda, R^\star)$. Indeed, in the main applications of Definition 3 such a rewrite system R will be determined by the model construction in Section 7 below.

We need some generalisations of Definition 3. We say that the pair (Λ, I) *satisfies* a set F of ground closures, written as $(\Lambda, I) \models F$ iff (Λ, I) satisfies all elements in F. If F and G are set of ground closures, we say that F *entails* G *wrt.* Λ, and write $F \models_\Lambda G$, iff for every E-interpretation I it holds $(\Lambda, I) \not\models F$ or $(\Lambda, I) \models G$. These generalisations also apply to (sets of) possibly non-ground constrained clauses by taking the (union of the) sets of all their ground closures.

Example 4 Let $\Lambda = \{\mathsf{f}(x) \to x, \mathsf{f}(\mathsf{c}) \not\to \mathsf{c}\}$, $R = \{\mathsf{f}(\mathsf{a}) \to \mathsf{a}, \mathsf{f}(\mathsf{b}) \to \mathsf{b}\}$ and $C \cdot \Gamma = \mathsf{f}(\mathsf{f}(\mathsf{a})) \approx x \cdot \mathsf{f}(x) \to x$. Let $\gamma_\mathsf{a} = \{x \mapsto \mathsf{a}\}$, $\gamma_\mathsf{b} = \{x \mapsto \mathsf{b}\}$ and $\gamma_\mathsf{c} = \{x \mapsto \mathsf{c}\}$. Suppose that all (ground) atoms are split atoms. Notice that $\Gamma\gamma_\mathsf{a}$, $\Gamma\gamma_\mathsf{b}$ and $\Gamma\gamma_\mathsf{c}$ consist of split rewrite literals. Then, $\Lambda \models (\Gamma, \gamma_\mathsf{a})$, as Λ produces $\{\mathsf{f}(x) \to x\}$ and $\{\mathsf{f}(\mathsf{a}) \to \mathsf{a}\}$ by the same literals. So we need to check $R^\star \models \mathsf{f}(\mathsf{f}(\mathsf{a})) \approx \mathsf{a}$, which is the case, to conclude $(\Lambda, R) \models (C \cdot \Gamma; \gamma_\mathsf{a})$. As $\Lambda \models (\Gamma, \gamma_\mathsf{b})$ but $R^\star \not\models \mathsf{f}(\mathsf{f}(\mathsf{a})) \approx \mathsf{b}$ we have $(\Lambda, R) \not\models (C \cdot \Gamma; \gamma_\mathsf{b})$. Finally, Λ does not produce $\{\mathsf{f}(\mathsf{c}) \to \mathsf{c}\}$, and so $\Lambda \not\models (\Gamma, \gamma_\mathsf{c})$. It follows $(\Lambda, R) \models (C \cdot \Gamma; \gamma_\mathsf{c})$. □

Constraints are compared in a similar way as clauses by taking a positive rewrite literal $l \to r$ as the multiset $\{l, r\}$, a negative literal as the multiset $l \not\to r$ as the multiset $\{l, l, r, r\}$, and using the two-fold multiset extension of the ordering $\succ$ on terms.[7]. Constrained clauses then are compared lexicographically, using first the clause ordering introduced earlier to compare the clause components, and then using the ordering on constraints. Again we use the symbol $\succ$ to denote this (strict) ordering on constrained clauses. It follows with well-known results that $\succ$ is a liftable, well-founded and strict ordering on ground constrained clauses. Observe that this definition has the desirable property that proper subsumption among ground constrained clauses is always order-decreasing (the subsuming constrained clause is smaller).

[7] Alternative definitions are possible. For instance, there is no need to make negative rewrite literal greater than its positive complement.

In order to obtain a total and well-founded ordering on ground closures, we combine the ordering on constrained clauses lexicographically with an arbitrary ordering $\succ'$ on ground closures that is total (up to variable renaming),[8] that is, we define $(C \cdot \Gamma; \gamma) \succ (C' \cdot \Gamma'; \gamma')$ iff $(C \cdot \Gamma)\gamma \succ (C' \cdot \Gamma')\gamma'$ or $(C \cdot \Gamma)\gamma = (C' \cdot \Gamma')\gamma'$ and $(C \cdot \Gamma; \gamma) \succ' (C' \cdot \Gamma'; \gamma')$.

5 Inference Rules on Constrained Clauses

In the following, we introduce a number of inference rules on rewrite literals and constrained clauses that will be used for defining the derivation rules of $\mathcal{ME}$+Sup. Within the calculus, which operates on sequents consisting of a context and constrained clause set, the rewrite literal will come from the current context and the clause from the current constrained clause set.

$$\textsf{Ref}\ \frac{s \not\approx t \vee C \cdot \Gamma}{(C \cdot \Gamma)\sigma}$$

where (i) σ is an mgu of s and t, and (ii) $(s \not\approx t)\sigma$ is maximal in $(s \not\approx t \vee C)\sigma$.

The next three rules combine a rewrite literal and a constrained clause.

$$\textsf{U-Sup-Neg}\ \frac{l \to r \qquad s[u]_p \not\approx t \vee C \cdot \Gamma}{(s[r]_p \not\approx t \vee C \cdot (\Gamma, l \to r))\sigma}$$

where (i) σ is an mgu of l and u, (ii) u is not a variable, (iii) $(l \approx r)\sigma$ is a split atom, (iv) $r\sigma \not\succeq l\sigma$, (v) $t\sigma \not\succeq s\sigma$, and (vi) $(s \not\approx t)\sigma$ is maximal in $(s \not\approx t \vee C)\sigma$.

$$\textsf{U-Sup-Pos}\ \frac{l \to r \qquad s[u]_p \approx t \vee C \cdot \Gamma}{(s[r]_p \approx t \vee C \cdot (\Gamma, l \to r))\sigma}$$

where (i) σ is an mgu of l and u, (ii) u is not a variable, (iii) $(l \approx r)\sigma$ is a split atom, (iv) $r\sigma \not\succeq l\sigma$, and if $(s \approx t)\sigma$ is a split atom then (v-a) $(s \approx t)\sigma$ is maximal in $(s \approx t \vee C)\sigma$ else (v-b) $t\sigma \not\succeq s\sigma$ and $(s \approx t)\sigma$ is strictly maximal in $(s \approx t \vee C)\sigma$, and (vi) if $l\sigma = s\sigma$ then $r\sigma \not\succeq t\sigma$.

U-Sup-Pos and U-Sup-Neg are the only rules that create new rewrite literals $(l \to r)\sigma$ in the constraint part (Sup-Neg and Sup-Pos below only merge existing constraints). Notice that because u is not a variable, in both cases $l\sigma$ is not a variable, even if l is. It follows easily that all expressions $C \cdot \Gamma$ derivable by the calculus are constrained clauses.

$$\textsf{U-Res}\ \frac{\neg A \qquad s \approx t \vee C \cdot \Gamma}{(C \cdot (\Gamma, s \not\to t))\sigma}$$

where $\neg A$ is a pseudo literal $\neg x$ or a negative rewrite literal $l \not\to r$, and (i) σ is an mgu of A and $s \to t$, (ii) $(s \approx t)\sigma$ is a split atom, (iii) $t\sigma \not\succeq s\sigma$, and (iv) $(s \approx t)\sigma$ is maximal in $(s \approx t \vee C)\sigma$.

The following three rules are intended to be applied to clauses from a current clause set. To formulate them we need one more definition: let $l \approx r$ be an equation and $x_1 \approx t_1 \vee \cdots \vee x_n \approx t_n$ a (possibly empty) clause of positive literals, where x_i is a variable and t_i a

[8] Since for every ground closure $(C \cdot \Gamma; \gamma)$ there are only finitely many closures $(C' \cdot \Gamma'; \gamma')$ such that $(C \cdot \Gamma)\gamma = (C' \cdot \Gamma')\gamma'$, the lexicographic combination is well-founded even if $\succ'$ is not well-founded.

term, for all $i = 1, \ldots, n$. We say that a substitution π *merges* $x_1 \approx t_1 \vee \cdots \vee x_n \approx t_n$ *with* $l \approx r$ iff π is an mgu of $\{l, x_1, \ldots, x_n\}$ and $r\pi \not\succeq l\pi$, and $t_i\pi \not\succeq l\pi$.

$$\textsf{Sup-Neg}\ \frac{l \approx r \vee C' \cdot \Gamma' \qquad s[u]_p \not\approx t \vee C \cdot \Gamma}{(s[r]_p \not\approx t \vee C \vee C' \cdot (\Gamma, \Gamma'))\sigma\pi}$$

where (i) σ is an mgu of l and u, (ii) u is not a variable, (iii) π merges $x_1 \approx t_1 \vee \cdots \vee x_n \approx t_n \subseteq C'\sigma$ with $(l \approx r)\sigma$, (iv) $\{x_1, \ldots, x_n\} \subseteq \mathit{Var}(\Gamma'\sigma)$, (v) $(l \approx r)\sigma\pi$ is a superposition atom, (vi) $r\sigma\pi \not\succeq l\sigma\pi$, (vii) $(l \approx r)\sigma\pi$ is strictly maximal in $(l \approx r \vee C')\sigma\pi$, (viii) $t\sigma\pi \not\succeq s\sigma\pi$, and (ix) $(s \not\approx t)\sigma\pi$ is maximal in $(s \not\approx t \vee C)\sigma\pi$.

The disequation $(s \not\approx t)\sigma\pi$ paramodulated into can be a split atom or a superposition atom. The need for merge substitutions is demonstrated in Example 9 below.

$$\textsf{Sup-Pos}\ \frac{l \approx r \vee C' \cdot \Gamma' \qquad s[u]_p \approx t \vee C \cdot \Gamma}{(s[r]_p \approx t \vee C \vee C' \cdot (\Gamma, \Gamma'))\sigma\pi}$$

where (i) σ is an mgu of l and u, (ii) u is not a variable, (iii) π merges $x_1 \approx t_1 \vee \cdots \vee x_n \approx t_n \subseteq C'\sigma$ with $(l \approx r)\sigma$, (iv) $\{x_1, \ldots, x_n\} \subseteq \mathit{Var}(\Gamma'\sigma)$, (v) $(l \approx r)\sigma\pi$ is a superposition atom, (vi) $r\sigma\pi \not\succeq l\sigma\pi$, (vii) $(l \approx r)\sigma\pi$ is strictly maximal in $(l \approx r \vee C')\sigma\pi$, and if $(s \approx t)\sigma\pi$ is a split atom then (viii-a) $(s \approx t)\sigma\pi$ is maximal in $(s \approx t \vee C)\sigma\pi$ else (viii-b) $t\sigma\pi \not\succeq s\sigma\pi$ and $(s \approx t)\sigma\pi$ is strictly maximal in $(s \approx t \vee C)\sigma\pi$.

Notice that $(s \approx t)\sigma\pi$ could be both a split atom and a superposition atom. In this case the weaker condition (viii-a) is used to take care of a ground instance of a Sup-Pos inference applied to a split atom, which requires the weaker condition.

In both Sup-Neg and Sup-Pos inference rules we assume the additional condition $\mathscr{C}\sigma\pi \not\succeq \mathscr{D}\sigma\pi$, where by $\mathscr{C}$ and $\mathscr{D}$ we mean their left and right premise, respectively.

Finally, we need an equality factoring rule as in the superposition calculus:

$$\textsf{Eq-Fact}\ \frac{l \approx r \vee s \approx t \vee C \cdot \Gamma}{(l \approx t \vee r \not\approx t \vee C \cdot \Gamma)\sigma}$$

where (i) σ is an mgu of l and s, (ii) $(l \approx r)\sigma$ is a superposition atom, (iii) $(l \approx r)\sigma$ is maximal in $(l \approx r \vee s \approx t \vee C)\sigma$, (iv) $r\sigma \not\succeq l\sigma$, and (v) $t\sigma \not\succeq s\sigma$.

In each of the inference rules above we assume the additional condition that $\Gamma\sigma$ ($\Gamma\sigma\pi$ and $\Gamma'\sigma\pi$ in case of Sup-Neg or Sup-Pos) consists of split rewrite literals.

An *inference system* ι is a set of inference rules. By an ι *inference* we mean an instance of an inference rule from ι such that all conditions are satisfied. An inference is *ground* if all its premises and the conclusion are ground.

The *base inference system* ι_{Base} consists of Ref, Eq-Fact, U-Sup-Neg, U-Sup-Pos, U-Res, Sup-Neg and Sup-Pos. If from a given ι_{Base} inference a ground ι_{Base} inference results by applying a substitution γ to all premises and the conclusion, we call the resulting ground inference a *ground instance via* γ *(of the* ι_{Base} *inference)*. This is not always the case, as, e.g., ordering constraints can become unsatisfiable after application of γ. An important consequence of the ordering restrictions stated with the inference rules is that the conclusion of a ground ι_{Base} inference is always strictly smaller than the right or only premise.

6 Inference Rules on Sequents

Sequents are the main objects manipulated by the $\mathcal{ME}$+Sup calculus. A *sequent* is a pair $\Lambda \vdash \Phi$ where Λ is a context and Φ is a set of constrained clauses. The following inference rules extend the inference rules ι_{Base} above to sequents.

$$\textsf{Deduce}\ \frac{\Lambda \vdash \Phi}{\Lambda \vdash \Phi, C \cdot \Gamma}$$

if one of the following cases applies:

- $C \cdot \Gamma$ is the conclusion of a Ref or Eq-Fact inference with a premise from Φ.
- $C \cdot \Gamma$ is the conclusion of a U-Sup-Neg, U-Sup-Pos or U-Res inference with a right premise from Φ and a left premise from Λ.
- $C \cdot \Gamma$ is the conclusion of a Sup-Neg or Sup-Pos inference with both premises from Φ.

and if additionally Λ produces Γ.[9]

In each case the second or only premise of the underlying ι_{Base} inference is called the *selected clause (of a* Deduce *inference)*. In inferences involving two premises, a fresh variant of the, say, right premise is taken, so that the two premises are variable disjoint.

$$\textsf{Split}\ \frac{\Lambda \vdash \Phi}{\Lambda, \overline{K} \vdash \Phi \qquad \Lambda, K \vdash \Phi}$$

if there is a constrained clause $\square \cdot \Gamma \in \Phi$ such that (i) $K \in \Gamma$, (ii) $s \approx t$ is a split atom, where $K = s \to t$ or $K = s \not\to t$, and (iii) neither K nor $\overline{K}$ is contradictory with Λ. A Split inference is *productive* if Λ produces Γ; the clause $\square \cdot \Gamma$ is called the *selected clause (of the* Split *inference)*.

The intuition behind Split is to make a constrained empty clause $\square \cdot \Gamma$ true, which is false when Λ produces Γ (in the sense of Definition 3). This is achieved by adding $\overline{K}$ to the current context. For example, if $\Lambda = \{\mathsf{P}(\mathsf{a}, y), \neg\mathsf{P}(x, \mathsf{b})\}$ and $\square \cdot \Gamma = \square \cdot \mathsf{P}(\mathsf{a}, \mathsf{b})$ then a (productive) Split inference will give $\{\mathsf{P}(\mathsf{a}, y), \neg\mathsf{P}(x, \mathsf{b}), \neg\mathsf{P}(\mathsf{a}, \mathsf{b})\}$, which no longer produces $\mathsf{P}(\mathsf{a}, \mathsf{b})$. Intuitively, the calculus tries to "repair" the current context towards a model for $\square \cdot \Gamma$.

Notice that a Split inference can never add a rewrite literal to a context that already contains a variant of it or its complement, as this would contradict condition (iii).[10] Because of the latter property the calculus will never derive contradictory contexts.

$$\textsf{Close}\ \frac{\Lambda \vdash \Phi}{\Lambda \vdash \Phi, \square \cdot \emptyset}$$

if there is a constrained clause $\square \cdot \Gamma \in \Phi$ such that $L \in_\sim \Lambda$ for every $L \in \Gamma$. The clause $\square \cdot \Gamma$ is called the *selected clause (of a* Close *inference)* and the variants of the L's in Λ are the *closing literals*. A sequent $\Lambda \vdash \Phi$ is *closed* if Φ contains $\square \cdot \emptyset$. The purpose of Close is to abandon a sequent that cannot be "repaired".

The $\iota_{\mathcal{ME}+\textsf{Sup}}$ inference system consists of the rules Deduce, Split and Close.

[9] or equivalently: if Λ produces every constraint of the instantiated constrained clauses that are premises of the underlying ι_{Base} inference, and Λ produces the instantiated left premises in case of U-Sup-Neg, U-Sup-Pos or Neg-Res, where these instances are obtained by applying the substitution used in that ι_{Base} inference.

[10] The Deduce rule could be strengthened to exclude adding variants to the clause sets in the conclusion. We ignore this (trivial) aspect.

In the introduction we mentioned that the $\mathcal{ME}$+Sup calculus can be configured to obtain a pure Superposition or a pure Model Evolution calculus (with equality). For the former, every ground atom is to be labelled as a superposition atom. Then, the only inference rules in effect are Ref, Sup-Neg, Sup-Pos and Eq-Fact, all of which are standard inference rules of the Superposition calculus. Furthermore, under the reasonable assumption that the input clauses are constraint-free, all derivable contexts will be $\{\neg x\}$, and also the constraints in all derivable clauses will be empty. In consequence, not even Close is applicable (unless the clause set in the premise already contains $\Box \cdot \emptyset$). In contrast, if all atoms are labelled as split atoms, then the only inference rules in effect are Ref,U-Sup-Neg, U-Sup-Pos, U-Res, Split and Close. The resulting calculus is similar to the $\mathcal{ME}_{\mathrm{E}}$ calculus [BT05] but not quite the same. On the one hand, $\mathcal{ME}_{\mathrm{E}}$ features *universal variables*, a practically important improvement, which $\mathcal{ME}$+Sup does not (yet) have. (Context literals containing "universal variables" are similar to unit *clauses*, i.e., they stand for all their instances, unconditionally). On the other hand, $\mathcal{ME}_{\mathrm{E}}$ needs to compute additional unifiers, for instance in the counterpart to the Close rule, which are not necessary in $\mathcal{ME}$+Sup.

7 Model Construction

To obtain the completeness result for $\mathcal{ME}$+Sup we associate to a sequent $\Lambda \vdash \Phi$ a convergent left-reduced rewrite system $R_{\Lambda \vdash \Phi}$. The general technique is taken from the completeness proof of the Superposition calculus [BG98, NR95] and adapted to our needs. One difference is that $\mathcal{ME}$+Sup requires the construction of a fully reduced rewrite system for its split atoms, whereas for the superposition atoms a left-reduced rewrite system is sufficient. Another difference is that certain aspects of lifting must be reflected already for the model generation. For the latter, we need a preliminary definition.

Definition 5 (Relevant Closure wrt. (Λ, R)**)** Let Λ be a context, R a rewrite system, and γ a ground substitution for a constrained clause $C \cdot \Gamma$. We say that $(C \cdot \Gamma; \gamma)$ is a *relevant (ground) closure wrt.* (Λ, R) iff

(i) $\Lambda \models (\Gamma, \gamma)$, and
(ii) $(\mathit{Var}(C) \cap \mathit{Var}(\Gamma))\gamma$ is irreducible wrt. R. □

Notice that for a clause with an empty constraint all its ground closures are relevant.

Example 6 If $\Lambda = \{\mathsf{P}(x), \mathsf{a} \to \mathsf{b}, \neg\mathsf{P}(\mathsf{b})\}$, $R = \{\mathsf{a} \to \mathsf{b}\}$ and $C \cdot \Gamma = x \approx \mathsf{b} \vee x \approx \mathsf{d} \cdot \mathsf{P}(x)$ then the substitution $\gamma = \{x \mapsto \mathsf{a}\}$ gives a ground closure $(C \cdot \Gamma; \gamma)$ that satisfies condition (i) but not (ii). With the substitution $\gamma = \{x \mapsto \mathsf{c}\}$ both (i) and (ii) are satisfied, and with $\gamma = \{x \mapsto \mathsf{b}\}$ the condition (i) is not satisfied but (ii) is. If $\Lambda = \{\mathsf{P}(\mathsf{a})\}$ then $(\Box \cdot \mathsf{P}(x); \gamma)$ is never a relevant closure, for any γ, because Λ does not produce $\mathsf{P}(x)$, and hence $\Lambda \not\models (\Box \cdot \mathsf{P}(x); \gamma)$. □

For a given sequent $\Lambda \vdash \Phi$, where Φ does not contain $\Box \cdot \emptyset$, we define by induction on the closure ordering $\succ$ sets of rewrite rules $\mathcal{E}_{\mathscr{C}}$ and $R_{\mathscr{C}}$, for every $\mathscr{C} \in \Phi^{\mathrm{gr}} \cup \Pi_\Lambda$. Here, for the purpose of comparing elements from $\Phi^{\mathrm{gr}} \cup \Pi_\Lambda$, a rewrite literal $l \to r \in \Pi_\Lambda$ is taken as the (ground) closure $((l \approx r \cdot \bot); \emptyset)$, where $\bot$ is a fresh symbol that is considered smaller than the empty multiset. This way, $\succ$ is a total ordering on $\Phi^{\mathrm{gr}} \cup \Pi_\Lambda$. For instance $((l \approx r \cdot \emptyset); \emptyset) \succ l \to r$ as $((l \approx r \cdot \emptyset); \emptyset) \succ ((l \approx r \cdot \bot); \emptyset)$, as $\emptyset \succ \bot$.

Assume that $\mathcal{E}_{\mathscr{D}}$ has already been defined for all $\mathscr{D} \in \Phi^{\mathrm{gr}} \cup \Pi_\Lambda$ with $\mathscr{C} \succ \mathscr{D}$ and let $R_{\mathscr{C}} = \bigcup_{\mathscr{C} \succ \mathscr{D}} \mathcal{E}_{\mathscr{D}}$. The set $\mathcal{E}_{\mathscr{C}}$ is defined differently depending on the type of $\mathscr{C}$. If $\mathscr{C}$ is rewrite literal $l \to r \in \Pi_\Lambda$ then let $\mathcal{E}_{l \to r} = \{l \to r\}$ if

1. $l \approx r$ is a split atom,
2. $l \succ r$, and
3. l and r are irreducible wrt. $R_{l \to r}$.

Otherwise $\varepsilon_{l \to r} = \emptyset$. If $\mathscr{C}$ is a ground closure $(C \cdot \Gamma; \gamma) \in \Phi^{\mathrm{gr}}$ then let $\varepsilon_{(C \cdot \Gamma; \gamma)} = \{s \to t\}$ if

1. $C\gamma = s \approx t \vee D$,
2. $s \approx t$ is strictly maximal in $C\gamma$,
3. $s \approx t$ is a superposition atom,
4. $s \succ t$,
5. $(C \cdot \Gamma; \gamma)$ is a relevant closure wrt. $(\Lambda, R_{(C \cdot \Gamma; \gamma)})$,
6. $R^{\star}_{(C \cdot \Gamma; \gamma)} \not\models C\gamma$,
7. $(R_{(C \cdot \Gamma; \gamma)} \cup \{s \to t\})^{\star} \not\models D$, and
8. s is irreducible wrt. $R_{(C \cdot \Gamma; \gamma)}$.

Otherwise $\varepsilon_{(C \cdot \Gamma; \gamma)} = \emptyset$.

Finally, $R = \bigcup_{\mathscr{C}} \varepsilon_{\mathscr{C}}$. If $\varepsilon_{l \to r} = \{l \to r\}$ then we say that $l \to r$ *generates* $l \to r$ in R. If $\varepsilon_{(C \cdot \Gamma; \gamma)} = \{l \to r\}$ then we say that $(C \cdot \Gamma; \gamma)$ *generates* $l \to r$ in R. Often we write $R_{\Lambda \vdash \Phi}$ instead of R to make clear that R is constructed from $\Phi^{\mathrm{gr}} \cup \Pi_{\Lambda}$.

It is easy to see that R is a left-reduced rewrite system and the rules contributed by Π_{Λ} are even fully reduced wrt. R. Since $\succ$ is a well-founded ordering, R is a convergent rewrite system. With well-known results it follows that satisfaction of ground literals $s \approx t$ (or $s \not\approx t$) in $R^{\star}$ can be decided by checking if the normal forms of s and t wrt. R are the same.

Notice that the evaluation of condition 5 for $\varepsilon_{(C \cdot \Gamma; \gamma)}$ refers to the context Λ, which is fixed prior to the model construction, and the rewrite system $R_{(C \cdot \Gamma; \gamma)}$ constructed so far. The definition can be seen to work in a hierarchical way, by, roughly, first building the set of those ground closures from Φ^{gr} whose constraints consists of rewrite literals and all are produced in Λ, and then generating R from that set, which involves checking irreducibility of substitutions wrt. $R_{(C \cdot \Gamma; \gamma)}$.

With respect to split atoms, the (completeness proof of the) calculus needs to consider those that are true because they are generated, and those that are false and irreducible. The following lemma provides a handle in such cases in terms of the "syntactic" concept of productivity.

Lemma 7 *Let $\mathscr{D} \in \Phi^{\mathrm{gr}} \cup \Pi_{\Lambda}$ and $l \approx r$ a ground split atom such that $l \succ r$. Then*

(i) if $l \to r \in R$ then Λ produces $l \to r$, and
(ii) if $\mathscr{D} \succ l \to r$, and l and r are irreducible wrt. $R_{\mathscr{D}}$ then Λ produces $l \not\to r$.

Proof Concerning (i), if $l \to r \in R$ then this is because $l \to r \in \Pi_{\Lambda}$ generates $l \to r$, and then Λ produces $l \to r$ by definition of Π_{Λ}.

Concerning (ii), suppose that l and r are irreducible wrt. $R_{\mathscr{D}}$. If $l \to r$ were contained in Π_{Λ}, then either $l \to r$ would be generated in $R_{\mathscr{D}}$, but then l would be reducible by $l \to r \in R_{\mathscr{D}}$, or $l \to r$ would not be generated in $R_{\mathscr{D}}$, but this is only possible if l or r is reducible by $R_{l \to r}$, and since $R_{l \to r} \subseteq R_{\mathscr{D}}$, it would again be reducible by $R_{\mathscr{D}}$. Hence $l \to r \notin \Pi_{\Lambda}$. Thanks to the presence of the pseudo-literal $\neg x$ in every context, it is not difficult to see that every context produces K or $\overline{K}$, for every literal K. Thus, with Λ not producing $l \to r$ conclude that Λ produces $l \not\to r$. □

Example 8 Let $\Lambda = \{\mathsf{a} \to x, \mathsf{b} \to \mathsf{c}, \mathsf{a} \not\to \mathsf{c}\}$, $\Phi = \emptyset$ and assume that all equations are split atoms. With $\mathsf{a} \succ \mathsf{b} \succ \mathsf{c}$ the induced rewrite system R is $\{\mathsf{b} \to \mathsf{c}\}$. To see why, observe that the rule $\mathsf{a} \to \mathsf{c}$ is not included in R, as Λ does not produce $\mathsf{a} \to \mathsf{c}$, and that $\mathsf{a} \to \mathsf{b}$, although

produced in Λ, is reducible by the smaller rule $\mathsf{b} \to \mathsf{c}$. Had we chosen to omit in the definition of $\varepsilon_{l \to r}$ the condition "r is irreducible wrt. $R_{l \to r}$"[11] the construction would have given $R = \{\mathsf{a} \to \mathsf{b}, \mathsf{b} \to \mathsf{c}\}$. This leads to the undesirable situation that a constrained clause, say, $\mathsf{a} \not\approx \mathsf{c} \cdot \emptyset$ is falsified by $R^\star$. But the calculus cannot modify Λ to revert this situation, and to detect the inconsistency (ordered) paramodulation into variables would be needed. □

Example 9 Let $\mathsf{a} \succ \mathsf{b} \succ \mathsf{c}$, $\Lambda = \{\mathsf{P}(x), \neg\mathsf{P}(\mathsf{b}), \neg\mathsf{P}(\mathsf{c})\}$ and $C \cdot \Gamma = y \approx \mathsf{b} \vee x \approx \mathsf{c} \cdot \mathsf{P}(x)$ be the only clause in Φ. Then the ground closure $(C \cdot \Gamma; \gamma) = (y \approx \mathsf{b} \vee x \approx \mathsf{c} \cdot \mathsf{P}(x); \{x \mapsto \mathsf{a}, y \mapsto \mathsf{a}\})$ generates $\mathsf{a} \to \mathsf{b}$ in R. This is, because it is a relevant closure wrt. $(\Lambda, R_{(C \cdot \Gamma; \gamma)}) = (\Lambda, \emptyset)$.

Now, a ground instance, say, via γ, of an inference with $C \cdot \Gamma$ as the left premise and a right premise, say, $C' \cdot \Gamma'$ will possibly not preserve relevancy of $(C' \cdot \Gamma'; \gamma)$ (wrt. $(C' \cdot \Gamma'; \gamma)$). This may happen because the conclusion can be bigger than the left premise (even if the right premise is bigger than the left premise, which is safe to assume), and so $x\gamma$ could be reducible wrt. $R_{(C' \cdot \Gamma'; \gamma)}$ *by the rule generated by the left premise*. For instance, if the right premise is $\mathsf{f}(\mathsf{a}) \not\approx \mathsf{f}(\mathsf{b}) \cdot \emptyset$ then a Sup-Neg inference yields $\mathsf{f}(\mathsf{b}) \not\approx \mathsf{f}(\mathsf{b}) \vee x \approx \mathsf{c} \cdot \mathsf{P}(x)$. But this is not a relevant closure wrt. $(\Lambda, R_{(C' \cdot \Gamma'; \gamma)})$, because $x\gamma = \mathsf{a}$ is reducible wrt. $R_{(C' \cdot \Gamma'; \gamma)} = \{\mathsf{a} \to \mathsf{b}\}$. This is a problem from the completeness perspective, because the calculus needs to reduce relevant closures of clauses that are false (in a certain interpretation) to smaller *relevant* closures. The suggested Sup-Neg step would thus not work in this case. The problem is avoided by a different Sup-Neg inference, one with a non-empty merge substitution:

$$\textsf{Sup-Neg}\ \frac{y \approx \mathsf{b} \vee x \approx \mathsf{c} \cdot \mathsf{P}(x) \qquad \mathsf{f}(\mathsf{a}) \not\approx \mathsf{f}(\mathsf{b}) \cdot \emptyset}{\mathsf{f}(\mathsf{b}) \not\approx \mathsf{f}(\mathsf{b}) \vee \mathsf{a} \approx \mathsf{c} \cdot \mathsf{P}(\mathsf{a})}$$

where $\sigma = \{y \mapsto \mathsf{a}\}$ and $\pi = \{x \mapsto \mathsf{a}\}$. Then, $(\mathsf{f}(\mathsf{b}) \not\approx \mathsf{f}(\mathsf{b}) \vee \mathsf{a} \approx \mathsf{c} \cdot \mathsf{P}(\mathsf{a}); \emptyset)$ is a relevant closure wrt. $(\Lambda, R_{(\mathsf{f}(\mathsf{b}) \not\approx \mathsf{f}(\mathsf{b}) \vee \mathsf{a} \approx \mathsf{c} \cdot \mathsf{P}(\mathsf{a}); \emptyset)})$. Situations like the one above are the *only* critical ones, and relevancy can always be preserved by a merge substitution. The following lemma provides a (slightly strengthened) formal account. □

Lemma 10 (ι_{Base} **Inferences Preserve Relevant Closures)** *Let $\Lambda \vdash \Phi$ be a sequent and assume an ι_{Base} inference with right (or only) premise $C \cdot \Gamma$, conclusion $C' \cdot \Gamma'$, and a ground instance via γ of the ι_{Base} inference such that*

- *(i) $(C \cdot \Gamma; \gamma)$ is a relevant closure wrt. $(\Lambda, R_{(C \cdot \Gamma; \gamma)})$,*
- *(ii-a) in case of* Sup-Neg *or* Sup-Pos*, the ground closure $(l \approx r \vee C'' \cdot \Gamma''; \gamma)$ generates the rule $(l \to r)\gamma$ in $R_{\Lambda \vdash \Phi}$, where $l \approx r \vee C'' \cdot \Gamma''$ is the left premise of the non-ground inference,*
- *(ii-b) in case of* U-Sup-Neg *or* U-Sup-Pos*, the left premise $(l \to r)\gamma$ generates the rule $(l \to r)\gamma$ in $R_{\Lambda \vdash \Phi}$, and*
- *(ii-c) in case of* U-Res*, the left premise is $\neg A\gamma = (s \not\to t)\gamma$ and $s\gamma$ and $t\gamma$ are irreducible wrt. $R_{(C \cdot \Gamma; \gamma)}$.*

Then, $(C' \cdot \Gamma'; \gamma)$ is a relevant closure wrt. $(\Lambda, R_{(C \cdot \Gamma; \gamma)})$, for some possibly different merge substitution in case of a Sup-Neg *or* Sup-Pos *inference.*

A proof is in the appendix.

We conclude this section with important monotonicity results of the model construction.

[11] This condition is absent in the model construction for superposition atoms. Its presence explains why paramodulation into smaller sides of positive split literals in clauses is necessary.

Lemma 11 *If $(s \approx t \vee D \cdot \Gamma; \gamma)$ generates $(s \to t)\gamma$ in R then $R^\star \models (s \approx t)\gamma$ and $R^\star \not\models D\gamma$, and $R^\star_{\mathscr{D}} \models (s \approx t)\gamma$ and $R^\star_{\mathscr{D}} \not\models D\gamma$ for every $\mathscr{D} \in \Phi^{\mathrm{gr}} \cup \Pi_\Lambda$ such that $\mathscr{D} \succ (s \approx t \vee D \cdot \Gamma; \gamma)$.*

Proof It can be shown that the rewrite system R is left-reduced and ordered, furthermore the left hand side of every rewrite rule in $R \setminus (R_{(s \approx t \vee D \cdot \Gamma; \gamma)} \cup \{(s \to t)\gamma\})$ is larger than every term occurring in a positive literal of $D\gamma$, hence these rules cannot be used to rewrite terms in positive literals of $D\gamma$. Therefore, if a literal of $D\gamma$ is false in $(R_{(s \approx t \vee D \cdot \Gamma; \gamma)} \cup \{(s \to t)\gamma\})^\star$, then it is false in $R^\star$. The same arguments apply for the second part of the lemma statement. For that, it suffices to observe that $R_{\mathscr{D}} \supseteq R_{(s \approx t \vee D \cdot \Gamma; \gamma)}$ as $\mathscr{D} \succ (s \approx t \vee D \cdot \Gamma; \gamma)$. □

Lemma 12 *If $R^\star_{(C \cdot \Gamma; \gamma)} \models C\gamma$ then $R^\star \models C\gamma$ and $R^\star_{\mathscr{D}} \models C\gamma$ for every $\mathscr{D} \in \Phi^{\mathrm{gr}} \cup \Pi_\Lambda$ such that $\mathscr{D} \succeq (C \cdot \Gamma; \gamma)$.*

Proof The left hand side of every rewrite rule in $R \setminus R_{(C \cdot \Gamma; \gamma)}$ is larger than every term occurring in a negative literal of $C\gamma$, hence these rules cannot be used to rewrite terms in negative literals of $C\gamma$. Therefore, if a literal of $C\gamma$ is true in $R^\star_{(C \cdot \Gamma; \gamma)}$, then it is true in $R^\star$. The same arguments apply for the second part of the lemma statement. For that, it suffices to observe that $R_{\mathscr{D}} \supseteq R_{(C \cdot \Gamma; \gamma)}$ as $\mathscr{D} \succeq (C \cdot \Gamma; \gamma)$. □

Corollary 13 *If $(\Lambda, R_{(C \cdot \Gamma; \gamma)}) \models (C \cdot \Gamma; \gamma)$ then $(\Lambda, R) \models (C \cdot \Gamma; \gamma)$ and $(\Lambda, R_{\mathscr{D}}) \models (C \cdot \Gamma; \gamma)$ for every $\mathscr{D} \in \Phi^{\mathrm{gr}} \cup \Pi_\Lambda$ such that $\mathscr{D} \succeq (C \cdot \Gamma; \gamma)$.*

Proof Suppose $(\Lambda, R_{(C \cdot \Gamma; \gamma)}) \models (C \cdot \Gamma; \gamma)$. If $\Lambda \not\models (\Gamma, \gamma)$ the claim is trivial. Hence suppose $\Lambda \models (\Gamma, \gamma)$. By definition, $R^\star_{(C \cdot \Gamma; \gamma)} \models C$. With Lemma 12 conclude $R^\star \models C\gamma$ and, trivially, $(\Lambda, R) \models (C \cdot \Gamma; \gamma)$. Similarly for the second part. □

8 Redundancy, Saturation and Static Completeness

To define concepts of redundancy we need a specific notion of relevant closures that takes the model construction into account. We extend Definition 5 and say that $(C \cdot \Gamma; \gamma)$ is a *relevant closure wrt. Λ* iff $(C \cdot \Gamma; \gamma)$ is a relevant closure wrt. $(\Lambda, R_{(C \cdot \Gamma; \gamma)})$. Relevancy of a closure $(C \cdot \Gamma; \gamma)$ wrt. Λ thus does not depend on rules from $R \setminus R_{(C \cdot \Gamma; \gamma)}$.

Definition 14 (Relevant Closures wrt. Λ) Let Φ be a set of constrained clauses. Define

$$\Phi^\Lambda = \{(C \cdot \Gamma; \gamma) \mid C \cdot \Gamma \in \Phi \text{ and } (C \cdot \Gamma; \gamma) \text{ is a relevant closure wrt. } \Lambda\} \ .$$

Let $\Lambda \vdash \Phi$ be a sequent and $\mathscr{D}$ a ground closure. Define

$$\Phi^\Lambda_{\mathscr{D}} = \{(C \cdot \Gamma; \gamma) \in \Phi^\Lambda \mid \mathscr{D} \succ (C \cdot \Gamma; \gamma)\}$$

as the set of relevant closures wrt. Λ of all constrained clauses from Φ that are all smaller wrt. $\succ$ than $\mathscr{D}$. □

Definition 15 (Redundant Ground Closure) Let $\Lambda \vdash \Phi$ be a sequent, and $(C \cdot \Gamma; \gamma)$ and $\mathscr{D}$ ground closures. We say that $(C \cdot \Gamma; \gamma)$ is *redundant wrt. $\Lambda \vdash \Phi$ and $\mathscr{D}$* iff $\Phi^\Lambda_{\mathscr{D}} \models_\Lambda (C \cdot \Gamma; \gamma)$, and we say that $(C \cdot \Gamma; \gamma)$ is *redundant wrt. $\Lambda \vdash \Phi$* iff $(C \cdot \Gamma; \gamma)$ is redundant wrt. $\Lambda \vdash \Phi$ and $\mathscr{D} = (C \cdot \Gamma; \gamma)$. □

In words, $(C\cdot\Gamma;\gamma)$ is redundant wrt. $\Lambda \vdash \Phi$ and $\mathscr{D}$ iff $(C\cdot\Gamma;\gamma)$ is entailed wrt. Λ by relevant closures wrt. Λ of clauses in Φ that are smaller than $\mathscr{D}$.

The following lemma provides a condition under which redundant ground closures are not generating. Because the completeness proof needs to consider situations only that satisfy the condition, redundant ground closures can never be generating then.

Lemma 16 *Let $\Lambda \vdash \Phi$ be a sequent and $(C\cdot\Gamma;\gamma)$ a ground closure of a clause $C\cdot\Gamma \in \Phi$. If $(C\cdot\Gamma;\gamma)$ is redundant wrt. $\Lambda \vdash \Phi$, and $(\Lambda, R_{(C\cdot\Gamma;\gamma)}) \models \Phi^{\Lambda}_{(C\cdot\Gamma;\gamma)}$ then $(C\cdot\Gamma;\gamma)$ does not generate a rewrite rule in $R_{\Lambda\vdash\Phi}$.*

Proof Suppose that $(C\cdot\Gamma;\gamma)$ is redundant wrt. $\Lambda \vdash \Phi$ and that $(\Lambda, R_{(C\cdot\Gamma;\gamma)}) \models \Phi^{\Lambda}_{(C\cdot\Gamma;\gamma)}$. By the definition of redundancy then $\Phi^{\Lambda}_{(C\cdot\Gamma;\gamma)} \models_{\Lambda} (C\cdot\Gamma;\gamma)$. With $(\Lambda, R_{(C\cdot\Gamma;\gamma)}) \models \Phi^{\Lambda}_{(C\cdot\Gamma;\gamma)}$ it follows $(\Lambda, R_{(C\cdot\Gamma;\gamma)}) \models (C\cdot\Gamma;\gamma)$.

If $(C\cdot\Gamma;\gamma)$ is not a relevant closure wrt. Λ, then by property 5 of the definition of model construction $(C\cdot\Gamma;\gamma)$ cannot generate a rewrite rule. Otherwise, by relevancy, Λ produces Γ and Λ produces $\Gamma\gamma$, and $\Gamma\gamma$ consists of split rewrite literals. In other words, $\Lambda \models (\Gamma,\gamma)$, and with $(\Lambda, R_{(C\cdot\Gamma;\gamma)}) \models (C\cdot\Gamma;\gamma)$ it follows $R^{\star}_{(C\cdot\Gamma;\gamma)} \models C\gamma$. But then, $(C\cdot\Gamma;\gamma)$ cannot generate a rewrite rule according to condition 6 in the definition of model construction. □

The notion of redundancy defined above is essential to prove completeness but difficult to exploit in practice (it rests on reducibility wrt. rewrite systems that are determined by the limit of a derivation only). The following, related definition avoids that.

For a context Λ let $\mathrm{grd}(\Lambda)$ denote the set of all ground literals in Λ.

Definition 17 (Universal Redundancy) Let $\Lambda \vdash \Phi$ be a sequent, and $(C\cdot\Gamma;\gamma)$ and $\mathscr{D}$ ground closures. We say that $(C\cdot\Gamma;\gamma)$ is *universally redundant wrt. $\Lambda \vdash \Phi$ and $\mathscr{D}$*, iff there exists an $L \in \Gamma$ such that $\overline{L}\gamma \in \mathrm{grd}(\Lambda)$, or there exist ground closures $(C_i\cdot\Gamma_i;\gamma_i)$ of constrained clauses $C_i\cdot\Gamma_i \in \Phi$ such that, for every i,

(i) if $L \in \Gamma_i$, then $L \in \mathrm{grd}(\Lambda)$ or there exists a $K \in \Gamma$ such that $L \sim K$ and $L\gamma_i = K\gamma$,
(ii) $\mathscr{D} \succ (C_i\cdot\Gamma_i;\gamma_i)$,
(iii) $C_1\gamma_1 \ldots C_n\gamma_n \models C\gamma$, and
(iv) if $x \in Var(C_i) \cap Var(\Gamma_i)$, then there exists a $y \in Var(C) \cap Var(\Gamma)$ such that $x\gamma_i = y\gamma$. □

Intuitively, universal redundancy of a closure ensures that, whenever the closure is relevant wrt. Λ or a possible extension of Λ, then it must be entailed by sufficiently small closures that are also relevant.

We say that $(C\cdot\Gamma;\gamma)$ is *universally redundant wrt. $\Lambda \vdash \Phi$*, iff $(C\cdot\Gamma;\gamma)$ is universally redundant wrt. $\Lambda \vdash \Phi$ and $\mathscr{D} = (C\cdot\Gamma;\gamma)$, and we say that $C\cdot\Gamma$ is *universally redundant wrt. $\Lambda \vdash \Phi$* iff $(C\cdot\Gamma;\gamma)$ is universally redundant wrt. $\Lambda \vdash \Phi$, for every ground closure $(C\cdot\Gamma;\gamma)$ of $C\cdot\Gamma$.

For instance, when A is a ground literal, any (possibly non-ground) clause of the form $C\cdot(A,\Gamma)$ is universally redundant wrt. every $\Lambda \vdash \Phi$ such that $\overline{A} \in \Lambda$. Dually, $C\cdot(A,\Gamma)$ is universally redundant wrt. every $\Lambda \vdash \Phi$ such that $A \in \Lambda$ and $C\cdot\Gamma \in \Phi$. Correspondingly, the simplification rule defined below can be used to delete $C\cdot(A,\Gamma)$ if $\overline{A} \in \Lambda$, and if $A \in \Lambda$ then $C\cdot(A,\Gamma)$ can be simplified to $C\cdot\Gamma$. This generalizes corresponding simplification rules by unit clauses in the propositional DPLL-procedure.

Observe that Definition 17 refers to a context Λ only be testing if *ground* rewrite literals are contained in it, a property that is preserved as Λ grows. We obtain the following result.

Lemma 18 *If $C \cdot \Gamma$ is universally redundant wrt. $\Lambda \vdash \Phi$, $\Lambda' \supseteq \Lambda$, and Φ' is obtained from Φ by deleting constrained clauses that are universally redundant wrt. $\Lambda \vdash \Phi$ and/or by adding arbitrary constrained clauses, then $C \cdot \Gamma$ is universally redundant wrt. $\Lambda' \vdash \Phi'$.*

Proof It is obvious from Def. 17 that a clause that is universally redundant wrt. $\Lambda \vdash \Phi$ remains universally redundant if arbitrary constrained clauses are added to Φ or if literals are added to Λ.

To prove that a clause that is universally redundant wrt. $\Lambda \vdash \Phi$ remains universally redundant if universally redundant clauses are deleted from Φ, it suffices to show that the clauses $C_i \cdot \Gamma_i \in \Phi$ in Definition 17 can always be chosen in such a way that they are not themselves universally redundant: Suppose that a ground closure $(C \cdot \Gamma; \gamma)$ is universally redundant wrt. $\Lambda \vdash \Phi$ and $\mathscr{D}$. If there exists an $L \in \Gamma$ such that $\overline{L}\gamma \in \mathrm{grd}(\Lambda)$, then deleting clauses from Φ does not change anything. Otherwise let $\{ (C_i \cdot \Gamma_i; \gamma_i) \mid 1 \leq i \leq n \}$ be a minimal set of ground closures of clauses in Φ (wrt. the multiset extension of the closure ordering) that satisfies the conditions of Definition 17. Suppose that one of the $(C_i \cdot \Gamma_i; \gamma_i)$, say $(C_1 \cdot \Gamma_1; \gamma_1)$, is universally redundant itself. Then either there exists an $L \in \Gamma_1$ such that $\overline{L}\gamma_1 \in \mathrm{grd}(\Lambda)$, but since Λ is assumed to be non-contradictory, this contradicts the fact that $L \in \mathrm{grd}(\Lambda)$, or there exist ground closures $(C_{1i} \cdot \Gamma_{1i}; \gamma_{1i})$ of constrained clauses $C_{1i} \cdot \Gamma_{1i} \in \Phi$ that satisfy the conditions of Definition 17 for $(C_1 \cdot \Gamma_1; \gamma_1)$. But then $\{ (C_i \cdot \Gamma_i; \gamma_i) \mid 2 \leq i \leq n \} \cup \{ (C_{1i} \cdot \Gamma_{1i}; \gamma_{1i}) \mid 1 \leq i \leq m \}$ would also satisfy the conditions of Definition 17 for $(C \cdot \Gamma; \gamma)$, contradicting the minimality of $\{ (C_i \cdot \Gamma_i; \gamma_i) \mid 1 \leq i \leq n \}$. □

We are going to establish some results that relate redundancy and universal redundancy.

Lemma 19 *If $(C \cdot \Gamma; \gamma)$ is universally redundant wrt. $\Lambda \vdash \Phi$ and $\mathscr{D}$, and $(C \cdot \Gamma; \gamma)$ is a relevant closure wrt. $(\Lambda, R_{\mathscr{D}})$, then $(C \cdot \Gamma; \gamma)$ is redundant wrt. $\Lambda \vdash \Phi$ and $\mathscr{D}$.*

Proof Assume that $(C \cdot \Gamma; \gamma)$ is universally redundant wrt. $\Lambda \vdash \Phi$ and $\mathscr{D}$ and that $(C \cdot \Gamma; \gamma)$ is a relevant closure wrt. $(\Lambda, R_{\mathscr{D}})$. Then Λ produces Γ and $\Gamma\gamma$ by the same literals. Consequently, there cannot exist a $K \in \Gamma$ such that $\overline{K}\gamma \in \mathrm{grd}(\Lambda)$. By property (iii) of universal redundancy, there are ground closures $(C_i \cdot \Gamma_i; \gamma_i)$ of constrained clauses $C_i \cdot \Gamma_i \in \Phi$ such that $C_1\gamma_1 \ldots C_n\gamma_n \models C\gamma$; from property (i) we conclude that $(C \cdot \Gamma; \gamma)$ is entailed by $\{(C_1 \cdot \Gamma_1; \gamma_1), \ldots, (C_n \cdot \Gamma_n; \gamma_n)\}$ wrt. Λ, i.e., $(C_1 \cdot \Gamma_1; \gamma_1), \ldots, (C_n \cdot \Gamma_n; \gamma_n) \models_\Lambda (C \cdot \Gamma; \gamma)$. It remains to show that $(C_i \cdot \Gamma_i; \gamma_i)$ is a relevant closure wrt. Λ, for all $i = 1, \ldots, n$. First, by property (i) again we get that Λ produces Γ_i and $\Gamma_i\gamma_i$ by the same literals. Second, because $(C \cdot \Gamma; \gamma)$ is a relevant closure wrt. $(\Lambda, R_{\mathscr{D}})$, $y\gamma$ is irreducible wrt. $R_{\mathscr{D}}$ for every $y \in \mathit{Var}(C) \cap \mathit{Var}(\Gamma)$. By property (ii), each $(C_i \cdot \Gamma_i; \gamma)$ is smaller than $\mathscr{D}$. It follows $R_{(C_i \cdot \Gamma_i; \gamma)} \subseteq R_{\mathscr{D}}$. By property (iv) then, $x\gamma_i$ is irreducible wrt. $R_{(C_i \cdot \Gamma_i; \gamma)}$ for every $x \in \mathit{Var}(C_i) \cap \mathit{Var}(\Gamma_i)$, which suffices to complete the proof. □

Corollary 20 *If $C \cdot \Gamma$ is universally redundant wrt. $\Lambda \vdash \Phi$, then every relevant closure of $C \cdot \Gamma$ wrt. Λ is redundant wrt. $\Lambda \vdash \Phi$.*

Proof Suppose that $C \cdot \Gamma$ is universally redundant wrt. $\Lambda \vdash \Phi$, i.e., that every ground closure $(C \cdot \Gamma; \gamma)$ is universally redundant wrt. $\Lambda \vdash \Phi$ and $(C \cdot \Gamma; \gamma)$, for every ground substitution γ for $C \cdot \Gamma$. Let γ be any ground substitution for $C \cdot \Gamma$ such that $(C \cdot \Gamma; \gamma)$ is a relevant closure wrt. Λ. By setting $\mathscr{D} = (C \cdot \Gamma; \gamma)$ the result follows immediately from Lemma 19. □

The restriction to relevant closures in Corollary 20 can not be dropped. For example, with $C \cdot \Gamma = y \approx \mathsf{b} \vee y \approx \mathsf{b} \cdot \mathsf{P}(y)$ and $\gamma = \{y \mapsto \mathsf{a}\}$ the ground closure $(C \cdot \Gamma; \gamma)$ is universally redundant wrt. $\mathsf{P}(x) \vdash (x \approx \mathsf{b} \cdot \mathsf{P}(x)), \mathsf{a} \approx \mathsf{c}$ (take the ground closure $(x \approx \mathsf{b} \cdot \mathsf{P}(x); \{x \mapsto \mathsf{a}\})$

to show that), but $(C \cdot \Gamma; \gamma)$ is not redundant wrt. this sequent, as the ground closure $(x \approx \mathsf{b} \cdot \mathsf{P}(x); \{x \mapsto \mathsf{a}\})$ needed to establish that, is not a *relevant* closure, as $x\gamma = \mathsf{a}$ is reducible wrt. $\mathsf{a} \to \mathsf{c} \in R_{(x \approx \mathsf{b} \cdot \mathsf{P}(x); \{x \mapsto \mathsf{a}\})}$.

Lemma 16 also holds in terms of universal redundancy.

Corollary 21 *Let $\Lambda \vdash \Phi$ be a sequent and $(C \cdot \Gamma; \gamma)$ a ground closure of a clause $C \cdot \Gamma \in \Phi$. If $(C \cdot \Gamma; \gamma)$ is universally redundant wrt. $\Lambda \vdash \Phi$ and $(\Lambda, R_{(C \cdot \Gamma; \gamma)}) \models \Phi^{\Lambda}_{(C \cdot \Gamma; \gamma)}$ then $(C \cdot \Gamma; \gamma)$ does not generate a rewrite rule in $R_{\Lambda \vdash \Phi}$.*

Proof Suppose that $(C \cdot \Gamma; \gamma)$ is universally redundant wrt. $\Lambda \vdash \Phi$. If $(C \cdot \Gamma; \gamma)$ is not a relevant closure wrt. Λ, then by property 5 of the definition of model construction $(C \cdot \Gamma; \gamma)$ cannot generate a rewrite rule. If $(C \cdot \Gamma; \gamma)$ is a relevant closure wrt. Λ, then by Corollary 20 $(C \cdot \Gamma; \gamma)$ is redundant wrt. $\Lambda \vdash \Phi$. Now apply Lemma 16. □

The following definition exploits the notion of universal redundancy in universally redundant *inferences*.

Definition 22 (Universally Redundant $\iota_{\mathcal{ME}+\mathsf{Sup}}$ Inference) Let $\Lambda \vdash \Phi$ and $\Lambda' \vdash \Phi'$ be sequents. An $\iota_{\mathcal{ME}+\mathsf{Sup}}$ inference with premise $\Lambda \vdash \Phi$ and selected clause $C \cdot \Gamma \in \Phi$ is *universally redundant wrt. $\Lambda' \vdash \Phi'$* iff for every ground closure $(C \cdot \Gamma; \gamma)$ of $C \cdot \Gamma$, $(C \cdot \Gamma; \gamma)$ is universally redundant wrt. $\Lambda' \vdash \Phi'$, or the following holds, depending on the inference rule applied:

Deduce: One of the following holds, where $C' \cdot \Gamma'$ is the conclusion of the underlying ι_{Base} inference:
- (i) Applying γ to all premises and $C' \cdot \Gamma'$ of the underlying ι_{Base} inference does not result in a ground instance via γ of this ι_{Base} inference.
- (ii) $(C' \cdot \Gamma'; \gamma)$ is universally redundant wrt. $\Lambda' \vdash \Phi'$ and $(C \cdot \Gamma; \gamma)$.
- (iii) In case of Sup-Neg or Sup-Pos, $(C'' \cdot \Gamma''; \gamma)$ is universally redundant wrt. $\Lambda' \vdash \Phi'$, where $C'' \cdot \Gamma''$ is the left premise.

Split: $C \cdot \Gamma = \Box \cdot \Gamma$ and Λ' does not produce Γ.
Close: $C \cdot \Gamma = \Box \cdot \emptyset \in \Phi'$. □

With a view to implementation, it is important to know that adding the conclusion of a Deduce inference to the current clause set renders the inference universally redundant. This follows from the following proposition.

Proposition 23 *Let $\Lambda \vdash \Phi$ be a sequent, $C \cdot \Gamma$ and $C' \cdot \Gamma'$ be two constrained clauses with $C \cdot \Gamma \in \Phi$, and γ a ground substitution. If $(C' \cdot \Gamma'; \gamma) \succ (C \cdot \Gamma; \gamma)$ then $(C \cdot \Gamma; \gamma)$ is universally redundant wrt. $\Lambda \vdash \Phi$ and $(C' \cdot \Gamma'; \gamma)$.*

Proof Assume $(C' \cdot \Gamma'; \gamma) \succ (C \cdot \Gamma; \gamma)$. By taking $\mathscr{D} = (C' \cdot \Gamma'; \gamma)$, $n = 1$, $C_1 \cdot \Gamma_1 = C \cdot \Gamma$ and $\gamma_1 = \gamma$ in Definition 17 the result follows trivially. □

The following lemma will be used later to prove completeness in presence of a simplification rule, which permits to delete constrained clauses from the current sequent.

Lemma 24 *If a* Deduce *inference is universally redundant wrt. $\Lambda \vdash \Phi$, $\Lambda' \supseteq \Lambda$, and Φ' is obtained from Φ by deleting constrained clauses that are universally redundant wrt. $\Lambda \vdash \Phi$ and/or by adding arbitrary constrained clauses, then it universally redundant wrt. $\Lambda' \vdash \Phi'$.*

Proof Analogously to the proof of Lemma 18. □

Summarizing, and referring to the notion of derivation trees formally defined in Section 9 below, the results so far indicate that a constrained clause that is universally redundant at some node of the derivation tree will remain universally redundant in all successor nodes (by Lemma 18), that all its relevant ground closures are redundant (and therefore cannot be minimal counterexamples in the model construction, by Corollary 20), and that its ground closures cannot generate rewrite rules (by Corollary 21). Consequently, a universally redundant clause can be deleted from a clause set without endangering refutational completeness. We emphasize that for clauses with empty constraints, universal redundancy coincides with the classical notion of redundancy for the Superposition calculus.

Definition 25 (Saturated Sequent) A sequent $\Lambda \vdash \Phi$ is *saturated* iff every $\iota_{\mathcal{ME}+\mathsf{Sup}}$ inference with premise $\Lambda \vdash \Phi$ is universally redundant wrt. $\Lambda \vdash \Phi$. □

The results so far allow us to establish our first main result.

Theorem 26 (Static Completeness) *If $\Lambda \vdash \Phi$ is a saturated sequent with a non-contradictory context Λ and $\square\cdot\emptyset \notin \Phi$ then the induced rewrite system $R_{\Lambda\vdash\Phi}$ satisfies all relevant closures of all clauses in Φ wrt. Λ , i.e., $(\Lambda, R_{\Lambda\vdash\Phi}) \models \Phi^{\Lambda}$. Moreover, if Ψ is a clause set and Φ includes Ψ, i.e., $\{D\cdot\emptyset \mid D \in \Psi\} \subseteq \Phi$, then $R^{\star}_{\Lambda\vdash\Phi} \models \Psi$.*

The stronger statement $(\Lambda, R_{\Lambda\vdash\Phi}) \models \Phi$ does in general not follow, as $(\Lambda, R_{\Lambda\vdash\Phi})$ possibly falsifies a *non-relevant* ground closure of a constrained clause in Φ. An example is the sequent

$$\Lambda \vdash \Phi \;=\; \mathsf{P}(x),\ \mathsf{a}\to\mathsf{b},\ \neg\mathsf{P}(\mathsf{b}) \vdash \mathsf{P}(x)\cdot\mathsf{P}(x)\ .$$

Suppose $\mathsf{a}\succ\mathsf{b}$. We get $R_{\Lambda\vdash\Phi} = \{\mathsf{a}\to\mathsf{b}\}$. By taking $\gamma = \{x\mapsto\mathsf{a}\}$ observe that $\Lambda \models (\mathsf{P}(x),\gamma)$ but $R^{\star}_{\Lambda\vdash\Phi} \not\models \mathsf{P}(x)\gamma$, hence $(\Lambda, R_{\Lambda\vdash\Phi}) \not\models (\mathsf{P}(x)\cdot\mathsf{P}(x);\gamma)$. Deriving $\square\cdot(\neg\mathsf{P}(x),\mathsf{P}(x))$ does not help to close $\Lambda \vdash \Phi$. But notice that x is a shared variable and $x\gamma$ is reducible wrt. $R_{\Lambda\vdash\Phi}$, and so $(\mathsf{P}(x)\cdot\mathsf{P}(x);\gamma)$ is not a *relevant* closure, and Theorem 26 is not violated.

Proof Let $\Lambda \vdash \Phi$ be a saturated sequent with a non-contradictory context and suppose $\square\cdot\emptyset \notin \Phi$. Once $(\Lambda, R_{\Lambda\vdash\Phi}) \models \Phi^{\Lambda}$ is established we get the second statement $R^{\star}_{\Lambda\vdash\Phi} \models \Psi$ by the following argument. Let $C\gamma$ be a ground instance of a clause $C \in \Psi$. It suffices to show $R^{\star}_{\Lambda\vdash\Phi} \models C\gamma$. With Definition 5 it follows that every ground closure of a constrained clause with empty constraint is always relevant, for every pair (Λ, R). Hence, and more formally, $(C\cdot\emptyset;\gamma) \in \{D\cdot\emptyset \mid D\in\Psi\}^{\Lambda}$. With $\{D\cdot\emptyset \mid D\in\Psi\} \subseteq \Phi$ conclude trivially $(C\cdot\emptyset;\gamma) \in \Phi^{\Lambda}$. With $(\Lambda, R_{\Lambda\vdash\Phi}) \models \Phi^{\Lambda}$ we get $(\Lambda, R_{\Lambda\vdash\Phi}) \models (C\cdot\emptyset;\gamma)$, which means $\Lambda \not\models (\emptyset,\gamma)$ or $R^{\star}_{\Lambda\vdash\Phi} \models C\gamma$, equivalently $R^{\star}_{\Lambda\vdash\Phi} \models C\gamma$.

It remains to show $(\Lambda, R_{\Lambda\vdash\Phi}) \models \Phi^{\Lambda}$. For that, we show below that every ground closure $(C\cdot\Gamma;\gamma) \in \Phi^{\Lambda}$ satisfies one of the following two properties:

(i) $(\Lambda, R_{(C\cdot\Gamma;\gamma)}) \models (C\cdot\Gamma;\gamma)$.
(ii) $(C\cdot\Gamma;\gamma)$ generates a rewrite rule in $R_{\Lambda\vdash\Phi}$.

By Corollary 13 and Lemma 11 we conclude in both cases $(\Lambda, R_{\Lambda\vdash\Phi}) \models (C\cdot\Gamma;\gamma)$, which suffices to complete the proof.

Recall we need to show that every ground closure $(C\cdot\Gamma;\gamma) \in \Phi^{\Lambda}$ satisfies (i) or (ii). We proceed by well-founded induction over the closure ordering. Assume, by contradiction, a relevant counterexample, that is, a ground closure of a constrained clause in Φ that is relevant wrt. Λ and satisfies neither (i) nor (ii). By well-foundedness of the closure ordering $\succ$ there is a minimal such counterexample $(C\cdot\Gamma;\gamma)$ wrt. $\succ$, for some $C\cdot\Gamma \in \Phi$ and ground substitution γ. By minimality of $(C\cdot\Gamma;\gamma)$, every ground closure of a clause in Φ that is

relevant wrt. Λ and that is smaller than $(C \cdot \Gamma; \gamma)$ satisfies (i) or (ii), hence, by Corollary 13 and Lemma 11 again, is satisfied by $(\Lambda, R_{C\cdot\Gamma;\gamma})$.

We distinguish various cases now, each leading to a contradiction.

Case 1: $(C \cdot \Gamma; \gamma)$ is redundant wrt. $\Lambda \vdash \Phi$ or $(C \cdot \Gamma; \gamma)$ is universally redundant wrt. $\Lambda \vdash \Phi$.
If $(C \cdot \Gamma; \gamma)$ is redundant wrt. $\Lambda \vdash \Phi$ then there are ground closures of clauses in Φ that are relevant wrt. Λ, each smaller wrt. $\succ$ than $(C \cdot \Gamma; \gamma)$, and that entail $(C \cdot \Gamma; \gamma)$ wrt. Λ. By the induction hypothesis, all these closures are satisfied by $(\Lambda, R_{(C\cdot\Gamma;\gamma)})$. As they entail $(C \cdot \Gamma; \gamma)$ wrt. Λ, we conclude $(\Lambda, R_{(C\cdot\Gamma;\gamma)}) \models (C \cdot \Gamma; \gamma)$, contradicting our assumption that (i) is not satisfied.

Because $(C \cdot \Gamma; \gamma)$ is assumed to be a relevant closure wrt. Λ, by Corollary 20 $(C \cdot \Gamma; \gamma)$ cannot be universally redundant wrt. $\Lambda \vdash \Phi$ either.

Case 2: $Var(C)\gamma$ is reducible wrt. $R_{(C\cdot\Gamma;\gamma)}$.
The $\mathcal{ME}$+Sup calculus does not need to paramodulate into or below variables. To explain the completeness of this restriction we have to know that property (i) is always satisfied if $Var(C)\gamma$ is reducible wrt. $R_{(C\cdot\Gamma;\gamma)}$. Because $(C \cdot \Gamma; \gamma)$ is a relevant closure we already know with Definition 5 that $(Var(C) \cap Var(\Gamma))\gamma$ is irreducible wrt. $R_{(C\cdot\Gamma;\gamma)}$. If $x\gamma$ is reducible for some $x \in Var(C) \setminus Var(\Gamma)$, then a term in the range of γ can be replaced by a smaller yet congruent term wrt. $R^{\star}_{(C\cdot\Gamma;\gamma)}$. Observe that this results in a smaller (wrt. $\succ$) relevant counterexample, thus contradicting the choice of $(C \cdot \Gamma; \gamma)$.

Case 3: $C = s \not\approx t \vee D$ with $(s \not\approx t)\gamma$ maximal in $(s \not\approx t \vee D)\gamma$.
Suppose that none of the preceding cases holds, that C is of the form $s \not\approx t \vee D$, and that $(s \not\approx t)\gamma$ is maximal in $(s \not\approx t \vee D)\gamma$.

Case 3.1: $s\gamma = t\gamma$.
If $s\gamma = t\gamma$ then there is a ground Deduce inference with premise $(C \cdot \Gamma)\gamma = (s \not\approx t \vee D \cdot \Gamma)\gamma$ and conclusion $(D \cdot \Gamma)\gamma$, which is an instance of a Deduce inference with an underlying Ref inference applied to $C \cdot \Gamma$. By saturation, this Deduce inference is universally redundant wrt. $\Lambda \vdash \Phi$. Because $(C \cdot \Gamma; \gamma)$ is not universally redundant wrt. $\Lambda \vdash \Phi$, the closure $(D \cdot \Gamma; \gamma)$ must be universally redundant wrt. $\Lambda \vdash \Phi$ and $(C \cdot \Gamma; \gamma)$ by definition of redundant inferences. Furthermore, by Lemma 10, $(D \cdot \Gamma; \gamma)$ is a relevant closure wrt. $(\Lambda, R_{(C\cdot\Gamma;\gamma)})$, hence, by Lemma 19, $(D \cdot \Gamma; \gamma)$ is redundant wrt. $\Lambda \vdash \Phi$ and $(C \cdot \Gamma; \gamma)$ and therefore follows from relevant closures of clauses in Φ wrt. Λ that are smaller than $(C \cdot \Gamma; \gamma)$. By the induction hypothesis, these clauses are satisfied by $(\Lambda, R_{(C\cdot\Gamma;\gamma)})$, hence $(\Lambda, R_{(C\cdot\Gamma;\gamma)}) \models (D \cdot \Gamma; \gamma)$, and, trivially, $(\Lambda, R_{(C\cdot\Gamma;\gamma)}) \models (C \cdot \Gamma; \gamma)$, contradicting our assumption. (In other cases below we will use similar arguments without explicit reference to Lemmas 10 and 19)

Case 3.2: $s\gamma \neq t\gamma$.
If $s\gamma \neq t\gamma$ then without loss of generality assume $s\gamma \succ t\gamma$. With $(\Lambda, R_{(C\cdot\Gamma;\gamma)}) \not\models (s \not\approx t \vee D \cdot \Gamma; \gamma)$ it follows $\Lambda \models (\Gamma, \gamma)$ and $R^{\star}_{(C\cdot\Gamma;\gamma)} \not\models (s \not\approx t \vee D)\gamma$, and so $R^{\star}_{(C\cdot\Gamma;\gamma)} \models (s \approx t)\gamma$. Because $R_{(C\cdot\Gamma;\gamma)}$ is a convergent (ordered) rewrite system, $s\gamma$ and $t\gamma$ must have the same normal forms. Recall we assumed $s\gamma \succ t\gamma$, and so $s\gamma$ must be reducible wrt. $R_{(C\cdot\Gamma;\gamma)}$. Suppose $s\gamma = s\gamma[l]_p$ for some position p and rule $l \to r \in R_{(C\cdot\Gamma;\gamma)}$. We distinguish two cases.

Case 3.2.1: $l \approx r$ is a split atom.
If $l \approx r$ is a split atom then with $l \to r \in R_{(C\cdot\Gamma;\gamma)}$ and Lemma 7-(i) it follows that Λ produces $l \to r$. For later use let $l' \to r' \in_{\sim} \Lambda$ be a fresh variant of a rewrite literal in Λ that produces $l \to r$ in Λ and assume that γ has already been extended so that $(l' \to r')\gamma = l \to r$.

The conclusions so far give that Deduce is applicable with underlying ground U-Sup-Neg inference with left premise $l \to r$, right premise $s\gamma[l]_p \not\approx t\gamma \vee D\gamma \cdot \Gamma\gamma$ and conclusion

$s\gamma[r]_p \not\approx t\gamma \vee D\gamma \cdot (\Gamma\gamma, l \to r)$. The next step is to show that this ground inference is a ground instance via γ of a U-Sup-Neg inference with premises $l' \to r'$ and $C \cdot \Gamma = s[u]_p \not\approx t \vee D \cdot \Gamma$ and conclusion $C' \cdot \Gamma' := (s[r']_p \not\approx t \vee D \cdot (\Gamma, l' \to r'))\sigma$, where σ is an mgu of l' and u. This follows from the observation that the position p in s indeed exists and that u cannot be a variable, because otherwise $u\gamma$ would be reducible wrt. $R_{(C\cdot\Gamma;\gamma)}$ (the rule $l \to r$ would rewrite it, as $l = u\gamma$), but we know that $Var(C)\gamma$ is irreducible wrt. $R_{(C\cdot\Gamma;\gamma)}$.

We need to know that Deduce is applicable with this U-Sup-Neg inference underlying it. For this, it only remains to be shown that Λ produces $(l' \to r')\sigma$. This, however, follows trivially from the fact that $l' \to r' \in \Lambda$ produces $l \to r$ in Λ, as obtained above, and $l' \to r' \gtrsim (l' \to r')\sigma \gtrsim l \to r$ (for if there were an $l'' \not\to r'' \in \Lambda$ such that $l' \to r' \gtrsim l'' \to r'' \gtrsim (l' \to r')\sigma$ then $l' \to r'$ would not produce $l \to r$ in Λ either).

By saturation, this Deduce inference is universally redundant wrt. $\Lambda \vdash \Phi$. Because the ground closure $(C \cdot \Gamma; \gamma)$ is not universally redundant wrt. $\Lambda \vdash \Phi$, the ground closure $(C' \cdot \Gamma'; \gamma)$ must be universally redundant wrt. $\Lambda \vdash \Phi$ and $(C \cdot \Gamma; \gamma)$ by definition of redundant inferences. Furthermore, by Lemma 10 $(C' \cdot \Gamma'; \gamma)$ is a relevant closure wrt. $(\Lambda, R_{(C\cdot\Gamma;\gamma)})$. With Lemma 19 it follows that $(C' \cdot \Gamma'; \gamma)$ is redundant wrt. $\Lambda \vdash \Phi$ and $(C \cdot \Gamma; \gamma)$ and hence follows from relevant closures of clauses in Φ wrt. Λ that are smaller than $(C \cdot \Gamma; \gamma)$. By the induction hypothesis, these clauses are satisfied by $(\Lambda, R_{(C\cdot\Gamma;\gamma)})$, hence $(\Lambda, R_{(C\cdot\Gamma;\gamma)}) \models (C' \cdot \Gamma'; \gamma)$. Recall we are considering a ground U-Sup-Neg inference that is a ground instance (via γ) of the U-Sup-Neg inference concluded above. Therefore, $C'\gamma = (s[r']_p \not\approx t \vee D)\sigma\gamma = (s[r]_p \not\approx t \vee D)\gamma$. With $l \to r \in R_{(C\cdot\Gamma;\gamma)}$ by congruence it follows $R^\star_{(C\cdot\Gamma;\gamma)} \models (s \not\approx t \vee D)\gamma$, a plain contradiction to the assumption above for case (3.2).

Case 3.2.2: $l \approx r$ is a superposition atom.
The proof is similar to case (3.2.1), however referring to Sup-Neg inferences instead of U-Sup-Neg inferences, and where the rewrite rule $l \to r \in R$ is generated by a ground closure $(C_0 \cdot \Gamma_0; \gamma)$ of a constrained clause $C_0 \cdot \Gamma_0 \in \Phi$ instead of a rewrite literal from Π_Λ. With Lemmas 10 and 11, and the same congruence argument using $l \to r$ we get the same contradiction as in case (3.2.1).

Case 4: $C = s \approx t \vee D$ with $(s \approx t)\gamma$ maximal in $(s \approx t \vee D)\gamma$.
Suppose that C is of the form $s \approx t \vee D$ and that $(s \approx t)\gamma$ is maximal in $(s \approx t \vee D)\gamma$. With $(C \cdot \Gamma; \gamma)$ being a counterexample it follows $\Lambda \models (\Gamma, \gamma)$ but $R^\star_{(C\cdot\Gamma;\gamma)} \not\models (s \approx t \vee D)\gamma$. From the latter follows $R^\star_{(C\cdot\Gamma;\gamma)} \not\models (s \approx t)\gamma$, and so $s\gamma = t\gamma$ is impossible. Hence suppose $s\gamma \neq t\gamma$.

Case 4.1: $(s \approx t)\gamma$ is a split atom.
If $(s \approx t)\gamma$ is a split atom we distinguish two further cases.

Case 4.1.1: $s\gamma$ or $t\gamma$ is reducible wrt. $R_{(C\cdot\Gamma;\gamma)}$.
If $s\gamma$ or $t\gamma$ is reducible wrt. $R_{(C\cdot\Gamma;\gamma)}$ then there is a rule $l \to r \in R_{(C\cdot\Gamma;\gamma)}$ such that $s\gamma = s\gamma[l]_p$ or $t\gamma = t\gamma[l]_p$, for some position p. If $l \to r$ is generated by a rewrite literal from Π_Λ then the same argumentation as in case (3.2.1) applies. The only changes are that instead of (ground instances of) U-Sup-Neg inferences now (ground instances of) U-Sup-Pos inferences are considered, and that $s\gamma \succ t\gamma$ does not apply.

If $l \to r$ is generated by a ground closure from Φ^{gr} then the same argumentation as in case (3.2.2) applies. The relevant inference rule in this case is Sup-Pos.

Case 4.1.2: $s\gamma$ and $t\gamma$ are irreducible wrt. $R_{(C\cdot\Gamma;\gamma)}$.
If $s\gamma$ and $t\gamma$ are irreducible wrt. $R_{(C\cdot\Gamma;\gamma)}$ then assume, w.l.o.g., $s\gamma \succ t\gamma$. The ordering on closures and rewrite literals is defined in such a way that the closure $(s \approx t \vee D \cdot \Gamma; \gamma)$ is necessarily greater than the rewrite literal $(s \to t)\gamma$. With Lemma 7-(ii) then conclude that Λ

produces $(s \not\approx t)\gamma$. This indicates that a Deduce inference with an underlying ground U-Res inference exists. More precisely, the left premise of that ground inference is $(s \not\approx t)\gamma$, the right premise is $(s \approx t \vee D \cdot \Gamma)\gamma$ and the conclusion is $(D \cdot (\Gamma, s \not\approx t))\gamma$. It is routine by now to check that this ground U-Res inference is a ground instance via γ of a U-Res inference with a right premise from Φ that is not universally redundant wrt. $\Lambda \vdash \Phi$, and a left premise from Λ.

As in case (3.2.1) we can show that the Deduce inference with the latter underlying U-Res inference exists. The rest of the proof uses the same arguments as in case (3.2.1), too.

Case 4.2: $(s \approx t)\gamma$ *is a superposition atom.*
If $(s \approx t)\gamma$ is a superposition atom assume, w.l.o.g., $s\gamma \succ t\gamma$ and distinguish two cases.

Case 4.2.1: $s\gamma$ *is reducible wrt.* $R_{(C \cdot \Gamma;\gamma)}$.
If s is reducible wrt. $R_{(C \cdot \Gamma;\gamma)}$ the argumentation is similar to case (3.2.2) and is omitted.

Case 4.2.2: $s\gamma$ *is irreducible wrt.* $R_{(C \cdot \Gamma;\gamma)}$.
In this case, either $(s \approx t \vee D \cdot \Gamma; \gamma)$ generates $(s \to t)\gamma$, so property (ii) would be satisfied, contradicting our assumption. Or a Eq-Fact inference exists, which shows that a smaller ground closure exists that, by redundancy, is satisfied in $(\Lambda, R_{(C \cdot \Gamma;\gamma)})$ and entails $(s \approx t \vee D \cdot \Gamma; \gamma)$. Then we get a contradiction to the assumption that $(s \approx t \vee D \cdot \Gamma; \gamma)$ is a minimal counterexample.

Case 5: $C = \square$.
Suppose $C = \square$. By assumption $\square \cdot \emptyset \notin \Phi$, and so $\square \cdot \emptyset \notin \Phi^{\mathrm{gr}}$. Hence $\Gamma \neq \emptyset$. First we are going to show that Split is applicable to $\Lambda \vdash \Phi$ with $\square \cdot \Gamma \in \Phi$.

Because $\square \cdot \Gamma$ is a relevant closure wrt. Λ, by Definition 5 Λ produces Γ and Λ produces $\Gamma\gamma$ by the same literals. We are given that Λ is not contradictory. This entails $\overline{L} \notin_\sim \Lambda$, for every $L \in \Gamma$. For, if there is a literal $L \in \Gamma$ with $\overline{L} \in_\sim \Lambda$ then Λ would produce $\overline{L}$. But in this case Λ can produce L only if $L \in_\sim \Lambda$, and so Λ would be contradictory.

It is also impossible that $L \in_\sim \Lambda$, for every $L \in \Gamma$ because then Close would be applicable, and by saturation Close would be universally redundant, which is the case only if $\square \cdot \emptyset \in \Phi$, which we have already excluded. Altogether conclude that there is a literal $K \in \Gamma$ such that neither K nor $\overline{K}$ is contradictory with Λ. This shows that a Split inference with selected clause $\square \cdot \Gamma$ exists, where K is the literal split on. Moreover, from above we know that Λ produces Γ. It follows that this Split inference is not universally redundant wrt. $\Lambda \vdash \Phi$. However, by saturation it is universally redundant wrt. $\Lambda \vdash \Phi$, a plain contradiction. □

Theorem 26 applies to a *statically* given sequent $\Lambda \vdash \Phi$. The connection to the *dynamic* derivation process of the $\mathcal{ME}$+Sup calculus will be given later, and Theorem 26 will be essential then in proving the completeness of the $\mathcal{ME}$+Sup calculus.

9 Derivations with Simplification

To make derivations in $\mathcal{ME}$+Sup practical, the universal redundancy criteria defined above should be made available not only to avoid inferences, but also to, e.g., delete universally redundant clauses that come up in derivations. The following generic simplification rule covers many practical cases.

$$\textsf{Simp}\ \frac{\Lambda \vdash \Phi, C \cdot \Gamma}{\Lambda \vdash \Phi, C' \cdot \Gamma'}$$

if

(i) $C \cdot \Gamma$ is universally redundant wrt. $\Lambda \vdash \Phi, C' \cdot \Gamma'$, and
(ii) $(\Lambda^{\mathrm{c}})^a \cup (\Phi \cup \{C \cdot \Gamma\})^{\mathrm{c}} \models (C' \cdot \Gamma')^{\mathrm{c}}$.

The Simp rule generalizes the widely-used simplification rules of the Superposition calculus, such as deletion of trivial equations $t \not\approx t$ from clauses, demodulation with unit clauses and (non-proper) subsumption; these rules immediately carry over to $\mathcal{ME}$+Sup as long as all involved clauses have empty constraints. Also, as said above, the usual unit propagation rules of the (propositional) DPLL procedure are covered in a more general form. As $\mathcal{ME}$+Sup is intended as a generalization of propositional DPLL (among others), it is mandatory to provide this feature.

Condition (ii) is needed for soundness. The $\cdot^a$-operator uniformly replaces each variable in each (unit) clause by a constant a. This way, all splits are effectively over complementary propositional literals.

Derivations. The purpose of the $\mathcal{ME}$+Sup calculus is to build for a given clause set a derivation tree over sequents all of whose branches end in a closed sequent iff the clause set is unsatisfiable. Formally, we consider ordered trees $\mathbf{T} = (\mathbf{N}, \mathbf{E})$ where $\mathbf{N}$ and $\mathbf{E}$ are the sets of nodes and edges of $\mathbf{T}$, respectively, and the nodes N are labelled with sequents. Often we will identify a node's label with the node itself.

Derivation trees $\mathbf{T}$ *(of a set* $\{C_1, \ldots, C_n\}$ *of clauses)* are defined inductively as follows: an *initial tree* is a derivation tree, i.e., a tree $\mathbf{T}$ with a root node only that is labeled with the sequent $\neg x \vdash C_1 \cdot \emptyset, \ldots, C_n \cdot \emptyset$; if $\mathbf{T}$ is a derivation tree, N is a leaf node of $\mathbf{T}$ and $\mathbf{T}'$ is a tree obtained from $\mathbf{T}$ by adding one or two child nodes to N so that N is the premise of an $\iota_{\mathcal{ME}\text{+Sup}}$ inference or a Simp inference, and the child node(s) is (are) its conclusion(s), then $\mathbf{T}'$ is derivation tree. In this case we say that $\mathbf{T}'$ is *derived* from $\mathbf{T}$. A *derivation (of* $\{C_1, \ldots, C_n\}$*)* is a possibly infinite sequence of derivation trees that starts with an initial tree and all subsequent derivation trees are derived from their immediate predecessor. A derivation $\Delta = ((\mathbf{N}_i, \mathbf{E}_i))_{i<\kappa}$, where $\kappa \in \mathbb{N} \cup \{\omega\}$, determines a *limit tree* $(\bigcup_{i<\kappa} \mathbf{N}_i, \bigcup_{i<\kappa} \mathbf{E}_i)$. It is easy to show that a limit tree of a derivation Δ is indeed a tree. But note that it will not be a derivation tree unless Δ is finite.

Now let $\mathbf{T}$ be the limit tree of some derivation, let $\mathbf{B} = (N_i)_{i<\kappa}$ be a branch in $\mathbf{T}$ with κ nodes, and let $\Lambda_i \vdash \Phi_i$ be the sequent labeling node N_i, for all $i < \kappa$. Define $\Lambda_{\mathbf{B}} = \bigcup_{i<\kappa} \bigcap_{i \le j < \kappa} \Lambda_j$[12] and $\Phi_{\mathbf{B}} = \bigcup_{i<\kappa} \bigcap_{i \le j < \kappa} \Phi_j$, the sets of *persistent context literals* and *persistent clauses*, respectively. These two sets can be combined to obtain the *limit sequent* $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ (of $\mathbf{T}$).

As usual, the completeness of $\mathcal{ME}$+Sup relies on a suitable notion of fairness, which is defined in terms of exhausted branches. When we say that "X is not persistent" we mean that X is not among the persistent context literals or X is not among the persistent clauses, depending on whether X is a rewrite literal or a constrained clause.

Definition 27 (Exhausted Branch) Let $\mathbf{T}$ be a limit tree and $\mathbf{B} = (N_i)_{i<\kappa}$ a branch in $\mathbf{T}$ with κ nodes. For all $i < \kappa$, let $\Lambda_i \vdash \Phi_i$ be the sequent labeling node N_i. The branch $\mathbf{B}$ is *exhausted* iff

(i) for all $i < \kappa$, every $\iota_{\mathcal{ME}\text{+Sup}}$ inference with premise $\Lambda_i \vdash \Phi_i$ and a persistent selected clause and a persistent left premise (in case of Deduce) is universally redundant wrt. $\Lambda_j \vdash \Phi_j$, for some $j < \kappa$ with $j \ge i$, and
(ii) $\square \cdot \emptyset \notin \Phi_{\mathbf{B}}$ □

[12] The definition of $\Lambda_{\mathbf{B}}$ is slightly more general as needed. Currently, there are no inference rules to delete context elements, and so $\Lambda_{\mathbf{B}}$ is always $\bigcup_{i<\kappa} \Lambda_i$.

A limit tree of a derivation is *fair* iff it is a refutation tree, that is, a finite tree all of whose leafs contain $\square \cdot \emptyset$ in the constrained clause part of their sequent, or it has an exhausted branch. A derivation is *fair* iff its limit tree is fair.

Notice that if in Definition 27, condition (i), the selected clause or the left premise (in case of Deduce) is universally redundant wrt. $\Lambda_i \vdash \Phi_i$, then the $\iota_{\mathcal{ME}+\mathsf{Sup}}$ inference is already redundant wrt. $\Lambda_i \vdash \Phi_i$. In other words, inferences with a universally redundant premise need not be carried out. In general, a fair proof procedure (and implementation) needs to make sure that every inference satisfying condition (i) eventually becomes universally redundant. This can always be achieved by actually carrying out the inference. (Proposition 23 provides the explanation for Deduce, for all other rules this is trivial.)

We are now going to establish some auxiliary results that justify to employ our concepts of redundancy in derivations.

The intended interpretation of a limit sequent $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ is the rewrite system $R_{\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}}$ which will denote by $R_{\mathbf{B}}$ for simplicity. As a further convenience, we denote the union of all context literals or all clauses of a branch $\mathbf{B} = (N_i)_{i<\kappa}$ by $\Lambda_{\mathbf{B}}^+ = \bigcup_{i<\kappa} \Lambda_i$ and $\Phi_{\mathbf{B}}^+ = \bigcup_{i<\kappa} \Phi_i$, respectively.

Lemma 28 *Let $C \cdot \Gamma$ be a constrained clause. If $C \cdot \Gamma$ is universally redundant wrt. $\Lambda_j \vdash \Phi_j$, for some $j < \kappa$, then $C \cdot \Gamma$ is universally redundant wrt. $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$.*

Proof The proof works in essentially the same way as in [BGW94]. Suppose that $C \cdot \Gamma$ is universally redundant wrt. $\Lambda_j \vdash \Phi_j$. Since $\Lambda_{\mathbf{B}} \supseteq \Lambda_j$ and $\Phi_{\mathbf{B}}^+ \supseteq \Phi_j$, Lemma 18 implies that $C \cdot \Gamma$ is universally redundant wrt. $\Lambda_j \vdash \Phi_{\mathbf{B}}^+$. Now observe that every constrained clause in $\Phi_{\mathbf{B}}^+ \setminus \Phi_{\mathbf{B}}$ has been deleted at some node of the branch $\mathbf{B}$, which is only possible if it was universally redundant wrt. some $\Lambda_k \vdash \Phi_k$ with $k < \kappa$. Again using Lemma 18, we see that every constrained clause in $\Phi_{\mathbf{B}}^+ \setminus \Phi_{\mathbf{B}}$ is universally redundant wrt. $\Lambda_j \vdash \Phi_{\mathbf{B}}^+$. Hence $\Phi_{\mathbf{B}}$ is obtained from $\Phi_{\mathbf{B}}^+$ by deleting universally redundant clauses, and using Lemma 18 a third time, we conclude that $C \cdot \Gamma$ is universally redundant wrt. $\Lambda_j \vdash \Phi_{\mathbf{B}}$. Because every ground rewrite literal in Λ_j is also contained in $\Lambda_{\mathbf{B}}$, the claim of the lemma follows immediately. □

Lemma 29 *Every* Deduce *inference that is universally redundant wrt. $\Lambda_j \vdash \Phi_j$, for some $j < \kappa$, is universally redundant wrt. $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$.*

Proof Analogously to the proof of Lemma 28 using Lemma 24. □

Proposition 30 (Exhausted Branches are Saturated) *If* **B** *is an exhausted branch of a limit tree of a fair derivation then $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ is saturated.*

Proof Suppose **B** is an exhausted branch of a limit tree of some fair derivation. We have to show that every $\iota_{\mathcal{ME}+\mathsf{Sup}}$ inference with premise $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ is universally redundant wrt. $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$. We do this by assuming such an inference and carrying out a case analysis wrt. the inference rule applied.

By Definition 27 there is no Close inference with premise $\Lambda_i \vdash \Phi_i$, for no $i < \kappa$, with a persistent closing clause and persistent closing literals. But then there is no Close inference with premise $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ either. (Because if so, for a large enough i there would be Close inference with premise $\Lambda_i \vdash \Phi_i$, which we excluded.) Thus there is nothing to show for Close.

If the inference rule is Split then let $\square \cdot \Gamma$ be the selected clause. There are only finitely many literals K, modulo renaming and modulo sign, that are more general than a given literal or set of literals such as Γ. Because no inference rule ever removes a literal from a context

or adds a variant or its complement to a literal that is already in a context, from some time k onwards, no more such literal K will be added to $\Lambda_k, \Lambda_{k+1}, \ldots$.

We are given that $\square \cdot \Gamma$ is persistent. Therefore suppose also $\square \cdot \Gamma \in \Lambda_k, \Lambda_{k+1}, \ldots$, or choose k big enough. Together this shows that a Split inference with premise $\Lambda_i \vdash \Phi_i$ exists (i could be k or smaller). By Definition 27 then, the Split inference is universally redundant wrt. $\Lambda_j \vdash \Phi_j$, for some $j < \kappa$ with $j \geq i$. By universal redundancy, this means that the selected clause $\square \cdot \Gamma$ is universally redundant wrt. $\Lambda_j \vdash \Phi_j$, or Λ_j does not produce Γ.

In the first case, use Lemma 28 to conclude that $\square \cdot \Gamma$ is universally redundant wrt. $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$, and so the Split inference with selected clause $\square \cdot \Gamma$ is universally redundant wrt. $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$.

In the second case let $j_1 < j_2 < \cdots$ be all those points greater than j such that each $\Lambda_{j_1}, \Lambda_{j_2}, \ldots$ produces Γ. By fairness, for each point j_l there must be a later point j'_l such that $\Lambda_{j'_l}$ does not produce Γ. This can be achieved only by adding literals to the context. With the argument above about the finitely many literals K with the properties mentioned there, the sequence $j_1 < j_2 < \cdots$ must be finite (it could be empty). In other words, there is a j_l such that for every $k \geq j_l$, Λ_k does not produce Γ. But then, $\Lambda_{\mathbf{B}}$ does not produce Γ either, which entails that $\square \cdot \Gamma$ is universally redundant wrt. $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$, and we are done, as in the first case.

If the inference rule is Deduce then by Definition 27 it is universally redundant wrt. $\Lambda_j \vdash \Phi_j$, for some $j \geq i$, and by Lemma 29 it is universally redundant wrt. $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$. □

Proposition 30 is instrumental in the proof of our main result, which is the following.

Theorem 31 (Completeness) *Let Ψ be a clause set and $\mathbf{T}$ be the limit tree of a fair derivation of Ψ. If $\mathbf{T}$ is not a refutation tree then Ψ is satisfiable; more specifically, for every exhausted branch $\mathbf{B}$ of $\mathbf{T}$ with limit sequent $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ and induced rewrite system $R_{\mathbf{B}} = R_{\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}}$ it holds $\Lambda_{\mathbf{B}}, R_{\mathbf{B}} \models (\Phi_{\mathbf{B}})^{\Lambda_{\mathbf{B}}}$ and $R^{\star}_{\mathbf{B}} \models \Psi$.*

Proof Suppose $\mathbf{T}$ is not a refutation tree and let $\mathbf{B}$ an exhausted branch of $\mathbf{T}$. By Proposition 30 the limit sequent $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ is saturated. It is easy to see that $\Lambda_{\mathbf{B}}$ is non-contradictory (the context in the initial sequent of the derivation is non-contradictory, and all inference rules preserve this property.) By Theorem 26 then $\Lambda_{\mathbf{B}}, R_{\mathbf{B}} \models (\Phi_{\mathbf{B}})^{\Lambda_{\mathbf{B}}}$.

To show $R^{\star}_{\mathbf{B}} \models \Psi$, let $C \in \Psi$ be any clause from Ψ, and it suffices to show $R^{\star}_{\mathbf{B}} \models C$. By definition of derivation, $C \cdot \emptyset \in \Phi_1$. If $C \cdot \emptyset \in \Phi_{\mathbf{B}}$ then the second part of Theorem 26 gives $R^{\star}_{\mathbf{B}} \models C$ immediately. Otherwise assume $C \cdot \emptyset \notin \Phi_{\mathbf{B}}$. Hence $C \cdot \emptyset$ has been removed at some time $k < \kappa$ from the clause set Φ_k of the sequent $\Lambda_k \vdash \Phi_k$ by an application of the Simp rule. By definition of Simp, $C \cdot \emptyset$ is universally redundant wrt. $\Lambda_{k+1} \vdash \Phi_{k+1}$. By Lemma 28, $C \cdot \emptyset$ is universally redundant wrt. $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$. Since $C \cdot \emptyset$ has an empty constraint, all its ground closures are relevant, and by Corollary 20, all relevant closures wrt. $\Lambda_{\mathbf{B}}$ are redundant wrt. $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$, hence they are entailed wrt. $\Lambda_{\mathbf{B}}$ by clauses in $(\Phi_{\mathbf{B}})^{\Lambda_{\mathbf{B}}}$. With $\Lambda_{\mathbf{B}}, R_{\mathbf{B}} \models (\Phi_{\mathbf{B}})^{\Lambda_{\mathbf{B}}}$, the first part of the theorem, which is already proved, we get $\Lambda_{\mathbf{B}}, R_{\mathbf{B}} \models C \cdot \emptyset$. With the constraint being empty, $R^{\star}_{\mathbf{B}} \models C$ follows immediately. □

The $\mathcal{ME}$+Sup calculus is also sound. The idea behind the soundness proof is to conceptually replace in a refutation tree every variable in every literal in all contexts by a constant, say, a. This results in a refutation tree where all splits are over complementary propositional literals. Regarding Close inferences, any closing clause will still be closing after instantiating all its variables in the same way. Furthermore, observe that the Ref, Sup-Neg, Sup-Pos and Eq-Fact inference rules are sound in the standard sense by taking the clausal forms of the premises and the conclusions. For the remaining ι_{Base} inference rules U-Sup-Pos, U-Sup-Neg and U-Res this is even simpler as the constraint in the conclusion contains the left

premise (they are strongly sound). The soundness of Simp follows from its condition (ii). This way, a set of ground instances can be identified that demonstrates the unsatisfiability of the clausal form of the constrained clause set in the root sequent. A formal soundness proof can be carried out as for the $\mathcal{ME}_{\mathrm{E}}$ calculus [BT05].

10 Querying the Limit Model

Many application areas not only require (refutational) theorem proving but also model computation. In software verification, for instance, non-valid conjectures come up frequently during the design process, which lead to non-provable, i.e., consistent input formulas for a theorem prover. Furthermore, certain application problems like consistency-based diagnosis, discourse representation, and planning can naturally be expressed as model computation tasks. Quite often then it is enough to compute a (one) designated model and being able to ask queries with respect to it, for instance, the model given by the rewrite system $R_{\mathbf{B}} = R_{\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}}$ for a finite exhausted branch $\mathbf{B}$.

If the sequent $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$ is non-ground and Σ contains at least one non-constant function symbol, then $R_{\mathbf{B}}$ is usually infinite. On the other hand, to test whether any given ground literal L is true in the interpretation $R^{\star}_{\mathbf{B}}$, it is sufficient to compute an initial fragment of the infinite union $R_{\mathbf{B}} = \bigcup_{\mathscr{C} \in (\Phi_{\mathbf{B}})^{\mathrm{gr}} \cup \Pi_{\Lambda_{\mathbf{B}}}} \mathcal{E}_{\mathscr{C}}$: Since $R^{\star}_{\mathbf{B}} \models s \approx t$ if and only if s and t can be reduced to the same term using $R_{\mathbf{B}}$, and since a ground term cannot be reduced by a ground rewrite rule whose left-hand side is larger than the term itself, it is clear that $R^{\star}_{\mathbf{B}} \models s \approx t$ if and only if $R^{\star}_{s,t} \models s \approx t$, where $R_{s,t} = \{l \to r \in R_{\mathbf{B}} \mid s \succeq l \text{ or } t \succeq l\}$. Under some additional restrictions, the set $R_{s,t}$ can be computed effectively: Let $\succ$ be a Knuth-Bendix ordering with strictly positive weights, and assume furthermore that for every constraint literal $l \to r$ or $l \not\to r$ of a constrained clause in $\Phi_{\mathbf{B}}$ all its instances are split literals.

A Knuth-Bendix ordering with strictly positive weights has the property that for any ground term u there are only finitely many ground terms $v \prec u$, which can moreover be enumerated effectively. Now observe that the closure ordering is defined in such a way that a closure $(C \cdot \Gamma; \gamma) \in \Phi^{\mathrm{gr}}$ can only contribute a rule $u\gamma \to v\gamma$ to $R_{s,t}$ if $x\gamma \preceq \max(s,t)$ for all $x \in \mathit{Var}(C)$. On the other hand, there is no such size restriction for variables in $\mathit{Var}(\Gamma) \setminus \mathit{Var}(C)$. However, almost all conditions of the model construction are independent of $x\gamma$ for $x \in \mathit{Var}(\Gamma) \setminus \mathit{Var}(C)$. The only exception is the requirement that $(C \cdot \Gamma; \gamma)$ must be a relevant closure wrt. $(\Lambda, R_{(C \cdot \Gamma; \gamma)})$ or, more precisely, the requirement that $\Lambda \models (\Gamma, \gamma)$. Given a ground substitution γ' for $\mathit{Var}(C)$ that satisfies the remaining conditions, it is therefore sufficient to check whether there *exists some* ground substitution γ'' for $\mathit{Var}(\Gamma) \setminus \mathit{Var}(C)$ such that $l\gamma'\gamma'' \succ r\gamma'\gamma''$, for every $l \to r \in \Gamma$ or $l \not\to r \in \Gamma$, and such that Λ produces Γ and Λ produces $\Gamma\gamma'\gamma''$ by the same literals – it is not necessary to enumerate all possible γ''. Since the first-order theory of the KBO is decidable [ZSM05], this can be done by checking the KBO formula

$$\exists x_1, \ldots, x_n \bigwedge_{L \in \Gamma} \big(\mathit{lhs}(L)\gamma' \succ \mathit{rhs}(L)\gamma' \wedge \bigvee_{K \in \Lambda, K \gtrsim L} \forall y_1, \ldots, y_m \bigwedge_{K' \in \Lambda, K \gtrsim \overline{K'}} \overline{K'} \neq L\gamma' \big),$$

where $\mathit{lhs}(L)$ and $\mathit{rhs}(L)$ denote the left-hand side and right-hand side of the literal L, $u \neq v$ is to be taken as an abbreviation for $u \prec v \vee u \succ v$, $\{x_1, \ldots, x_n\} = \mathit{Var}(\Gamma) \setminus \mathit{Var}(C)$ and $\{y_1, \ldots, y_m\}$ are the variables occurring in all the K'.[13] The following is now obvious:

[13] If $\mathit{lhs}(L) \succ \mathit{rhs}(L)$ for every $L \in \Gamma$ then all ordering conditions become true, so it is sufficient to solve the remaining disunifiability problem [Com91].

Theorem 32 (Validity in the Limit Model) *Let $\succ$ be a reduction ordering that is total on ground Σ-terms, such that for any ground term u there are only finitely many ground terms $v \prec u$ which can be enumerated effectively. Let* **B** *be a finite exhausted branch of a limit tree of a fair derivation with respect to $\succ$. Then it is decidable whether $R^{\star}_{\mathbf{B}} \models s \approx t$ for any ground literal $s \approx t$.*

11 Conclusions

Our main result is the completeness of the new $\mathcal{ME}$+Sup calculus. On the theoretical side, we plan to investigate how it can be exploited to obtain decision procedures for fragments of first-order logic that are beyond the scope of current superposition or instance-based methods. Ultimately, we will need an implementation to see how the labelling function is best exploited in practice for general refutational theorem proving.

Acknowledgements. We thank the reviewers for their critical and helpful remarks, which helped a lot to improve the presentation of this paper. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

A Proof of Lemma 10

Proof For convenience we abbreviate $R := R_{(C\cdot\Gamma;\gamma)}$ below. Assume that all preconditions of the lemma statement hold. With (i) then, by Definitions 5 and 2, $\Gamma\gamma$ consists of split rewrite literals, Λ produces Γ and Λ produces $\Gamma\gamma$ by the same literals, and $(Var(C) \cap Var(\Gamma))\gamma$ is irreducible wrt. R. We have to show

(1) $\Gamma'\gamma$ consists of split rewrite literals,
(2) Λ produces Γ' and Λ produces $\Gamma'\gamma$ by the same literals, and
(3) $(Var(C') \cap Var(\Gamma'))\gamma$ is irreducible wrt. R.

The proof of (1) follows easily from inspection of the ι_{Base} inference rules. Each inference rule requires that the (instantiated) constraints in the constrained clauses in the premise consist of split rewrite literals. Furthermore, U-Sup-Neg, U-Sup-Pos and U-Res, as the only rules that add new rewrite literals, come with conditions that force that for the new rewrite literals.

Let σ ($\sigma\pi$ in case of Sup-Pos- or Sup-Neg inference) be the mgu used in the ι_{Base} inference. Assume σ ($\sigma\pi$) is idempotent, which is the case with usual unification algorithms. Because γ gives a ground instance of the given ι_{Base} inference, γ must be a unifier for the same terms as σ ($\sigma\pi$). Because σ ($\sigma\pi$) is a most general unifier, there is a substitution δ such that $\gamma = \sigma\delta$ ($\gamma = \sigma\pi\delta$). With the idempotency of σ ($\sigma\pi$) we get $\gamma = \sigma\delta = \sigma\sigma\delta = \sigma\gamma$ ($\gamma = \sigma\pi\delta = \sigma\pi\sigma\pi\delta = \sigma\pi\gamma$).

It remains to show (2) and (3). We first establish some useful *facts*:

(i) Λ produces $\Gamma\sigma$ ($\Gamma\sigma\pi$) and Λ produces $\Gamma\sigma\gamma$ ($\Gamma\sigma\pi\gamma$) by the same literals.
Proof: Consider an arbitrary literal $L \in \Gamma$ and suppose that $K \in \Lambda$ produces L and $L\gamma$ in Λ. If K didn't produce $L\sigma$ in Λ then there would be a $K' \in \Lambda$ with $K \gtrsim \overline{K'} \gtrsim L\sigma$. With $\gamma = \sigma\delta$ and by transitivity of $\gtrsim$ we would get $K \gtrsim \overline{K'} \gtrsim L\gamma$, and so K would not produce $L\gamma$ either. With $\gamma = \sigma\gamma$ obtained above the second claim is trivial.
The proof that Λ produces $\Gamma\sigma\pi$ and Λ produces $\Gamma\sigma\pi\gamma$ by the same literals is the same after replacing σ by $\sigma\pi$.

(ii) For every term t, if $Var(t)\gamma$ is irreducible wrt. R then $Var(t\sigma)\gamma$ ($Var(t\sigma\pi)\gamma$) is irreducible wrt. R.
Proof: Let t be a term and suppose $Var(t)\gamma$ is irreducible wrt. R. Chose a variable $y \in Var(t\sigma)$ arbitrarily. It suffices to show that $y\gamma$ is irreducible wrt. R. With $y \in Var(t\sigma)$, y must occur in a term $x\sigma$, for some variable $x \in Var(t)$ ($y = x$ is possible). With y being a subterm of $x\sigma$, $y\gamma$ is a subterm of $x\sigma\gamma$. With the identity $\gamma = \sigma\gamma$ above we get that $y\gamma$ is a subterm of $x\gamma$. We assumed $Var(t)\gamma$ irreducible wrt. R. With $x \in Var(t)$, $x\gamma$ is irreducible wrt. R, and it is clear that its subterm $y\gamma$ then irreducible wrt. R, too.
The proof that $Var(t\sigma\pi)\gamma$ is irreducible wrt. R is the same after replacing σ by $\sigma\pi$.

(iii) $(Var(C\sigma) \cap Var(\Gamma\sigma))\gamma$ is irreducible wrt. R.
Proof: From above we know that $(Var(C) \cap Var(\Gamma))\gamma$ is irreducible wrt. R. If $x \in Var(C) \cap Var(\Gamma)$ take $t = x$ and conclude with fact (ii) that $Var(x\sigma)\gamma$ is irreducible wrt. R. Because this holds for every $x \in Var(C) \cap Var(\Gamma)$ we get that $Var((Var(C) \cap Var(\Gamma))\sigma)\gamma$ is irreducible wrt. R. The next step is to show $Var((Var(C) \cap Var(\Gamma))\sigma) = Var(C\sigma) \cap Var(\Gamma\sigma)$. The claim then follows immediately.
It is not difficult to see $Var((Var(C) \cap Var(\Gamma))\sigma) \subseteq Var(C\sigma) \cap Var(\Gamma\sigma)$. We are going to exclude the possibility that there is a variable y in the right set but not in the left set of the inequality. The existence of such a variable y can only be explained with two variables $x_C \in Var(C) \setminus Var(\Gamma)$ and $x_\Gamma \in Var(\Gamma) \setminus Var(C)$ such that $y \in Var(x_C\sigma)$ and $y \in Var(x_\Gamma\sigma)$. However, recall that σ (or $\sigma\pi$) is a unifier between terms in the clause part only of the right premise and possibly the left premise. Also, we are always taking fresh variants of a premise in inference rules. Together this entails that extraneous variables x_Γ cannot be moved by σ (or $\sigma\pi$), and that x_Γ is not in the codomain of σ. Under these (safe) assumptions then we can conclude that the claimed variable y does not exist.
The above argumentation can be used in the same way to conclude that $(Var(C\sigma\pi) \cap Var(\Gamma\sigma\pi))\gamma$ is irreducible wrt. R.

To prove (2) and (3) we carry out a case analysis with respect to the ι_{Base} inference rule applied.

Case a: Ref.
In case of a Ref inference let the premise be $s \not\approx t \vee D \cdot \Gamma$ and the conclusion $(D \cdot \Gamma)\sigma$. Then (2) follows directly from fact (i). Regarding (3), we already know that $(Var(s \not\approx t \vee D) \cap Var(\Gamma))\gamma$ is irreducible wrt. R. By fact (iii) then $(Var((s \not\approx t \vee D)\sigma) \cap Var(\Gamma\sigma))\gamma$ is irreducible wrt. R. The subset $(Var(D\sigma) \cap Var(\Gamma\sigma))\gamma$ then is trivially irreducible wrt. R, too, which proves (3).

Case b: U-Sup-Neg.
In case of a U-Sup-Neg inference let the left premise be $l \to r$, the right premise $C \cdot \Gamma = s[u]_p \approx t \vee D \cdot \Gamma$ and the conclusion $C' \cdot \Gamma' = (s[r]_p \approx t \vee D \cdot (\Gamma, l \to r))\sigma$. First we show (2), i.e., that Λ produces $(\Gamma \cup \{l \to r\})\sigma$ and Λ produces $(\Gamma \cup \{l \to r\})\sigma\gamma$ by the same literals. With respect to the subsets $\Gamma\sigma$ and $\Gamma\sigma\gamma$ the claim follows from fact (i). With respect to $(l \to r)\sigma$ and $(l \to r)\sigma\gamma$ recall we are given that $(l \to r)\sigma$ generates $(l \to r)\gamma$ in $R_{\Lambda \vdash \Phi}$, and hence $l \to r \in R$. By Lemma 7-(i) then Λ produces $(l \to r)\gamma$, that is, some literal $K \in \Lambda$ produces $(l \to r)\gamma$ in Λ. It remains to show that K produces $(l \to r)\sigma$ in Λ. With K producing $(l \to r)\gamma$ in Λ, and $(l \to r)\gamma$ being an instance of $(l \to r)\sigma$ the proof is similar to the one of fact (i) above and is omitted.

To prove (3) we show that $(Var((s[r]_p \approx t \vee D)\sigma) \cap Var((\Gamma \cup \{l \to r\})\sigma))\gamma$ is irreducible wrt. R. Fact (iii) above only gives us that $(Var((s[u]_p \approx t \vee D)\sigma) \cap Var(\Gamma\sigma))\gamma$ is irreducible wrt. R. To get the desired result from that it is enough to show that $Var(r\sigma)\gamma$

and $Var(l\sigma)\gamma$ are irreducible wrt. R, because only $r\sigma$ and $l\sigma$ (via $(l \to r)\sigma$) can contribute additional variables beyond those in $Var((s[u]_p \approx t \vee D)\sigma) \cap Var(\Gamma\sigma)$.

Regarding $Var(r\sigma)\gamma$, with fact (ii) it suffices to show that $Var(r)\gamma$ is irreducible wrt. R. With $(l \to r)\gamma$ generating $(l \to r)\gamma$ in $R_{\Lambda \vdash \Phi}$ it is impossible that $r\gamma$ (and $l\gamma$) are reducible wrt. $R_{(l \to r)\gamma}$. With $l\gamma \succ r\gamma$ it is clear that $(l \to r)\gamma$ cannot reduce $r\gamma$, nor can any other rule greater than $(l \to r)\gamma$. This shows that $r\gamma$ is irreducible even wrt. R.

Regarding $Var(l\sigma)\gamma$ we use a different argumentation. From the lemma precondition (ii-a) we know that the ground closure $(l \approx r \vee C'' \cdot \Gamma''; \gamma)$ generates $(l \to r)\gamma$ in $R_{\Lambda \vdash \Phi}$. Hence, trivially, $(l \to r)\gamma \in R_{\Lambda \vdash \Phi}$. Trivially, $(l \to r)\gamma$ reduces $l\gamma$, but no other rule in $R_{\Lambda \vdash \Phi}$ can reduce $l\gamma$. This is, because that rule would either have to be smaller, and so $(l \to r)\gamma$ would not be generated, or its left hand side would have to be equal to $l\gamma$, but with $(l \to r)\gamma \in R_{\Lambda \vdash \Phi}$ that rule would be reducible by the (smaller) rule $(l \to r)\gamma$, thus not generated, and thus not be in $R_{\Lambda \vdash \Phi}$. Because $R \subseteq R_{\Lambda \vdash \Phi}$ by construction, the only rule in R that can reduce $l\gamma$ thus is $(l \to r)\gamma$.

An important detail now is that the target term u unified with l by σ in the non-ground inference is not a variable. Every variable $x \in Var(l\sigma)$ therefore must be a proper subterm of $l\sigma$. And so $x\gamma$ is a proper subterm of $l\sigma\gamma$ $(= l\gamma)$. But then $x\gamma$ can be reducible wrt. R only by rules in R that are strictly smaller than $(l \to r)\gamma$, which, as argued, do not exist. In conclusion, $x\gamma$, and hence $Var(l\sigma)$ is irreducible wrt. R (and also irreducible wrt. $R_{(l \to r)\gamma}$).

Case c: U-Sup-Pos.
The proof for the case of a U-Sup-Pos inference is essentially the same and is omitted.

Case d: U-Res.
In case of a U-Res inference let the left premise be $\neg A$, the right premise $C \cdot \Gamma = s \approx t \vee D \cdot \Gamma$ and the conclusion $C' \cdot \Gamma' = (D \cdot (\Gamma, s \not\approx t))\sigma$. First we show that Λ produces $(\Gamma \cup \{s \not\approx t\})\sigma$ and Λ produces $(\Gamma \cup \{s \not\approx t\})\sigma\gamma$ by the same literals. With respect to the subsets $\Gamma\sigma$ and $\Gamma\sigma\gamma$ this follows from fact (i).

With respect to $(s \not\approx t)\sigma$ and $(s \not\approx t)\sigma\gamma$ recall that by the lemma precondition (ii-c) we know that $s\gamma$ and $t\gamma$ are irreducible wrt. R. The ordering on ground closures and ground rewrite literals is defined (cf. Section 7) in such a way that $(C \cdot \Gamma; \gamma) \succ (s \to t)\gamma$ whenever C contains an (the) equation $s \approx t$. By Lemma 7-(ii) then Λ produces $(s \not\approx t)\gamma$, that is, some literal $K \in \Lambda$ produces $(s \not\approx t)\gamma$ in Λ. To complete the proof of (2) it remains to be shown that K produces $(s \not\approx t)\sigma$ in Λ. With K producing $(s \not\approx t)\gamma$ in Λ, and $(s \not\approx t)\gamma$ being an instance of $(s \not\approx t)\sigma$ the proof is again similar to the one of fact (i) above and is omitted.

We still need to show (3), that $(Var(D\sigma) \cap Var((\Gamma \cup \{l \not\approx r\})\sigma))\gamma$ is irreducible wrt. R. Fact (iii) above only gives us that $(Var((s \approx t \vee D)\sigma) \cap Var(\Gamma\sigma))\gamma$ is irreducible wrt. R. But only $r\sigma$ and $l\sigma$ (via $(l \not\approx r)\sigma$) can contribute additional variables beyond those in $Var((s \approx t \vee D)\sigma) \cap Var(\Gamma\sigma)$. It is therefore enough to show that $Var(r\sigma)\gamma$ and $Var(l\sigma)\gamma$ are irreducible wrt. R. This however follows immediately from the fact above that $s\gamma$ and $t\gamma$ are irreducible wrt. R and fact (ii).

Case e: Sup-Neg.
In case of a Sup-Neg inference the left premise is $l \approx r \vee C'' \cdot \Gamma''$, the right premise is $C \cdot \Gamma = s[u]_p \approx t \vee D \cdot \Gamma$ and the conclusion is $C' \cdot \Gamma' = (s[r]_p \approx t \vee D \vee C'' \cdot (\Gamma, \Gamma''))\sigma\pi$. Recall we are given that $(l \approx r \vee C'' \cdot \Gamma''; \gamma)$ generates $(l \to r)\gamma$ in $R_{\Lambda \vdash \Phi}$. By definition then, $(l \approx r \vee C'' \cdot \Gamma''; \gamma)$ is a relevant closure wrt. $(\Lambda, R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)})$, which means that Λ produces Γ'' and Λ produces $\Gamma''\gamma$ by the same literals, and that $(Var(l \approx r \vee C'') \cap Var(\Gamma''))\gamma$ is irreducible wrt. $R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$.

For (2) we need to show that Λ produces $(\Gamma \cup \Gamma'')\sigma\pi$ and Λ produces $(\Gamma \cup \Gamma'')\sigma\pi\gamma$ by the same literals. With respect to the subsets $\Gamma\sigma\pi$ and $\Gamma\sigma\pi\gamma$ the claim follows again from

fact (i). With respect to the subsets $\Gamma''\sigma\pi$ and $\Gamma''\sigma\pi\gamma$ we can use the same argumentation as in fact (i) above, but starting from the fact that Λ produces Γ'' and Λ produces $\Gamma''\gamma$ by the same literals, yielding that Λ produces $\Gamma''\sigma\pi$ and Λ produces $\Gamma''\sigma\pi\gamma$ by the same literals.

Before proceeding with (3), proper, recall that $(l \approx r \vee C'' \cdot \Gamma''; \gamma)$ generates $(l \to r)\gamma$ in $R_{\Lambda \vdash \Phi}$, thus $(l \to r)\gamma \in R$ (because in ground Sup-Neg inferences the left premise always smaller than the right premise). It can be shown that the only rule in $R \setminus R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$ that can rewrite $(l \approx r \vee C'')\gamma$ is $(l \to r)\gamma$ itself: rewriteability by any other rule would give a contradiction to the maximality of $(l \approx r)\gamma$ in $(l \approx r \vee C'')\gamma$ or that rule could not have been generated, as its left-hand side would be reducible by $(l \to r)\gamma$. Moreover, $(l \to r)\gamma$ can rewrite $(l \approx r \vee C'')\gamma$ only at topmost positions of greater sides wrt. $\succ$ of positive equations in $(l \approx r \vee C'')\gamma$. All other terms in $(l \approx r \vee C'')\gamma$ are irreducible wrt. $R \setminus R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$. In particular, as $l\gamma \succ r\gamma$, $r\gamma$ is irreducible wrt. $R \setminus R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$.

For (3) we need to show that $(Var((s[r]_p \approx t \vee D \vee C'')\sigma\pi) \cap Var((\Gamma \cup \Gamma'')\sigma\pi))\gamma$ is irreducible wrt. R.

The first sub-proof is to show that $(Var(C''\sigma\pi) \cap Var(\Gamma''\sigma\pi))\gamma$ is irreducible wrt. R. From above we (only) know so far that $(Var(l \approx r \vee C'') \cap Var(\Gamma''))\gamma$ and hence $(Var(C'') \cap Var(\Gamma''))\gamma$ is irreducible wrt. $R_{(l \approx r \vee C'' \cdot \Gamma'')\gamma}$. Using the results from further above, the only rule in $R \setminus R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$ that can reduce $C''\gamma$ is $(l \to r)\gamma$, and only at top-level positions of bigger sides of positive equations in $C''\gamma$. Let $(x_1 \approx t_1 \vee \cdots x_n \approx t_n)\gamma$ be the biggest subclause of $C''\gamma$ such that $l\gamma = x_1\gamma = \cdots = x_n\gamma$. The substitution π must now be chosen as an mgu for the terms $l\sigma, x_1\sigma, \ldots, x_n\sigma$. Observe this is possible because $r\gamma \succ l\gamma$ and $l\gamma \succ x_1\gamma, \ldots, l\gamma \succ x_n\gamma$. Because $l\sigma$ is not a variable, none of the terms $x_1\sigma\pi, \ldots, x_n\sigma\pi$ is a variable either. Thus every variable occurring in one of these terms must be a proper subterm of (the same) term $l\sigma\pi$. This entails that every term in $Var(x_i\sigma\pi)\gamma$ is a proper subterm of $l\sigma\pi\gamma$ $(= l\gamma)$, for all $i = 1, \ldots, n$ and hence irreducible by $(l \to r)\gamma$. Because of the choice as the biggest subclause above, there is no term in $Var(C''\sigma\pi)\gamma$ left that can be reduced by $l \to r$, or any other rule in $R \setminus R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$. It follows trivially that $(Var(C''\sigma\pi) \cap Var(\Gamma''\sigma\pi))\gamma$ is irreducible wrt. $R \setminus R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$. To complete the first sub-proof it remains to be shown that $(Var(C''\sigma\pi) \cap Var(\Gamma''\sigma\pi))\gamma$ is irreducible wrt. $R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$. This, however, can be done with the same arguments as in fact (iii), however using the rewrite system $R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$ instead of R, and starting from $(Var(l \approx r \vee C'') \cap Var(\Gamma''))\gamma$ being irreducible wrt. $R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$.

Fact (iii) above gives us that $(Var((s[u]_p \approx t \vee D)\sigma\pi) \cap Var(\Gamma\sigma\pi))\gamma$ is irreducible wrt. R. This result can be combined with the result from the first sub-proof to obtain that $(Var((s[u]_p \approx t \vee D \vee C'')\sigma\pi) \cap Var((\Gamma \cup \Gamma'')\sigma\pi))\gamma$ is irreducible wrt. R. The arguments for that are the same as in fact (iii) and assume that the substitution $\sigma\pi$ does not move extraneous variables in constraints and that these variables do not occur in the codomain of $\sigma\pi$.

But this result is not quite what we need. For (3) we need to show that $(Var((s[r]_p \approx t \vee D \vee C'')\sigma\pi) \cap Var((\Gamma \cup \Gamma'')\sigma\pi))\gamma$ is irreducible wrt. R. Observe that any additional variables in the latter set must stem from $Var(r\sigma\pi)$. It is therefore enough to show that $(Var(r\sigma\pi) \cap Var(\Gamma''\sigma\pi))\gamma$ is irreducible wrt. R.

From further above we know that $r\gamma$ is irreducible wrt. $R \setminus R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$. Because every term in $Var(r)\gamma$ is a (possibly non-proper) subterm of $r\gamma$, $Var(r)\gamma$ is irreducible wrt. $R \setminus R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$, too. Also from above we know that $(Var(l \approx r \vee C'') \cap Var(\Gamma''))\gamma$ is irreducible wrt. $R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$. With the same arguments as in fact (iii) conclude that $(Var((l \approx r \vee C'')\sigma\pi) \cap Var(\Gamma''\sigma\pi))\gamma$ is irreducible wrt. $R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$. Trivially, $(Var(r\sigma\pi) \cap Var(\Gamma''\sigma\pi))\gamma$ is irreducible wrt. $R_{(l \approx r \vee C'' \cdot \Gamma''; \gamma)}$. Together, $(Var(r\sigma\pi) \cap Var(\Gamma''\sigma\pi))\gamma$ is irreducible wrt. R. This completes the proof of this case.

Case f: Sup-Pos.
In case of a Sup-Pos inference the proof is exactly the same as in case (e).

Case g: Eq-Fact.
Finally, in case of a Eq-Fact inference observe that $Var(l \approx t \vee r \not\approx t \vee C) \subseteq Var(l \approx r \vee s \approx t \vee C)$. The proof then follows easily with facts (i) and (iii). □

References

[Bau07] Peter Baumgartner. Logical engineering with instance-based methods. In Frank Pfenning, editor, *CADE-21 – The 21st International Conference on Automated Deduction*, volume 4603 of *Lecture Notes in Artificial Intelligence*, pages 404–409. Springer, July 2007.

[BG98] Leo Bachmair and Harald Ganzinger. Chapter 11: Equational Reasoning in Saturation-Based Theorem Proving. In Wolfgang Bibel and Peter H. Schmitt, editors, *Automated Deduction. A Basis for Applications*, volume I: Foundations. Calculi and Refinements, pages 353–398. Kluwer Academic Publishers, 1998.

[BGW94] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Refutational theorem proving for hierarchic first-order theories. *Applicable Algebra in Engineering, Communication and Computing*, 5(3/4):193–212, April 1994.

[BN98] F. Baader and T. Nipkow. *Term Rewriting and all that*. Cambridge University Press, Cambridge, 1998.

[BT03] Peter Baumgartner and Cesare Tinelli. The Model Evolution Calculus. In Franz Baader, editor, *CADE-19 – The 19th International Conference on Automated Deduction*, volume 2741 of *Lecture Notes in Artificial Intelligence*, pages 350–364. Springer, 2003.

[BT05] Peter Baumgartner and Cesare Tinelli. The model evolution calculus with equality. In Robert Nieuwenhuis, editor, *CADE-20 – The 20th International Conference on Automated Deduction*, volume 3632 of *Lecture Notes in Artificial Intelligence*, pages 392–408. Springer, 2005.

[Com91] Hubert Comon. Disunification: a survey. In Jean-Louis Lassez and Gordon Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*. MIT Press, 1991.

[GK03] H. Ganzinger and K. Korovin. New directions in instantiation-based theorem proving. In *Proc. 18th IEEE Symposium on Logic in Computer Science,(LICS'03)*, pages 55–64. IEEE Computer Society Press, 2003.

[GK04] H. Ganzinger and K. Korovin. Integrating equational reasoning into instantiation-based theorem proving. In *Computer Science Logic (CSL'04)*, volume 3210 of *Lecture Notes in Computer Science*, pages 71–84. Springer, 2004.

[JW07] Swen Jacobs and Uwe Waldmann. Comparing instance generation methods for automated reasoning. *J. Autom. Reason.*, 38(1-3):57–78, 2007.

[Kor08] Konstantin Korovin. iprover - an instantiation-based theorem prover for first-order logic (system description). In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 2008*, volume 5195 of *Lecture Notes in Computer Science*, pages 292–298. Springer, 2008.

[Kor09] Konstantin Korovin. Instantiation-based automated reasoning: From theory to practice. In Renate A. Schmidt, editor, *Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings*, volume 5663 of *Lecture Notes in Computer Science*, pages 163–166. Springer, 2009.

[LM09] Chrystopher Lynch and Ralph Eric McGregor. Combining instance generation and resolution. In Silvio Ghilardi and Roberto Sebastiani, editors, *Frontiers of Combining Systems (FroCoS 2009)*, volume 5749 of *LNAI*, pages 304–318. Springer Verlag, September 2009.

[NR95] Robert Nieuwenhuis and Albert Rubio. Theorem Proving with Ordering and Equality Constrained Clauses. *Journal of Symbolic Computation*, 19:321–351, 1995.

[PZ00] David A. Plaisted and Yunshan Zhu. Ordered Semantic Hyper Linking. *Journal of Automated Reasoning*, 25(3):167–217, 2000.

[WSH+07] Christoph Weidenbach, Renate Schmidt, Thomas Hillenbrand, Rostislav Rusev, and Dalibor Topic. System description: Spass version 3.0. In Frank Pfenning, editor, *CADE-21 — 21st International Conference on Automated Deduction*, volume 4603 of *Lecture Notes in Artificial Intelligence*, pages 514–520. Springer, 2007.

[ZSM05] Ting Zhang, Henny B. Sipma, and Zohar Manna. The decidability of the first-order theory of term algebras with Knuth-Bendix order. In Robert Nieuwenhuis, editor, *20th International Conference on Automated Deduction (CADE'05)*, volume 3632 of *Lecture Notes in Artificial Intelligence*, pages 131–148. Springer-Verlag, 2005.