# Superposition and Model Evolution Combined

Peter Baumgartner
NICTA*and Australian National University, Canberra, Australia
Peter.Baumgartner@nicta.com.au

Uwe Waldmann
MPI für Informatik, Saarbrücken, Germany
uwe@mpi-inf.mpg.de

May 18, 2009

**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 both—rather complementary—calculi in a single framework. For instance, 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.

## 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. The inference rules are controlled by a rather 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 research-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

*NICTA is funded by the Australian Government's *Backing Australia's Ability* initiative.

stand rather separated. They provide decision procedure 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.

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, which is interesting, e.g., to obtain a decision procedure for function-free clause logic. 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) | $x \leq z \vee \neg(x \leq y) \vee \neg(y \leq z)$ | (4) | $\mathsf{select}(\mathsf{store}(a, i, e), i) \approx e$ |
| (2) | $x \leq y \vee y \leq x$ | (5) | $\mathsf{select}(\mathsf{store}(a, i, e), j) \approx \mathsf{select}(a, j) \vee i \approx j$ |
| (3) | $x \approx y \vee \neg(x \leq y) \vee \neg(y \leq x)$ | (6) | $i \leq j \vee \neg(\mathsf{select}(\mathsf{a0}, i) \leq \mathsf{select}(\mathsf{a0}, j))$ |

The clauses (1)-(3) axiomatize a total order, clauses (4)-(5) axiomatize arrays, and clause (6) says that the array a0 is sorted and that there are no duplicates in a0 (the converse of (6), $\neg(i \leq j) \vee \mathsf{select}(\mathsf{a0}, i) \leq \mathsf{select}(\mathsf{a0}, j)$, is entailed by (1)-(3),(6)). This clause set is satisfiable, but Superposition equipped with standard redundancy criteria (with or without selection of negative literals) does not terminate on these clauses. This is, essentially, because the length of the clauses derived cannot be bounded. The clauses (1) and (2) are enough to cause non-termination, and $\mathcal{ME}_{\mathrm{E}}$ does not terminate on (1)-(6) either. However, $\mathcal{ME}$+Sup does terminate when all $\leq$-atoms are labelled as "split atoms" and all other atoms are "superposition atoms".[1] Intuitively, the $\mathcal{ME}$-part of $\mathcal{ME}$+Sup takes care of computing a model for the split atoms through a *context*, the main data structure of $\mathcal{ME}$ to represent interpretations, and the Superposition part of $\mathcal{ME}$+Sup takes care of (implicitly) computing a model for the superposition atoms. Of course, clauses like the above are typically embedded in a larger specification and a $\mathcal{ME}$+Sup prover might not behave as well then. Yet, termination on the above clauses demonstrates that only finitely many inferences among (1)-(6) are needed, this way removing search space.

To demonstrate how $\mathcal{ME}$+Sup can be used to effectively provide a new splitting rule for Superposition consider the clauses (1) and (2) from above. Let us now "split" clause (1) into two non-variable disjoint clauses by introducing a name s:

$$\text{(1a)} \quad x \leq z \vee \neg(x \leq y) \vee \neg\mathsf{s}(y, z) \qquad \text{(1b)} \quad \mathsf{s}(y, z) \vee \neg(y \leq z)$$

[1] In general, split atoms can be equations, too, and the signatures of the split and the superposition atoms need not be disjoint. We intended to keep the examples simple.

Now declare all $\leq$-atoms as superposition atoms and all $\mathsf{s}$-atoms as split atoms. Further, all $\mathsf{s}$-atoms must be strictly greater than all $\leq$-atoms (this can be achieved using standard orderings and using a two-sorted signature). In effect then, resolution and factoring inferences are all blocked on clauses that contain $\mathsf{s}$-literals, as the usual maximality restrictions for resolution and factorisation apply in $\mathcal{ME}$+Sup, too. Therefore, only factorisation is applicable, to clause (2), yielding $x \leq x$. The only inference rule that is applicable now is Neg-U-Res, which gives $\neg(y \leq z) \cdot \neg\mathsf{s}(y, z)$. (This is a *constrained clause*, a pair $C \cdot \Gamma$, where $C$ is a clause and the constraints $\Gamma$ are split atoms or their negation.) That is, $\mathsf{s}(y, z)$ has been shifted into the constraint part, put aside for later processing by $\mathcal{ME}$ rules. The literal $\neg(y \leq z)$ is now maximal in $\neg(y \leq z) \cdot \neg\mathsf{s}(y, z)$, and resolution between this clause and (2) gives $z \leq y \cdot \neg\mathsf{s}(y, z)$. Similarly, resolution between $\neg(y \leq z) \cdot \neg\mathsf{s}(y, z)$ and $x \leq x$ gives the constrained empty clause $\Box \cdot \neg\mathsf{s}(x, x)$. This does not make a refutation, because a model that assigns true to $\mathsf{s}(x, x)$, and hence falsifies the constraint, has not been excluded. Indeed, to constrained empty clauses the $\mathcal{ME}$-style split rule is applicable, resulting in two cases (contexts), with $\mathsf{s}(x, x)$ and $\neg\mathsf{s}(x, x)$, respectively. Notice this is a *non*-ground splitting. The derivation stops at this point, as no inference rule is applicable, and $\mathsf{s}(x, x)$ specifies a model. The other case with $\neg\mathsf{s}(x, x)$ can be used to derive the empty clause $\Box \cdot \emptyset$, which stands for "false".

**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. 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 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.

## 2 Formal Preliminaries

We consider signatures $\Sigma$ 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 $\Sigma$. Terms (over $\Sigma$ and $X$) are defined as usual. If $t$ is a term we denote by $\mathcal{V}ar(t)$ the set of $t$'s variables. A term $t$ is *ground* iff $\mathcal{V}ar(t) = \emptyset$. 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 $i = 1, \ldots, n$. A substitution $\sigma$ is applied to a term $t$, written as $t\sigma$, by simultaneously applying $\sigma$ to the variables in $t$. A *renaming* is a substitution that is a bijection of $X$ onto itself. We write $s \gtrsim t$, iff there is a substitution $\sigma$ such that $s\sigma = t$.[2] 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 $\rho$ such that $s\rho = t$. We write $s \gtrsim\!\!\!\!\!\sim\ t$ if $s \gtrsim t$ but

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

$s \not\preceq t$. 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 a non-equational literal like $P(t_1, \ldots, t_n)$, which is meant to stand for the equation $P(t_1, \ldots, t_n) \approx \mathrm{tt}$, where tt is a fresh constant that is smaller than all other terms, and similarly for negative literals. 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 $\Sigma$-terms.[3] 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, ground 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 or a clause (set), which stands for the set of all its ground instances (of all clauses in the set). An *E-interpretation* is an interpretation that is also a congruence relation on the ground terms. If $I$ is an

[3] A *reduction ordering* is a strict partial ordering that is well-founded and is closed unter 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 $\delta$.

interpretation, we denote by $I^\star$ the smallest congruence relation on all ground terms that includes $I$, which 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-know that any left-reduced (and hence any fully reduced) ordered rewrite system $R$ is convergent,[4] 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. $R$.

**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*.[5] 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 the term "literal" to refer to equational literals as introduced earlier or to rewrite literals.

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

[5] 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.

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. A non-equational literal $P(t_1, \ldots, t_n)$ in a context stands for $P(t_1, \ldots, t_n) \to \mathrm{tt}$, and similarly for negative literals. Where $L$ is a rewrite literal and $\Lambda$ a context, we write $L \in_\sim \Lambda$ if $L$ is a variant of a literal in $\Lambda$. A rewrite literal $L$ is *contradictory with a context* $\Lambda$ iff $\overline{L} \in_\sim \Lambda$. A context $\Lambda$ is *contradictory* iff it contains a rewrite literal that is contradictory with $\Lambda$. For instance, if $\Lambda = \{f(x) \to a, f(x) \not\to x\}$ then $f(y) \not\to a$ and $f(y) \to y$ are contradictory with $\Lambda$, while $f(a) \to a$, $a \not\to f(x)$ and $f(x) \to y$ 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 stands for its *produced* literals, defined as follows:

**Definition 3.1 (Productivity [BT03])** *Let $L$ be a rewrite literal and $\Lambda$ a context. A rewrite literal $K$* produces *$L$ in $\Lambda$ iff $K \gtrsim L$ and there is no $K' \in \Lambda$ such that $K \gtrsim \overline{K'} \gtrsim L$.*[6] *The context $\Lambda$* produces *$L$ iff it contains a literal $K$ that produces $L$ in $\Lambda$, and $\Lambda$* produces *a multiset $\Gamma$ of rewrite literals iff $\Lambda$ produces each $L \in \Gamma$.*

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

For the model construction in Section 7 we will need the set of positive ground rewrite rules produced by $\Lambda$, $\Pi_\Lambda := \{l \to r \mid \Lambda \text{ produces } l \to r \text{ and } l \to r \text{ is ground}\}$. For instance, if $\Lambda = \{f(x) \to x\}$ and $\Sigma$ consists of a constant $a$ and the unary function symbol $f$ then $\Pi_\Lambda = \{f(a) \to a, f(f(a)) \to f(a), \ldots\}$. We note that productivity of rewrite literals corresponding to *split* atoms only is relevant for the calculus.

## 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 $\Gamma$)*, 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 $C \cdot \Gamma \cup \{K\}$.[7]

Applying a substitution $\sigma$ to $C \cdot \Gamma$, written as $(C \cdot \Gamma)\sigma$, means to apply $\sigma$ to $C$ and all literals in $\Gamma$. A constrained clause $C \cdot \Gamma$ is *ground* iff both $C$ and $\Gamma$ are ground. If $\gamma$ is a substitution such that $(C \cdot \Gamma)\gamma$ is ground, then $(C \cdot \Gamma)\gamma$ is called a *ground instance* of $C \cdot \Gamma$ (via $\gamma$). For a set of constrained clauses $\Phi$, $\Phi^{\mathrm{gr}}$ is the set of all ground instances of all elements in $\Phi$.

Constraints are compared in a similar way as clauses by taking the multiset extension of a (any) total ordering on ground rewrite literals. Constrained clauses then are

[6] In [BT03] the first condition is replaced by the stronger condition "$K$ is an msg of $L$ in $\Lambda$", where $K$ is a *most specific generalization (msg) of $L$ in $\Lambda$* iff $K \gtrsim L$ and there is no $K' \in \Lambda$ such that $K \gtrsim K' \gtrsim L$. Working with the stated condition is somewhat simpler and achieves the same for all purposes.

[7] As will become clear later, literals $x \to t$ can never occur in constraints, because, in essence, paramodulation into variables is unnecessary.

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 total on ground constrained clauses. Observe that this definition has the desirable property that proper subsumption among constrained clauses is always order-decreasing (the subsuming constrained clause is smaller).

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 $L_1 \vee \cdots \vee L_m \vee l_1 \not\approx r_i \vee \cdots \vee l_k \not\approx r_k \vee l_{k+1} \approx r_{k+1} \vee \cdots \vee l_n \approx r_n$ 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 $\Gamma$ are evaluated wrt. a context $\Lambda$ in terms of productivity. The following definition makes this precise.

We say that a ground constraint $\Gamma$ *consists of split rewrite literals* iff $l \approx r$ is a split atom and $l \succ r$, for every $l \to r \in \Gamma$ or $l \not\to r \in \Gamma$. A possibly non-ground constraint $\Gamma$ *consists of split rewrite literals* if some ground instance of $\Gamma$ does.

**Definition 4.1 (Satisfiaction of Constrained Clauses)** *Let $C \cdot \Gamma$ be a ground constrained clause, $\Lambda$ a context, and $I$ an E-Interpretation. We say that* $\Lambda$ satisfies $\Gamma$ *and write $\Lambda \models \Gamma$ iff $\Gamma$ consists of split rewrite literals and $\Lambda$ produces $\Gamma$. We say that the pair* $(\Lambda, I)$ satisfies $C \cdot \Gamma$ *and write $\Lambda, I \models C \cdot \Gamma$ iff $\Lambda \not\models \Gamma$ or $I \models C$.*

The pair $(\Lambda, I)$ *satisfies* a possibly non-ground constrained clause (set) $F$, written as $\Lambda, I \models F$ iff $(\Lambda, I)$ satisfies all ground instances of (all elements in) $F$. For a set of constrained clauses $\Phi$ we say that $\Phi$ *entails* $C \cdot \Gamma$ *wrt.* $\Lambda$, and write $\Phi \models_\Lambda C \cdot \Gamma$ iff for every E-interpretation $I$ it holds $\Lambda, I \not\models \Phi$ or $\Lambda, I \models C \cdot \Gamma$.

The definitions above are also applied to pairs $(\Lambda, R)$, where $R$ is a rewrite system, by implicitly taking $(\Lambda, R^\star)$. Indeed, in the main applications of Definition 4.1 such a rewrite system $R$ will be determined by the model construction in Section 7 below.

**Example 4.2** Let $\Lambda = \{f(x) \to x, f(c) \not\to c\}$, $R = \{f(a) \to a, f(b) \to b\}$ and $C \cdot \Gamma = f(f(a)) \approx x \cdot f(x) \to x$. Let $\gamma_a = \{x \mapsto a\}$, $\gamma_b = \{x \mapsto b\}$ and $\gamma_c = \{x \mapsto c\}$. Suppose that all (ground) atoms are split atoms. Notice that $\Gamma\gamma_a$, $\Gamma\gamma_b$ and $\Gamma\gamma_c$ consist of split rewrite literals. Then, $R \models \Gamma\gamma_a$, as $\Lambda$ produces $\{f(a) \to a\}$ and so we need to check $R^\star \models f(f(a)) \approx a$, which is the case, to conclude $\Lambda, R \models (C \cdot \Gamma)\gamma_a$. As $R \models \Gamma\gamma_b$ but $R^\star \not\models f(f(a)) \approx b$ we have $\Lambda, R \not\models (C \cdot \Gamma)\gamma_a$. Finally, $\Lambda$ does not produce $\{f(c) \to c\}$, and with $\Lambda \not\models \Gamma\gamma_c$ it follows $\Lambda, R \models (C \cdot \Gamma)\gamma_c$ □

## 5 Inference Rules on Constrained Clauses

We are going to define several inference rules on constrained clauses, which will be embedded into the $\mathcal{ME}$+Sup calculus below.

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

where (i) $\sigma$ is a 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, which will be taken from a current context, and a constrained clause, which will be taken from a current clause set.

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

where (i) $\sigma$ is a 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) $\sigma$ is a 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.

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

where $\neg A$ is a pseudo literal $\neg x$ or a negative rewrite literal $l \not\to r$, and (i) $(s \approx t)\sigma$ is a split atom, (ii) $\sigma$ is a mgu of $A$ and $s \to t$, (iii) $(s \approx t)\sigma$ is a split atom, (iv) $t\sigma \not\succeq s\sigma$, and (v) $(s \approx t)\sigma$ 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 $C = x_1 \approx t_1 \vee \cdots \vee x_n \approx t_n$ a (possibly empty) clause of positive literals, where $x_i$ is a variable and $t_i$ a term, for all $i = 1, \ldots, n$. We say that a substitution $\pi$ *merges* $C$ *with* $l \approx r$ iff $\pi$ is an mgu of $l, x_1, \ldots, x_n$, $r\pi \not\succeq l\pi$, 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) $\sigma$ is a mgu of $l$ and $u$, (ii) $u$ is not a variable, (iii) $\pi$ merges $x_1 \approx t_1 \vee \cdots \vee x_n \approx t_n \subseteq C'\sigma$ with $(l \approx r)\sigma$, (iv) $\{x_1, \ldots, x_n\} \subseteq \mathcal{V}ar(\Gamma'\sigma)$, (v) $(l \approx r)\sigma$ 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) $t\sigma \not\succeq s\sigma$, and (ix) $(s \not\approx t)\sigma$ is maximal in $(s \not\approx t \vee C)\sigma$.

The need for merge substitutions is demonstrated in Example 7.5 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) $\sigma$ is a mgu of $l$ and $u$, (ii) $u$ is not a variable, (iii) $\pi$ merges $x_1 \approx t_1 \vee \cdots \vee x_n \approx t_n \subseteq C'\sigma$ with $(l \approx r)\sigma$, (iv) $\{x_1, \ldots, x_n\} \subseteq \mathcal{V}ar(\Gamma'\sigma)$, (v) $(l \approx r)\sigma$ 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 $(s \approx t)\sigma$ is a split atom then (viii-a) $(s \approx t)\sigma$ is maximal in $(s \approx t \vee C)\sigma$ else (viii-b) $t\sigma \not\succeq s\sigma$ and $(s \approx t)\sigma$ is strictly maximal in $(s \approx t \vee C)\sigma$.

Notice that $(s \approx t)\sigma$ 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 $\mathcal{C}\sigma\pi \not\succeq \mathcal{D}\sigma\pi$, where by $\mathcal{C}$ and $\mathcal{D}$ we mean their left and right premise, respectively.

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

where (i) $\sigma$ 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* $\iota$ is a set of inference rules. By an $\iota$ *inference* we mean an instance of an inference rule from $\iota$ such that all conditions are satisfied. An inference is *ground* if all its premises and the conclusion are ground.

The *base inference system* $\iota_{\text{Base}}$ consists of Ref, Fact, U-Sup-Neg, U-Sup-Pos, Neg-U-Res, Sup-Neg and Sup-Pos. If from a given $\iota_{\text{Base}}$ inference a ground $\iota_{\text{Base}}$ inference results by applying a substitution $\gamma$ to all premises and the conclusion, we call the resulting ground inference a *ground instance via* $\gamma$ *(of the* $\iota_{\text{Base}}$ *inference).* This is not always the case, as, e.g., ordering constraints can become unsatisfiable after application of $\gamma$. An important consequence of the ordering restrictions stated with the inference rules is that the conclusion of a ground $\iota_{\text{Base}}$ 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 $\Lambda$ is a context and $\Phi$ is a set of constrained clauses. The following

inference rules extend the inference rules $\iota_{\text{Base}}$ 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 Fact inference with a premise from $\Phi$.
- $C \cdot \Gamma$ is the conclusion of a U-Sup-Neg, U-Sup-Pos or Neg-U-Res inference with a right premise from $\Phi$ and a left premise $K \in \Lambda$ that produces $K\sigma$ in $\Lambda$, where $\sigma$ is the mgu used in that inference.
- $C \cdot \Gamma$ is the conclusion of a Sup-Neg or Sup-Pos inference with both premises from $\Phi$.

In each case the second or only premise of the underlying $\iota_{\text{Base}}$ 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 $\Lambda$. A Split inference is *productive* if $\Lambda$ produces $\Gamma$; 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 $\Lambda$ produces $\Gamma$ (in the sense of Definition 4.1). This is achieved by adding $\overline{K}$ to the current context. For example, if $\Lambda = \{P(a, y), \neg P(x, b)\}$ and $\square \cdot \Gamma = \square \cdot P(a, b)$ then a (productive) Split inference will give $\{P(a, y), \neg P(x, b), \neg P(a, b)\}$, which no longer produces $P(a, b)$. Intuitively, the calculus tries to "repair" the current context towards a model for a constrained empty clause.

Notice that a Split inference can never add a rewrite to a context that already contains a variant of it or its complement, as this would contradict condition (iii).[8] 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 $\Lambda$ are the *closing literals*. A sequent $\Lambda \vdash \Phi$ is *closed* if $\Phi$ contains $\square \cdot \emptyset$. The purpose of Close is to abandon a sequent that cannot be "repaired".

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

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

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 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, Neg-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. 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 7.1 (Relevant Instance wrt. $(\Lambda, R)$)** *Let $\Lambda$ be a context, $R$ a rewrite system, and $\gamma$ a ground substitution for a constrained clause $C \cdot \Gamma$. We say that $(C \cdot \Gamma)\gamma$*[9] *is a* relevant instance (of $C \cdot \Gamma$) wrt. $(\Lambda, R)$ *iff*

*(i)* $\Gamma\gamma$ *consists of rewrite split literals,*

*(ii)* $\Lambda$ *produces* $\Gamma$ *and* $\Lambda$ *produces* $\Gamma\gamma$ *by the same literals (see below), and*

*(iii)* $(\mathcal{V}ar(C) \cap \mathcal{V}ar(\Gamma))\gamma$ *is irreducible wrt. $R$.*

In the previous definition, item (ii) is to be understood to say that, for each $L \in \Gamma$, there is a literal $K \in \Lambda$ that produces both $L$ and $L\gamma$ in $\Lambda$.

Notice that in order for $C \cdot \Gamma$ to have relevant instances it is not necessary that $C \cdot \Gamma$ is taken from a specific clause set. Indeed, conclusions of inference rules may well have relevant instances but need not be contained in a clause set as long as inferences are universally redundant (see below). Notice also that for a clause with an empty constraint all its instances are relevant.

[9] Strictly speaking, the definition works with pairs $(C \cdot \Gamma, \gamma)$ instead of ground instances $(C \cdot \Gamma)\gamma$, but this causes no problems as $\gamma$ will always be clear from the context. Similarly in other definitions below.

**Example 7.2** If $\Lambda = \{P(x), a \to b, \neg P(b)\}$, $R = \{a \to b\}$ and $C \cdot \Gamma = x \approx b \vee x \approx d \cdot P(x)$ then the substitution $\gamma = \{x \mapsto a\}$ gives a ground instance that satisfies condition (ii) but not (iii). With the substitution $\gamma = \{x \mapsto c\}$ both (ii) and (iii) are satisfied, and with $\gamma = \{x \mapsto b\}$ the condition (ii) is not satisfied but (iii) is. If $\Lambda = \{P(a)\}$ then $\Box \cdot P(x)$ does not have relevant instances (although $\Lambda$ produces the ground constraint $P(a)$) because $\Lambda$ does not produce $P(x)$. The calculus needs to make sure that such "irrelevant" constrained clauses need not be considered, as (in particular) Close cannot be applied to, say, $\{P(a)\} \vdash \Box \cdot P(x)$ although $\{P(a)\}, \emptyset \not\models \Box \cdot P(x)$. $\Box$

For a given sequent $\Lambda \vdash \Phi$, where $\Phi$ does not contain $\Box \cdot \emptyset$, we define by induction on the clause ordering $\succ$ sets of rewrite rules $\epsilon_{\mathcal{C}}$ and $R_{\mathcal{C}}$, for every $\mathcal{C} \in \Phi^{\mathrm{gr}} \cup \Pi_\Lambda$. Here, for the purpose of comparing (positive) rewrite literals, $l \to r$ is taken as the constrained clause $l \approx r \cdot \bot$, 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 $(l \approx r \cdot \emptyset) \succ l \to r$ as $(l \approx r \cdot \emptyset) \succ (l \approx r \cdot \bot)$, as $\emptyset \succ \bot$.

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

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

Otherwise $\epsilon_{C \cdot \Gamma} = \emptyset$.

Finally, $R = \bigcup_{\mathcal{C}} \epsilon_{\mathcal{C}}$. If $\epsilon_{l \to r} = \{l \to r\}$ then we say that $l \to r$ *generates* $l \to r$ in $R$. If $\epsilon_{C \cdot \Gamma} = \{l \to r\}$ then we say that $C \cdot \Gamma$ *generates* $l \to r$ in $R$ via $C' \cdot \Gamma'$. 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 not difficult to show that $R$ is a left-reduced rewrite system and the rules contributed by $\Pi_\Lambda$ 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 $\epsilon_{C\cdot\Gamma}$ refers to the context $\Lambda$, which is fixed prior to the model construction, and the rewrite system $R_{C\cdot\Gamma}$ constructed so far. The definition can be seen to work in a hierarchical way, by first building the set of those constrained clauses from $\Phi^{\mathrm{gr}}$ whose constraints are produced in $\Lambda$, and then generating $R$ from that set, which involves checking irreducibility of substitutions wrt. $R_{C\cdot\Gamma}$.

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

*(i) if $l \to r \in R$ then $\Lambda$ produces $l \to r$, and*

*(ii) if $\mathcal{D} \succ l \to r$, and $l$ and $r$ are irreducible wrt. $R_\mathcal{D}$ then $\Lambda$ 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 $\Lambda$ produces $l \to r$ by definition of $\Pi_\Lambda$.

Concerning (ii), suppose that $l$ and $r$ are irreducible wrt. $R_\mathcal{D}$. If $l \to r$ were contained in $\Pi_\Lambda$, then either $l \to r$ would be generated in $R_\mathcal{D}$, but then $l$ would be reducible by $l \to r \in R_\mathcal{D}$, or $l \to r$ would not be generated in $R_\mathcal{D}$, but this is only possible if $l$ or $r$ is reducible by $R_{l\to r}$, and since $R_{l\to r} \subseteq R_\mathcal{D}$, it would again be reducible by $R_\mathcal{D}$. 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 $\Lambda$ not producing $l \to r$ conclude that $\Lambda$ produces $l \not\to r$. □

**Example 7.4** Let $\Lambda = \{a \to x, b \to c, a \not\to c\}$, $\Phi = \emptyset$ and assume that all equations are split atoms. With $a \succ b \succ c$ the induced rewrite system $R$ is $\{b \to c\}$. To see why, observe that the rule $a \to c$ is not included in $R$, as $\Lambda$ does not produce $a \to c$, and that $a \to b$, although produced in $\Lambda$, is reducible by the smaller rule $b \to c$. Had we chosen to omit in the definition of $\epsilon_{C\cdot\Gamma}$ the condition "$r$ is irreducible wrt. $R_{l\to r}$" [10] the construction would have given $R = \{a \to b, b \to c\}$. This leads to the undesirable situation that a constrained clause, say, $a \not\approx c \cdot \emptyset$ is falsified by $R^\star$. But the calculus cannot modify $\Lambda$ to revert this situation, and to detect the inconsistency (ordered) paramodulation into variables would be needed. □

**Example 7.5** Let $a \succ b \succ c$, $\Lambda = \{P(x), \neg P(b), \neg P(c)\}$ and $C \cdot \Gamma = y \approx b \vee x \approx c \cdot P(x)$ be the only clause in $\Phi$. Then the instance $a \approx b \vee a \approx c \cdot P(a)$ generates $a \to b$ in $R$. This is, because $a \approx b \vee a \approx c \cdot P(a)$ is relevant instance of $y \approx b \vee x \approx c \cdot P(x)$ wrt.

[10] 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.

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

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

where $\sigma = \{y \mapsto a\}$ and $\pi = \{x \mapsto a\}$. Then, $f(b) \not\approx f(b) \vee a \approx c \cdot P(a)$ is a relevant instance (of itself) wrt. $\Lambda$. 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 formal account. □

**Lemma 7.6 ($\iota_{\text{Base}}$-inferences Preserve Relevant Instances)** *Let $\Lambda \vdash \Phi$ be a sequent and assume an $\iota_{\text{Base}}$ inference with right (or only) premise $C \cdot \Gamma$, conclusion $C' \cdot \Gamma'$, and a ground instance via $\gamma$ of the $\iota_{\text{Base}}$ inference such that*

- *(i) $(C \cdot \Gamma)\gamma$ is a relevant instance of $C \cdot \Gamma$ wrt. $(\Lambda, R_{(C\cdot\Gamma)\gamma})$,*
- *(ii-a) in case of* Sup-Neg *or* Sup-Pos*, the left premise $(l \approx r \vee C'' \cdot \Gamma'')\gamma$ generates $(l \to r)\gamma$ in $R_{\Lambda \vdash \Phi}$ via $l \approx r \vee C'' \cdot \Gamma''$ (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* Neg-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, the conclusion $(C' \cdot \Gamma')\gamma$ of the ground inference is a relevant instance of $C' \cdot \Gamma'$ 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.

**Lemma 7.7** *If $s \approx t \vee D \cdot \Gamma$ generates $s \to t$ in $R$ then $R^\star \models s \approx t$ and $R^\star \not\models D$, and $R^\star_{\mathcal{D}} \models s \approx t$ and $R^\star_{\mathcal{D}} \not\models D$ for every $\mathcal{D} \in \Phi^{\text{gr}} \cup \Pi_\Lambda$ such that $\mathcal{D} \succ s \approx t \vee D \cdot \Gamma$.*

*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 $R \setminus (R_{s \approx t \vee D \cdot \Gamma} \cup \{s \to t\})$ is larger than every term occurring in a positive literal of $D$, hence these rules cannot be used to rewrite terms in positive literals of $D$. Therefore, if a literal of $D$ is false in $(R_{s \approx t \vee D \cdot \Gamma} \cup \{s \to t\})^\star$, 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 $R_{\mathcal{D}} \supseteq R_{s \approx t \vee D \cdot \Gamma}$ as $\mathcal{D} \succ s \approx t \vee D \cdot \Gamma$. □

**Lemma 7.8** *If $R^\star_{C \cdot \Gamma} \models C$ then $R^\star \models C$ and $R^\star_{\mathcal{D}} \models C$ for every $\mathcal{D} \in \Phi^{\mathrm{gr}} \cup \Pi_\Lambda$ such that $\mathcal{D} \succeq C \cdot \Gamma$.*

*Proof.* The left hand side of every rewrite rule in $R \setminus R_{C \cdot \Gamma}$ is larger than every term occurring in a negative literal of $C$, hence these rules cannot be used to rewrite terms in negative literals of $C$. Therefore, if a literal of $C$ is true in $R^\star_{C \cdot \Gamma}$, 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 $R_{\mathcal{D}} \supseteq R_{C \cdot \Gamma}$ as $\mathcal{D} \succeq C \cdot \Gamma$. □

**Corollary 7.9** *If $\Lambda, R_{C \cdot \Gamma} \models C \cdot \Gamma$ then $\Lambda, R \models C \cdot \Gamma$ and $\Lambda, R_{\mathcal{D}} \models C \cdot \Gamma$ for every $\mathcal{D} \in \Phi^{\mathrm{gr}} \cup \Pi_\Lambda$ such that $\mathcal{D} \succeq C \cdot \Gamma$.*

*Proof.* Suppose $\Lambda, R_{C \cdot \Gamma} \models C \cdot \Gamma$. If $\Lambda \not\models \Gamma$ the claim is trivial. Hence suppose $\Lambda \models \Gamma$. By definition, $R^\star_{C \cdot \Gamma} \models C$. With Lemma 7.8 conclude $R^\star \models C$ and, trivially, $\Lambda, R \models C \cdot \Gamma$. Similarly for the second part. □

## 8 Redundancy, Saturation and Static Completeness

To define concepts of redundancy we need a specific notion of relevant instances that takes the model construction into account. We extend Definition 7.1 and say that $(C \cdot \Gamma)\gamma$ is a *relevant instance of $C \cdot \Gamma$ wrt. $\Lambda$* iff $(C \cdot \Gamma)\gamma$ is a relevant instance of $C \cdot \Gamma$ wrt. $(\Lambda, R_{(C \cdot \Gamma)\gamma})$. Relevancy of an instance $(C \cdot \Gamma)\gamma$ wrt. $\Lambda$ thus does not depend on rules from $R \setminus R_{(C \cdot \Gamma)\gamma}$.

**Definition 8.1 (Relevant Instances wrt. $\Lambda$)** *Let $\Phi$ be a set of constrained clauses. Define*

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

*Let $\Lambda \vdash \Phi$ be a sequent and $\mathcal{D}$ a ground constrained clause. Define*

$$\Phi^\Lambda_{\mathcal{D}} = \{C \cdot \Gamma \in \Phi^\Lambda \mid \mathcal{D} \succ C \cdot \Gamma\} \ .$$

In words, $\Phi^\Lambda_{\mathcal{D}}$ is the set of relevant instances wrt. $\Lambda$ of all constrained clauses from $\Phi$ that are all smaller wrt. $\succ$ than $\mathcal{D}$.

**Definition 8.2 (Redundant Constrained Clause)** *Let $\Lambda \vdash \Phi$ be a sequent and $C \cdot \Gamma$ a ground constrained clause. We say that $C \cdot \Gamma$ is* redundant wrt. $\Lambda \vdash \Phi$ and $\mathcal{D}$ *iff* $\Phi^\Lambda_{\mathcal{D}} \models_\Lambda C \cdot \Gamma$, *and we say that $C \cdot \Gamma$ is* redundant wrt. $\Lambda \vdash \Phi$ *iff $C \cdot \Gamma$ is redundant wrt. $\Lambda \vdash \Phi$ and $C \cdot \Gamma$.*

In words, $C \cdot \Gamma$ is redundant wrt. $\Lambda \vdash \Phi$ and $\mathcal{D}$ iff $C \cdot \Gamma$ is entailed wrt. $\Lambda$ by relevant instances wrt. $\Lambda$ of clauses in $\Phi$ that are smaller than $\mathcal{D}$.

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

**Lemma 8.3** *Let $\Lambda \vdash \Phi$ be a sequent and $(C \cdot \Gamma)\gamma$ a ground instance 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 $R_{\Lambda \vdash \Phi}$ via $C \cdot \Gamma$.*

*Proof.* Suppose that $(C \cdot \Gamma)$ 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 instance wrt. $\Lambda$, then by property 5 of the definition of model construction $(C \cdot \Gamma)\gamma$ cannot generate a rewrite rule. Otherwise, by relevancy, $\Lambda$ 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 $\Lambda$ let $\mathrm{grd}(\Lambda)$ denote the set of all ground literals in $\Lambda$.

**Definition 8.4 (Universal Redundancy)** *Let $\Lambda \vdash \Phi$ be a sequent, $\mathcal{D}$ a ground constrained clause, and $\gamma$ a ground substitution for a constrained clause $C \cdot \Gamma$. We say that $(C \cdot \Gamma)\gamma$ is* universally redundant wrt. $\Lambda \vdash \Phi$ and $\mathcal{D}$, *iff there exists an $L \in \Gamma$ such that $\overline{L}\gamma \in \mathrm{grd}(\Lambda)$, or there exist ground instances $(C_i \cdot \Gamma_i)\gamma_i$ of constrained clauses $C_i \cdot \Gamma_i \in \Phi$ such that*

*(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) $\mathcal{D} \succ (C_i \cdot \Gamma_i)\gamma_i$ for every $i$,*

*(iii) $C_1\gamma_1 \ldots C_n\gamma_n \models C\gamma$, and*

*(iv) if $x \in \mathcal{V}ar(C_i) \cap \mathcal{V}ar(\Gamma_i)$, then there exists a $y \in \mathcal{V}ar(C) \cap \mathcal{V}ar(\Gamma)$ such that $x\gamma_i = y\gamma$.*

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 $(C \cdot \Gamma)\gamma$, 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 substitution $\gamma$ for $C \cdot \Gamma$.

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.

Also, a constrained clause $C \cdot \Gamma'$ is universally redundant wrt. any sequent containing a constrained clause $C \cdot \Gamma$ such that $\Gamma \subset \Gamma'$. This can be exploited to finitely bound the number of derivable constrained clauses under certain conditions. For instance, if the clause parts cannot grow in length, e.g., by disabling superposition by labelling all atoms as split atoms, and if the term depth is limited, too, then $\mathcal{ME}$+Sup derivations can be finitely bounded. In other words, $\mathcal{ME}$+Sup can be used as a decision procedure for Bernays-Schönfinkel formulas with equality.

Observe that Definition 8.4 refers to a context $\Lambda$ only be testing if *ground* rewrite literals are contained in it, a property that is preserved as $\Lambda$ grows. We obtain the following general result.

**Lemma 8.5** *If $C \cdot \Gamma$ is universally redundant wrt. $\Lambda \vdash \Phi$, $\Lambda' \supseteq \Lambda$, and $\Phi'$ is obtained from $\Phi$ 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. 8.4 that a clause that is universally redundant wrt. $\Lambda \vdash \Phi$ remains universally redundant if arbitrary constrained clauses are added to $\Phi$ or if literals are added to $\Lambda$.

To prove that a clause that is universally redundant wrt. $\Lambda \vdash \Phi$ remains universally redundant if universally redundant clauses are deleted from $\Phi$, it suffices to show that the clauses $C_i \cdot \Gamma_i \in \Phi$ in Definition 8.4 can always be chosen in such a way that they are not themselves universally redundant: Suppose that a ground constrained clause $(C \cdot \Gamma)\gamma$ is universally redundant wrt. $\Lambda \vdash \Phi$ and $\mathcal{D}$. If there exists an $L \in \Gamma$ such that $\overline{L}\gamma \in \mathrm{grd}(\Lambda)$, then deleting clauses from $\Phi$ 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 instances of clauses in $\Phi$ (wrt. the multiset extension of the clause ordering) that satisfies the conditions of Definition 8.4. 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 $\Lambda$ is assumed to be non-contradictory, this contradicts the fact that $L \in \mathrm{grd}(\Lambda)$, or there exist ground instances $(C_{1i} \cdot \Gamma_{1i})\gamma_{1i}$ of constrained clauses $C_{1i} \cdot \Gamma_{1i} \in \Phi$ that satisfy the conditions of Definition 8.4 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 8.4 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 8.6** *If $(C \cdot \Gamma)\gamma$ is universally redundant wrt. $\Lambda \vdash \Phi$ and $\mathcal{D}$, and $(C \cdot \Gamma)\gamma$ is a relevant instance of $C \cdot \Gamma$ wrt. $(\Lambda, R_{\mathcal{D}})$, then $(C \cdot \Gamma)\gamma$ is redundant wrt. $\Lambda \vdash \Phi$ and $\mathcal{D}$.*

*Proof.* Assume that $(C \cdot \Gamma)\gamma$ is universally redundant wrt. $\Lambda \vdash \Phi$ and $\mathcal{D}$ and that $(C \cdot \Gamma)\gamma$ is a relevant instance of $C \cdot \Gamma$ wrt. $(\Lambda, R_{\mathcal{D}})$. Then $\Lambda$ produces $\Gamma$ 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 instances $(C_i \cdot \Gamma_i)\gamma_i$ of constrained clauses $C_i \cdot \Gamma_i \in \Phi$ such that $C_1\gamma_1 \dots C_n\gamma_n \models C\gamma$; from property (i) we conclude that $(C \cdot \Gamma)\gamma$ is entailed by $\{(C_1 \cdot \Gamma_1)\gamma_1, \dots, (C_n \cdot \Gamma_n)\gamma_n\}$ wrt. $\Lambda$, i.e., $(C_1 \cdot \Gamma_1)\gamma_1, \dots, (C_n \cdot \Gamma_n)\gamma_n \models_\Lambda (C \cdot \Gamma)\gamma$. It remains to show that $(C_i \cdot \Gamma_i)\gamma_i$ is a relevant instance of $(C_i \cdot \Gamma_i)$ wrt. $\Lambda$, for all $i = 1, \dots, n$. First, by property (i) again we get that $\Lambda$ produces $\Gamma_i$ and $\Gamma_i\gamma_i$ by the same literals. Second, because $(C \cdot \Gamma)\gamma$ is a relevant instance of $C \cdot \Gamma$ wrt. $(\Lambda, R_{\mathcal{D}})$, $y\gamma$ is irreducible wrt. $R_{\mathcal{D}}$ for every $y \in \mathcal{V}ar(C) \cap \mathcal{V}ar(\Gamma)$. By property (ii), each $(C_i \cdot \Gamma_i)\gamma$ is smaller than $\mathcal{D}$. It follows $R_{(C_i \cdot \Gamma_i)\gamma} \subseteq R_{\mathcal{D}}$. By property (iv) then, $x\gamma_i$ is irreducible wrt. $R_{(C_i \cdot \Gamma_i)\gamma}$ for every $x \in \mathcal{V}ar(C_i) \cap \mathcal{V}ar(\Gamma_i)$, which suffices to complete the proof. □

**Corollary 8.7** *If $C \cdot \Gamma$ is universally redundant wrt. $\Lambda \vdash \Phi$, then every relevant instance of $C \cdot \Gamma$ wrt. $\Lambda$ is redundant wrt. $\Lambda \vdash \Phi$.*

*Proof.* Suppose that $C \cdot \Gamma$ is universally redundant wrt. $\Lambda \vdash \Phi$, i.e., that $(C \cdot \Gamma)\gamma$ is universally redundant wrt. $\Lambda \vdash \Phi$ and $(C \cdot \Gamma)\gamma$, for every ground substitution $\gamma$ for $C \cdot \Gamma$. Let $\gamma$ be any ground substitution for $C \cdot \Gamma$ such that $(C \cdot \Gamma)\gamma$ is a relevant instance of $C \cdot \Gamma$ wrt. $\Lambda$. By setting $\mathcal{D} = (C \cdot \Gamma)\gamma$ the result follows immediately from immediate from Lemma 8.6. □

The restriction to relevant instances in Corollary 8.7 can not be dropped. For example, with $C \cdot \Gamma = y \approx b \vee y \approx b \cdot P(y)$ and $\gamma = \{y \mapsto a\}$ the instance $(C \cdot \Gamma)\gamma$ is universally redundant wrt. $P(x) \vdash (x \approx b \cdot P(x)), a \approx c$ (take the instance $a \approx b \cdot P(a)$ of $x \approx b \cdot P(x)$ to show that), but $(C \cdot \Gamma)\gamma$ is not redundant wrt. this sequent, as the instance $a \approx b \cdot P(a)$ needed to establish that, is not a *relevant* instance, as $x\gamma = a$ is reducible wrt. $a \to c \in R_{a \approx b \cdot P(a)}$.

Lemma 8.3 also holds in terms of universal redundancy.

**Corollary 8.8** *Let $\Lambda \vdash \Phi$ be a sequent and $(C \cdot \Gamma)\gamma$ a ground instance 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 $R_{\Lambda \vdash \Phi}$ via $C \cdot \Gamma$.*

*Proof.* Suppose that $(C \cdot \Gamma)\gamma$ is universally redundant wrt. $\Lambda \vdash \Phi$. If $(C \cdot \Gamma)\gamma$ is not a relevant instance wrt. $\Lambda$, 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 instance wrt. $\Lambda$, then Corollary 8.7 $(C \cdot \Gamma)\gamma$ is redundant wrt. $\Lambda \vdash \Phi$. Now apply Lemma 8.3. □

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

**Definition 8.9 (Universally Redundant $\iota_{\mathcal{ME}+\mathsf{Sup}}$ Inference)** *Let $\Lambda \vdash \Phi$ and $\Lambda' \vdash \Phi'$ be sequents. A $\iota_{\mathcal{ME}+\mathsf{Sup}}$ 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 substitution $\gamma$, $(C \cdot \Gamma)\gamma$ is*

*universally redundant wrt.* $\Lambda' \vdash \Phi'$*, or the following holds, depending on the inference rule applied:*

Deduce: *One of the following holds:*

*(i) Applying* $\gamma$ *to all premises and the conclusion* $C' \cdot \Gamma'$ *of the underlying* $\iota_{\text{Base}}$ *inference does not result in a ground instance via* $\gamma$ *of this* $\iota_{\text{Base}}$ *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*, where* $C'' \cdot \Gamma''$ *is the left premise,* $(C'' \cdot \Gamma'')\gamma$ *is universally redundant wrt.* $\Lambda' \vdash \Phi'$*.*

Split: $C \cdot \Gamma = \square \cdot \Gamma$ *and* $\Lambda'$ *does not produce* $\Gamma$*.*

Close: $C \cdot \Gamma = \square \cdot \emptyset \in \Phi'$*.*

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 8.10** *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* $\gamma$ *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 $\mathcal{D} = (C' \cdot \Gamma')\gamma$, $n = 1$, $C_1 \cdot \Gamma_1 = C \cdot \Gamma$ and $\gamma_1 = \gamma$ in Definition 8.4 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 8.11** *If a* Deduce *inference is universally redundant wrt.* $\Lambda \vdash \Phi$*,* $\Lambda' \supseteq \Lambda$*, and* $\Phi'$ *is obtained from* $\Phi$ *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 8.5. □

Summarizingly, 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 8.5), that all its relevant ground instances are redundant (and therefore cannot be minimal counterexamples in the model construction, by Corollary 8.7), and that its ground instances cannot generate rewrite rules (by Corollary 8.8). 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 8.12 (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 8.13 (Static Completeness)** *If* $\Lambda \vdash \Phi$ *is a saturated sequent with a non-contradictory context* $\Lambda$ *and* $\square \cdot \emptyset \notin \Phi$ *then the induced rewrite system* $R_{\Lambda \vdash \Phi}$ *satisfies all relevant instances of all clauses in* $\Phi$ *wrt.* $\Lambda$ *, i.e.,* $\Lambda, R_{\Lambda \vdash \Phi} \models \Phi^{\Lambda}$. *Moreover, if* $\Psi$ *is a clause set and* $\Phi$ *includes* $\Psi$*, 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 instance of a constrained clause in $\Phi$. An example is the sequent

$$\Lambda \vdash \Phi \;=\; P(f(x)),\ f(a) \to a \vdash \square \cdot P(f(x)), f(x) \to x \ .$$

Observe that Close is not applicable. Further, $\Lambda$ does not produce the constraint $\{P(f(x)), f(x) \to x\}$ and hence the Split application with selected clause $\square \cdot P(f(x)), f(x) \to x$ is universally redundant wrt. $\Lambda \vdash \Phi$. Altogether, $\Lambda \vdash \Phi$ is saturated. However, $\Lambda, R_{\Lambda \vdash \Phi} \not\models \square \cdot P(f(a)), f(a) \to a$ as $\Lambda \models \{P(f(a)), f(a) \to a\}$ and no rewrite system satisfies $\square$. Hence $\Lambda, R_{\Lambda \vdash \Phi} \not\models \square \cdot P(f(x)), f(x) \to x$. But this does not violate Theorem 8.13, as $\square \cdot P(f(a)), f(a) \to a$ is not a *relevant* instance of $\square \cdot P(f(x)), f(x) \to x$. Although $x\{x \mapsto a\}$ is irreducible wrt. $R_{\square \cdot P(f(a)), f(a) \to a} = \emptyset$, $\Lambda$ does not produce $f(x) \to x$, and hence does not produce $\{P(f(x)), f(x) \to x\}$ and $\{P(f(a)), f(a) \to a\}$ by the same literals.

*Proof.* Let $\Lambda \vdash \Phi$ be a saturated sequent with a non-contradictory context and suppose $\square \cdot \emptyset \notin \Phi$. To complete the proof of the first statement we show that every relevant instance $C \cdot \Gamma$ of a clause in $\Phi$ wrt. $\Lambda$ satisfies one of the following two properties:

(i) $\Lambda, R_{C \cdot \Gamma} \models C \cdot \Gamma$.

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

Using Lemma 7.7 and Corollary 7.9 we conclude in both cases that $\Lambda, R_{\Lambda \vdash \Phi} \models C \cdot \Gamma$ and $\Lambda, R_{D \cdot \Gamma'} \models C \cdot \Gamma$ for every ground instance $D \cdot \Gamma' \succ C \cdot \Gamma$.

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 argumentation. 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 7.1 it follows that every ground instance of a constrained clause with empty constraint is always relevant, for every pair $(\Lambda, R)$. Hence, and more formally, $(C \cdot \emptyset)\gamma \subseteq \{D \cdot \emptyset \mid D \in \Psi\}^{\Lambda}$. With $\{D \cdot \emptyset \mid D \in \Psi\} \subseteq \Phi$ conclude trivially $(C \cdot \emptyset)\gamma \subseteq \Phi^{\Lambda}$. With $\Lambda, R_{\Lambda \vdash \Phi} \models \Phi^{\Lambda}$ we get $\Lambda, R_{\Lambda \vdash \Phi} \models (C \cdot \emptyset)\gamma$, which means $\Lambda \not\models \emptyset$ or $R^{\star}_{\Lambda \vdash \Phi} \models C\gamma$, equivalently $R^{\star}_{\Lambda \vdash \Phi} \models C\gamma$.

Recall that we have to show that every relevant instance wrt. $\Lambda$ satisfies (i) or (ii). We prove this be contradiction. If there is a relevant counterexample, that is, a relevant instance wrt. $\Lambda$ of a clause in $\Phi$ that does not satisfy (i) or (ii), then, by well-foundedness of the clause ordering, there is a minimal such instance wrt. the clause ordering $\succ$. From

now on let $C \cdot \Gamma$ be such a minimal relevant counterexample. Let $C \cdot \Gamma = (C' \cdot \Gamma')\gamma$, where $C' \cdot \Gamma' \in \Phi$. By minimality of $C \cdot \Gamma$, every relevant instance wrt. $\Lambda$ of a clause in $\Phi$ that is smaller than $C \cdot \Gamma$ satisfies (i) and (ii), hence it is satisfied by $(\Lambda, R_{C \cdot \Gamma})$.

*(1) $C \cdot \Gamma$ is redundant wrt. $\Lambda \vdash \Phi$ or $C \cdot \Gamma$ is universally redundant wrt. $\Lambda \vdash \Phi$.*
If $C \cdot \Gamma$ is redundant wrt. $\Lambda \vdash \Phi$ then there are relevant instances wrt. $\Lambda$ of clauses in $\Phi$, each smaller wrt. $\succ$ than $C \cdot \Gamma$, and that entail $C \cdot \Gamma$ wrt. $\Lambda$. By the induction hypothesis, and with Corollary 7.9, all these instances are true in $(\Lambda, R_{C \cdot \Gamma})$. As they entail $C \cdot \Gamma$ wrt. $\Lambda$, we conclude that $\Lambda, R_{C \cdot \Gamma} \models C \cdot \Gamma$, hence $C \cdot \Gamma$ satisfies (i), contradicting our assumption.

Because $C \cdot \Gamma$ is assumed to be a relevant instance wrt. $\Lambda$, by Lemma 8.6 $C \cdot \Gamma$ cannot be universally redundant wrt. $\Lambda \vdash \Phi$ either.

*(2) $\mathcal{V}ar(C')\gamma$ is reducible wrt. $R_{C \cdot \Gamma}$.*
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 $\mathcal{V}ar(C')\gamma$ is reducible wrt. $R_{C \cdot \Gamma}$. Because $C \cdot \Gamma$ is a relevant instance of $C' \cdot \Gamma'$ we already know with Definition 7.1 that $(\mathcal{V}ar(C') \cap \mathcal{V}ar(\Gamma'))\gamma$ is irreducible wrt. $R_{C \cdot \Gamma}$. If $x\gamma$ is reducible for some $x \in \mathcal{V}ar(C') \setminus \mathcal{V}ar(\Gamma')$, then a term in the range of $\gamma$ can be replaced by a smaller yet congruent term wrt. $R^\star_{C \cdot \Gamma}$. Observe that this results in a smaller (wrt. $\succ$) relevant counterexample, thus contradicting the choice of $C \cdot \Gamma$.

*(3) $C = s \not\approx t \vee D$ with maximal literal $s \not\approx t$.*
Suppose that none of the preceding cases holds and $C \cdot \Gamma = s \not\approx t \vee D \cdot \Gamma$ and $s \not\approx t$ is maximal in $s \not\approx t \vee D$.

*(3.1) $s = t$.*
If $s = t$ then there is a Deduce inference with premise $C = s \not\approx t \vee D \cdot \Lambda$ and conclusion $D \cdot \Lambda$, 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 $C \cdot \Gamma$ is not universally redundant wrt. $\Lambda \vdash \Phi$, the clause $D \cdot \Gamma$ must be universally redundant wrt. $\Lambda \vdash \Phi$ and $C \cdot \Gamma$ by definition of redundant inferences. Furthermore, by Lemma 7.6, $D \cdot \Gamma$ is a relevant instance wrt. $(\Lambda, R_{C \cdot \Gamma})$, hence, by Lemma 8.6, $D \cdot \Gamma$ is redundant wrt. $\Lambda \vdash \Phi$ and $C \cdot \Gamma$ and follows from relevant instances of clauses in $\Phi$ wrt. $\Lambda$ that are smaller than $C \cdot \Gamma$. By the minimality assumption, and with Corollary 7.9, these clauses are satisfied by $(\Lambda, R_{C \cdot \Gamma})$, hence $\Lambda, R_{C \cdot \Gamma} \models D \cdot \Gamma$, and, trivially, $\Lambda, R_{C \cdot \Gamma} \models C \cdot \Gamma$, contradicting our assumption. (In other cases below we will use similar arguments without explicit reference to Lemmas 7.6, 8.6, and Corollary 7.9.)

*(3.2) $s \neq t$.*
If $s \neq t$ then without loss of generality assume $s \succ t$. With $\Lambda, R_{C \cdot \Gamma} \not\models s \not\approx t \vee D \cdot \Gamma$ it follows that $\Lambda \models \Gamma$ and $R^\star_{C \cdot \Gamma} \not\models s \not\approx t \vee D$.

From $R^\star_{C \cdot \Gamma} \not\models s \not\approx t \vee D$ it follows in particular $R^\star_{C \cdot \Gamma} \models s \approx t$. Because $R_{C \cdot \Gamma}$ is a convergent (ordered) rewrite system, $s$ and $t$ must have the same normal forms. In particular, thus, $s$ is reducible wrt. $R_{C \cdot \Gamma}$. Suppose $s = s[l]_p$ for some position $p$ and rule $l \to r \in R_{C \cdot \Gamma}$. We distinguish two cases.

*(3.2.1) $l \approx r$ is a split atom.*
If $l \approx r$ is a split atom then with $l \to r \in R_{C\cdot\Gamma}$ and Lemma 7.3-(i) it follows that $\Lambda$ produces $l \to r$. For later use let $l' \to r' \in_\sim \Lambda$ be a fresh variant of a rewrite literal that produces $l \to r$ in $\Lambda$ and assume that $\gamma$ 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 $s[l]_p \not\approx t \vee D \cdot \Gamma$ and conclusion $s[r]_p \not\approx t \vee D \cdot \Gamma, l \to r$. The next step is to show that this ground inference is a ground instance via $\gamma$ 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 $(s'[r]_p \not\approx t' \vee D' \cdot \Gamma', l' \to r')\sigma$, where mgu $\sigma$ 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 $u\gamma$ is reducible wrt. $R_{C\cdot\Gamma}$ (the rule $l \to r$ would rewrite it, as $l = u\gamma$), but we know that $\mathcal{V}ar(C')\gamma$ is irreducible wrt. $R_{C\cdot\Gamma}$.

We need to know that Deduce is applicable with the (possibly non-ground) U-Sup-Neg inference underlying it. For this, it only remains to show that $\Lambda$ produces $(l' \to r')\sigma$. This, however, follows trivially from the fact that $l' \to r' \in \Lambda$ produces $l \to r$ in $\Lambda$, 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 $\Lambda$ either).

By saturation, the Deduce inference is universally redundant wrt. $\Lambda \vdash \Phi$. Because the instance $C \cdot \Gamma$ of its premise $C' \cdot \Gamma'$ is not universally redundant wrt. $\Lambda \vdash \Phi$, the conclusion $s[r]_p \not\approx t \vee D \cdot \Gamma, l \to r$ must be universally redundant wrt. $\Lambda \vdash \Phi$ and $C \cdot \Gamma$ by definition of redundant inferences. Furthermore, the conclusion is a relevant instance, hence it is redundant wrt. $\Lambda \vdash \Phi$ and $C \cdot \Gamma$ and follows from relevant instances of clauses in $\Phi$ that are smaller than $C \cdot \Gamma$. By the minimality assumption, these clauses are satisfied by $(\Lambda, R_{C\cdot\Gamma})$, hence $\Lambda, R_{C\cdot\Gamma} \models s[r]_p \not\approx t \vee D \cdot \Gamma, l \to r$. By definition, this means $\Lambda \not\models \Gamma \cup \{l \to r\}$ or $R^\star_{C\cdot\Gamma} \models s[r]_p \not\approx t \vee D$. The next step is to show $\Lambda \models \Gamma \cup \{l \to r\}$ which allows us to conclude $R^\star_{C\cdot\Gamma} \models s[r]_p \not\approx t \vee D$.

To show $\Lambda \models \Gamma \cup \{l \to r\}$ we need to show, by Definition 4.1, that (a) $\Gamma \cup \{l \to r\}$ consists of split rewrite literals and that (b) $\Lambda$ produces $\Gamma \cup \{l \to r\}$. Recall first $\Lambda \models \Gamma$ as an assumption from case (3.2) above. But then, (a) is immediate from $\Lambda \models \Gamma$ and from $l \approx r$ being a split atom and $l \succ r$. Similarly, (b) follows from $R_{C\cdot\Gamma} \models \Gamma$ and $l \to r \in R_{C\cdot\Gamma}$, and hence $l \to r \in \Pi_\Lambda$ and so $\Lambda$ produces $l \to r$.

We can thus conclude $R^\star_{C\cdot\Gamma} \models s[r]_p \not\approx t \vee D$ now, as announced. With $l \to r \in R_{C\cdot\Gamma}$ by congruence it follows $R^\star_{C\cdot\Gamma} \models s \not\approx t \vee D$, however $R^\star_{C\cdot\Gamma} \not\models s \not\approx t \vee D$ was assumed for case (3.2) above.

*(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 instance $(C_0 \cdot \Gamma_0)\gamma$ of a constrained clause $C_0 \cdot \Gamma_0 \in \Phi$ instead of a rewrite literal from $\Pi_\Lambda$. As we have shown in case (1), a generating clause cannot be redundant (and hence cannot be universally redundant). It follows that the constraints of both premises of the ground inferences must be produced by $\Lambda$. Their union, which is the constraint of the conclusion thus is trivially produced by $\Lambda$, too. Regarding the clause parts, the same

congruence argument using $l \to r$ as in case (3.2.1) applies. Using Lemma 7.7, we see that the remaining literals of $(C_0 \cdot \Gamma_0)\gamma$ are false wrt. $R_{C\cdot\Gamma}$, leading to a contradiction.

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

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

*(4.1.1) $s$ or $t$ is reducible wrt. $R_{C\cdot\Gamma}$.*
If $s$ or $t$ is reducible wrt. $R_{C\cdot\Gamma}$ then there is a rule $l \to r \in R_{C\cdot\Gamma}$ such that $s = s[l]_p$ or $t = t[l]_p$, for some position $p$. If $l \to r$ is generated by a rewrite literal from $\Pi_\Lambda$ 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 $s \succ t$ does not apply.

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

*(4.1.2) $s$ and $t$ are irreducible wrt. $R_{C\cdot\Gamma}$.*
If $s$ and $t$ are irreducible wrt. $R_{C\cdot\Gamma}$ then assume, w.l.o.g., $s \succ t$. The ordering on clauses and rewrite literals is defined in such a way that every constrained clause containing $s \approx t$ is greater than the rewrite literal $s \to t$. With Lemma 7.3-(ii) then conclude that $\Lambda$ produces $s \not\to t$. This indicates that a Deduce inference with an underlying ground Neg-U-Res inference exists. More precisely, the left premise of that ground inference is $s \not\to t$, the right premise is $s \approx t \vee D \cdot \Gamma$ and the conclusion is $D \cdot \Gamma, s \not\to t$. It is routine by now to check that this ground Neg-U-Res inference is a ground instance via $\gamma$ of a Neg-U-Res inference with a right premise from $\Phi$ that is not universally redundant wrt. $\Lambda \vdash \Phi$, and a left premise from $\Lambda$.

As in case (3.2.1) we can show that the Deduce inference with the latter underlying Neg-U-Res inference exists. In particular, that $\Lambda$ produces its instantiated left premise (via $\sigma$) can be shown with the same arguments. The rest of the proof uses the same arguments as in case (3.2.1), too.

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

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

*(4.2.2) $s$ is irreducible wrt. $R_{C\cdot\Gamma}$.*
In this case, either $s \approx t \vee D \cdot \Gamma$ generates $s \to t$, so property (ii) would be satisfied. Or a Fact inference exists, which provides a smaller constrained clause that, by redundancy, is true in $(\Lambda, R_{C\cdot\Gamma})$ and entails $s \approx t \vee D \cdot \Gamma$. In both cases, we get a contradiction to the assumption that $s \approx t \vee D \cdot \Gamma$ is a minimal counterexample.

*(5) $C = \square$.*

Suppose $C \cdot \Gamma = \square \cdot \Gamma$. 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 $\square \cdot \Gamma' \in \Phi$, where $\Gamma = \Gamma'\gamma$.

Because $\square \cdot \Gamma$ is a relevant instance of $\square \cdot \Gamma'$ wrt. $\Lambda$, by Definition 7.1 $\Lambda$ produces $\Gamma'$ and $\Lambda$ produces $\Gamma$ by the same literals. We are given that $\Lambda$ 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 $\Lambda$ would produce $\overline{L}$. But in this case $\Lambda$ can produce $L$ only if $L \in_\sim \Lambda$, and so $\Lambda$ 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 $\Lambda$. 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 $\Lambda$ produces $\Gamma'$. 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. $\square$

Theorem 8.13 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 8.13 will be essential then in proving the completeness of the $\mathcal{ME}$+Sup calculus.

# 9 Derivations with Simplification

To make derivation 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.

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

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 $\iota_{\mathcal{ME}+\mathsf{Sup}}$ inference, a Simp inference or a Cancel 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. Each derivation $\mathcal{D} = ((\mathbf{N}_i, \mathbf{E}_i))_{i<\kappa}$, 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 $\mathcal{D}$ is indeed a tree. But note that it will not be a derivation tree unless $\mathcal{D}$ 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 $\kappa$ nodes, and let $\Lambda_i \vdash \Phi_i$ be the sequent labeling node $N_i$, for all $i < \kappa$. Define $\Lambda_\mathbf{B} = \bigcup_{i<\kappa} \bigcap_{i\leq j<\kappa} \Lambda_j$[11] and $\Phi_\mathbf{B} = \bigcup_{i<\kappa} \bigcap_{i\leq j<\kappa} \Phi_j$, 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 9.1 (Exhausted Branch)** *Let* $\mathbf{T}$ *be a limit tree and* $\mathbf{B} = (N_i)_{i<\kappa}$ *a branch in* $\mathbf{T}$ *with* $\kappa$ *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* $\iota_{\mathcal{ME}+\mathsf{Sup}}$ *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* $j \geq i$*, and*

*(ii)* $\square \cdot \emptyset \notin \Phi_\mathbf{B}$

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 9.1, 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

[11] 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 $\bigcup_{i<\kappa} \Lambda_i$.

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 8.10 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 9.2** *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 8.5 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 8.5, 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 8.5 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 $\Lambda_j$ is also contained in $\Lambda_\mathbf{B}$, the claim of the lemma follows immediately. □

**Lemma 9.3** *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 9.2 using Lemma 8.11. □

**Proposition 9.4 (Exhausted Branches are Saturated)** *If $\mathbf{B}$ is an exhausted branch of a limit tree of a fair derivation then $\Lambda_\mathbf{B} \vdash \Phi_\mathbf{B}$ is saturated.*

*Proof.* Suppose $\mathbf{B}$ 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 9.1 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 $\Box \cdot \Gamma$ 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 $\Gamma$. 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 9.1 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 $\Lambda_j$ does not produce $\Gamma$. In the first case, use Lemma 9.2 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 $\Gamma$. By fairness, for each point $j_l$ there must be a later point $j'_l$ such that $\Lambda_{j'_l}$ does not produce $\Gamma$. 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 $k \geq j_l$, $\Lambda_k$ does not produce $\Gamma$. But then, $\Lambda_{\mathbf{B}}$ does not produce $\Gamma$ either, which entails that $\square \cdot \Gamma$ is universally redundant wrt. $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$, and we are done, as in first case.

If the inference rule is Deduce then by Definition 9.1 it is universally redundant wrt. $\Lambda_j \vdash \Phi_j$, for some $j \geq i$, and by Lemma 9.3 it is universally redundant wrt. $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$. $\square$

Proposition 9.4 is instrumental in the proof of our main result, which is the following.

**Theorem 9.5 (Completeness)** *Let $\Psi$ be a clause set and $\mathbf{T}$ be the limit tree of a fair derivation of $\Psi$. If $\mathbf{T}$ is not a refutation tree then $\Psi$ 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 9.4 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 8.13 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 $\Psi$, 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 8.13 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 $\Phi_k$ 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 9.2, $C \cdot \emptyset$ is universally redundant wrt. $\Lambda_{\mathbf{B}} \vdash \Phi_{\mathbf{B}}$. Since $C \cdot \emptyset$ has an empty constraint, all its ground instances are relevant, and by Corollary 8.7, all relevant instances 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 Fact inference rules are sound in the standard sense by taking the clausal forms of the premises and the conclusions. For the remaining $\iota_{\text{Base}}$ inference rules U-Sup-Pos, U-Sup-Neg and Neg-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 completeness proof can be carried out as for the $\mathcal{ME}_{\text{E}}$ calculus [BT05].

## 10 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 of an earlier version of this paper and Cesare Tinelli for their helpful comments.

## References

[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.

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

[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.

# A Proof of Lemma 7.6

**Lemma 7.6 ($\iota_{\text{Base}}$-inferences Preserve Relevant Instances)** *Let $\Lambda \vdash \Phi$ be a sequent and assume an $\iota_{\text{Base}}$ inference with right (or only) premise $C \cdot \Gamma$, conclusion $C' \cdot \Gamma'$, and a ground instance via $\gamma$ of the $\iota_{\text{Base}}$ inference such that*

- *(i) $(C \cdot \Gamma)\gamma$ is a relevant instance of $C \cdot \Gamma$ wrt. $(\Lambda, R_{(C\cdot\Gamma)\gamma})$,*
- *(ii-a) in case of* Sup-Neg *or* Sup-Pos*, the left premise $(l \approx r \vee C'' \cdot \Gamma'')\gamma$ generates $(l \to r)\gamma$ in $R_{\Lambda \vdash \Phi}$ via $l \approx r \vee C'' \cdot \Gamma''$ (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* Neg-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, the conclusion $(C' \cdot \Gamma')\gamma$ of the ground inference is a relevant instance of $C' \cdot \Gamma'$ wrt. $(\Lambda, R_{(C\cdot\Gamma)\gamma})$, for some possibly different merge substitution in case of a* Sup-Neg *or* Sup-Pos *inference.*

*Proof.* For convenience we abbreviate $R := R_{(C\cdot\Gamma)\gamma}$ below.

With (i), by Definition 7.1, $\Gamma\gamma$ consists of split rewrite literals, $\Lambda$ produces $\Gamma$ and $\Lambda$ produces $\Gamma\gamma$ by the same instances, and $(\mathcal{V}ar(C) \cap \mathcal{V}ar(\Gamma))\gamma$ is irreducible wrt. $R$. We have to show

(1) $\Gamma'\gamma$ consists of split rewrite literals,

(2) $\Lambda$ produces $\Gamma'$ and $\Lambda$ produces $\Gamma'\gamma$ by the same instances, and

(3) $(\mathcal{V}ar(C') \cap \mathcal{V}ar(\Gamma'))\gamma$ is irreducible wrt. $R$.

The proof of (1) follows easily from inspection of the $\iota_{\text{Base}}$ 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 Neg-U-Res, as the only rules that add new rewrite literals, come with conditions that force that for the new rewrite literals.

It remains to show (2) and (3).

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

For later use we prove the following *facts*:

(i) $\Lambda$ produces $\Gamma\sigma$ ($\Gamma\sigma\pi$) and $\Lambda$ 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 $\Lambda$. If $K$ didn't produce $L\sigma$ in $\Lambda$ 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 $\Lambda$ produces $\Gamma\sigma\pi$ and $\Lambda$ produces $\Gamma\sigma\pi\gamma$ by the same literals is the same after replacing $\sigma$ by $\sigma\pi$.

(ii) For every term $t$, if $\mathcal{V}ar(t)\gamma$ is irreducible wrt. $R$ then $\mathcal{V}ar(t\sigma)\gamma$ ($\mathcal{V}ar(t\sigma\pi)\gamma$) is irreducible wrt. $R$.

*Proof:* Let $t$ be a term and suppose $\mathcal{V}ar(t)\gamma$ is irreducible wrt. $R$. Chose a variable $y \in \mathcal{V}ar(t\sigma)$ arbitrarily. It suffices to show that $y\gamma$ is irreducible wrt. $R$. With $y \in \mathcal{V}ar(t\sigma)$, $y$ must occur in a term $x\sigma$, for some variable $x \in \mathcal{V}ar(t)$ ($y = x$ 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 $\mathcal{V}ar(t)\gamma$ irreducible wrt. $R$. With $x \in \mathcal{V}ar(t)$, $x\gamma$ is irreducible wrt. $R$, and it is clear that its subterm $y\gamma$ then irreducible wrt. $R$, too.

The proof that $\mathcal{V}ar(t\sigma\pi)\gamma$ is irreducible wrt. $R$ is the same after replacing $\sigma$ by $\sigma\pi$.

(iii) $(\mathcal{V}ar(C\sigma) \cap \mathcal{V}ar(\Gamma\sigma))\gamma$ is irreducible wrt. $R$.

*Proof:* From above we know that $(\mathcal{V}ar(C) \cap \mathcal{V}ar(\Gamma))\gamma$ is irreducible wrt. $R$. If $x \in \mathcal{V}ar(C) \cap \mathcal{V}ar(\Gamma)$ take $t = x$ and conclude with fact (ii) that $\mathcal{V}ar(x\sigma)\gamma$ is irreducible wrt. $R$. Because this holds for every $x \in \mathcal{V}ar(C) \cap \mathcal{V}ar(\Gamma)$ we get that $\mathcal{V}ar((\mathcal{V}ar(C) \cap \mathcal{V}ar(\Gamma))\sigma)\gamma$ is irreducible wrt. $R$. The next step is to show $\mathcal{V}ar((\mathcal{V}ar(C) \cap \mathcal{V}ar(\Gamma))\sigma) = \mathcal{V}ar(C\sigma) \cap \mathcal{V}ar(\Gamma\sigma)$. The claim then follows immediately.

It is not difficult to see $\mathcal{V}ar((\mathcal{V}ar(C) \cap \mathcal{V}ar(\Gamma))\sigma) \subseteq \mathcal{V}ar(C\sigma) \cap \mathcal{V}ar(\Gamma\sigma)$. 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 $x_C \in \mathcal{V}ar(C) \setminus \mathcal{V}ar(\Gamma)$ and $x_\Gamma \in \mathcal{V}ar(\Gamma) \setminus \mathcal{V}ar(C)$ such that $y \in \mathcal{V}ar(x_C\sigma)$ and $y \in \mathcal{V}ar(x_\Gamma\sigma)$. However, recall that $\sigma$ (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 $x_\Gamma$ cannot be moved by $\sigma$ (or $\sigma\pi$), and that $x_\Gamma$ is not in the codomain of $\sigma$. 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 $(\mathcal{V}ar(C\sigma\pi) \cap \mathcal{V}ar(\Gamma\sigma\pi))\gamma$ is irreducible wrt. $R$.

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

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

In case of a U-Sup-Neg inference let the left premise be $l \to r$, the right premise $C \cdot \Gamma = s[u]_p \approx t \vee C'' \cdot \Gamma$ and the conclusion $C' \cdot \Gamma' = (s[r]_p \approx t \vee C'' \cdot \Gamma, l \to r)\sigma$. First we show (2), i.e., that $\Lambda$ produces $(\Gamma, l \to r)\sigma$ and $\Lambda$ produces $(\Gamma, l \to r)\sigma\gamma$ 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 that $(l \to r)\sigma$ generates $(l \to r)\gamma$ in $R_{\Lambda \vdash \Phi}$, and hence $l \to r \in R$. By Lemma 7.3-(i) then $\Lambda$ produces $(l \to r)\gamma$, that is, some literal $K \in \Lambda$ produces $(l \to r)\gamma$ in $\Lambda$. It remains to show that $K$ produces $(l \to r)\sigma$ in $\Lambda$. With $K$ producing $(l \to r)\gamma$ in $\Lambda$, 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 $(\mathcal{V}ar((s[r]_p \approx t \vee C'')\sigma) \cap \mathcal{V}ar((\Gamma, l \to r)\sigma))\gamma$ is irreducible wrt. $R$. Fact (iii) above only gives us that $(\mathcal{V}ar((s[u]_p \approx t \vee C'')\sigma) \cap \mathcal{V}ar(\Gamma\sigma))\gamma$ is irreducible wrt. $R$. To get the desired result from that it is enough to show that $\mathcal{V}ar(r\sigma)\gamma$ and $\mathcal{V}ar(l\sigma)\gamma$ are irreducible wrt. $R$, because only $r\sigma$ and $l\sigma$ (via $(l \to r)\sigma$) can contribute additional variables beyond those in $\mathcal{V}ar((s[u]_p \approx t \vee C'')\sigma) \cap \mathcal{V}ar(\Gamma\sigma)$.

Regarding $\mathcal{V}ar(r\sigma)\gamma$, with fact (ii) it suffices to show that $\mathcal{V}ar(r)\gamma$ 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 $\mathcal{V}ar(l\sigma)\gamma$ we use a different argumentation. Another consequence of $(l \to r)\gamma$ being generated is that $(l \to r)\gamma \in R$. However, $(l \to r)\gamma \in R$ reduces $l\gamma$, but it can be shown that no other rule in $R$ greater than $(l \to r)\gamma$ can reduce $l\gamma$. This is, because that rules left hand side would have to be $l\gamma$, but with $(l \to r)\gamma \in R$ it would be reducible by a smaller rule, and thus it would not have been generated, and thus not be in $R_{\Lambda \vdash \Phi}$. An important detail now is that the target term $u$ unified with $l$ by $\sigma$ in the non-ground inference is not a variable. Every variable $x \in \mathcal{V}ar(l\sigma)$ 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 those rules that are already in $R_{(l \to r)\gamma}$. Because $(l \to r)\gamma$ is generated, we know that $l\gamma$ is irreducible wrt. $R_{(l \to r)\gamma}$. Together this entails that $x\gamma$, and hence $\mathcal{V}ar(l\sigma)$ is irreducible wrt. $R_{(l \to r)\gamma}$, and also wrt. $R$.

The proof for the case of a U-Sup-Pos inference is the same.

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

With respect to $(s \not\to t)\sigma$ and $(s \not\to t)\sigma\gamma$ recall we are given $s\gamma$ and $t\gamma$ are irreducible wrt. $R$. The ordering on clauses and rewrite literals is defined in such a way that every constrained clause containing $s \not\approx t$ is greater than the rewrite literal $s \to t$. By Lemma 7.3-(ii) then $\Lambda$ produces $(s \not\to t)\gamma$, that is, some literal $K \in \Lambda$ produces $(s \not\to t)\gamma$

in $\Lambda$. It remains to show that $K$ produces $(s \not\approx t)\sigma$ in $\Lambda$. With $K$ producing $(s \not\approx t)\gamma$ in $\Lambda$, 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 $(\mathcal{V}ar(C''\sigma) \cap \mathcal{V}ar((\Gamma, l \not\approx r)\sigma))\gamma$ is irreducible wrt. $R$. Fact (iii) above only gives us that $(\mathcal{V}ar((s \approx t \vee C'')\sigma) \cap \mathcal{V}ar(\Gamma\sigma))\gamma$ is irreducible wrt. $R$. But only $r\sigma$ and $l\sigma$ (via $(l \not\approx r)\sigma$) can contribute additional variables beyond those in $\mathcal{V}ar((s \approx t \vee C'')\sigma) \cap \mathcal{V}ar(\Gamma\sigma)$. It is therefore enough to show that $\mathcal{V}ar(r\sigma)\gamma$ and $\mathcal{V}ar(l\sigma)\gamma$ 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).

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

For (2) we need to show that $\Lambda$ produces $(\Gamma, \Gamma''')\sigma\pi$ and $\Lambda$ produces $(\Gamma, \Gamma''')\sigma\pi\gamma$ 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, this time starting with $\Lambda$ produces $\Gamma'''$ and $\Lambda$ produces $\Gamma'''\gamma$ by the same literals, yielding that $\Lambda$ produces $\Gamma'''\sigma\pi$ and $\Lambda$ produces $\Gamma'''\sigma\pi\gamma$ by the same literals.

Before proceeding with (3), 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 $(\mathcal{V}ar((s[r]_p \approx t \vee C'' \vee C''')\sigma\pi) \cap \mathcal{V}ar((\Gamma, \Gamma''')\sigma\pi))\gamma$ is irreducible wrt. $R$.

The first sub-proof is to show that $(\mathcal{V}ar(C'''\sigma\pi) \cap \mathcal{V}ar(\Gamma'''\sigma\pi))\gamma$ is irreducible wrt. $R$. From above we (only) know so far that $(\mathcal{V}ar(l \approx r \vee C''') \cap \mathcal{V}ar(\Gamma'''))\gamma$ and hence $(\mathcal{V}ar(C''') \cap \mathcal{V}ar(\Gamma'''))\gamma$ 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 $\pi$ 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 $\mathcal{V}ar(x_i\sigma\pi)\gamma$ 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 $\mathcal{V}ar(C'''\sigma\pi)\gamma$ left that can be reduced by $l \to r$, or any other rule in $R \setminus R_{(l \approx r \vee C''' . \Gamma''')\gamma}$. It follows trivially that $(\mathcal{V}ar(C'''\sigma\pi) \cap \mathcal{V}ar(\Gamma'''\sigma\pi))\gamma$ is irreducible wrt. $R \setminus R_{(l \approx r \vee C''' . \Gamma''')\gamma}$. To complete the first sub-proof it remains to be shown that $(\mathcal{V}ar(C'''\sigma\pi) \cap \mathcal{V}ar(\Gamma'''\sigma\pi))\gamma$ is irreducible wrt. $R_{(l \approx r \vee C''' . \Gamma''')\gamma}$. This, however, can be done with the same arguments as in fact (iii), however using the rewrite system $R_{(l \approx r \vee C''' . \Gamma''')\gamma}$ instead of $R$, and starting from $(\mathcal{V}ar(l \approx r \vee C''') \cap \mathcal{V}ar(\Gamma'''))\gamma$ being irreducible wrt. $R_{(l \approx r \vee C''' . \Gamma''')\gamma}$

Fact (iii) above gives us that $(\mathcal{V}ar((s[u]_p \approx t \vee C'')\sigma\pi) \cap \mathcal{V}ar(\Gamma\sigma\pi))\gamma$ is irreducible wrt. $R$. This result can be combined with the result from the first sub-proof to obtain that $(\mathcal{V}ar((s[u]_p \approx t \vee C'' \vee C''')\sigma\pi) \cap \mathcal{V}ar((\Gamma, \Gamma''')\sigma\pi))\gamma$ 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 $(\mathcal{V}ar((s[r]_p \approx t \vee C'' \vee C''')\sigma\pi) \cap \mathcal{V}ar((\Gamma, \Gamma''')\sigma\pi))\gamma$ is irreducible wrt. $R$. Observe that any additional variables in the latter set must stem from $\mathcal{V}ar(r\sigma\pi)$. It is therefore enough to show that $(\mathcal{V}ar(r\sigma\pi) \cap \mathcal{V}ar(\Gamma'''\sigma\pi))\gamma$ is irreducible wrt. $R$.

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

In case of a Sup-Pos inference the proof is exactly the same.

Finally, in case of a Fact inference observe that $\mathcal{V}ar(l \approx t \vee r \not\approx t \vee C) \subseteq \mathcal{V}ar(l \approx r \vee s \approx t \vee C)$. The proof then follows easily with facts (i) and (iii). □